DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Advanced Encryption Schemes from Novel Lattice Assumptions

A Thesis

Submitted by

SIMRAN KUMARI

For the award of the degree

Of

DOCTOR OF PHILOSOPHY

May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
CHENNAI – 600036

Advanced Encryption Schemes from Novel Lattice Assumptions

A Thesis

Submitted by

SIMRAN KUMARI

For the award of the degree

Of

DOCTOR OF PHILOSOPHY

May 2025

For my brother.

THESIS CERTIFICATE

This is to undertake that the Thesis titled **ADVANCED ENCRYPTION SCHEMES FROM NOVEL LATTICE ASSUMPTIONS**, submitted by me to the Indian Institute of Technology Madras, for the award of **Doctor of Philosophy**, is a bonafide record of the research work done by me under the supervision of **Prof. Shweta Agrawal**. The contents of this Thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Chennai 600036 **Simran Kumari**

Date: May 11, 2025

Prof. Shweta Agrawal
Research advisor
Professor
Department of CSE
IIT Madras

LIST OF PUBLICATIONS

I. PUBLICATIONS FROM THE THESIS

1. Shweta Agrawal, Simran Kumari, Shota Yamada. Pseudorandom FE and iO with Applications. In TCC, 2025. (To appear).

2. Shweta Agrawal, Simran Kumari, Shota Yamada. Attribute Based Encryption for Turing Machines from Lattices. In CRYPTO, 2024. https://doi.org/10.1007/978-3-031-68382-4_11.

3. Shweta Agrawal, Simran Kumari, Anshu Yadav, Shota Yamada. Trace and Revoke with Optimal Parameters from Polynomial Hardness. In EUROCRYPT, 2023. https://doi.org/10.1007/978-3-031-30620-4_20.

II. OTHER MANUSCRIPTS

1. Shweta Agrawal, Simran Kumari, Ryo Nishimaki. Laconic Pre-Constrained Encryption. Cryptology ePrint Archive. https://eprint.iacr.org/2024/1294

ACKNOWLEDGEMENTS

This PhD journey has been one of the most defining chapters of my life – filled with moments of excitement, self-doubt, and perseverance. There were days of triumph and days of frustration, but through it all, I was fortunate to have people who believed in me even when I struggled to believe in myself. I take this opportunity to thank all who stood by me, encouraged me, and contributed to my growth during this PhD journey.

First and foremost, I express my gratitude to Prof. Shweta Agrawal, without whom this journey would not have been possible. From the very beginning, it was her immense efforts that set me on this path, and throughout, she has been a constant source of support, patience, and inspiration. Even through my many missteps, she was consistently patient and kind – never giving up on me, always encouraging me to grow, and guiding me with clarity through both professional and personal challenges. Her commitment to the growth of her students, willingness to help anyone who seeks it, dedication to strengthening the cryptography community in India, and relentless efforts to support women in science are truly admirable. If I can contribute even a fraction of what she has, it would be one of my greatest achievements.

I sincerely thank all my co-authors: Prof. Shweta Agrawal, Dr. Ryo Nishimaki from NTT Tokyo, Dr. Anshu Yadav from IST Austria, and Dr. Shota Yamada from AIST Tokyo. Shweta has introduced me to many exciting research problems – several of which now form the foundation of this thesis. Her sharp insights and engaging discussions have constantly pushed me to think deeper, and I know I will continue learning from her. I am greatly indebted to Anshu, Shota, and Ryo for enriching my research journey with many insightful technical discussions and guidance in writing papers. I would also like to express my gratitude to Dr. Venkata Koppula, Dr. Ryo Nishimaki, and Dr. Sikhar Patranabis for their valuable discussions during my internships at IIT Delhi, NTT Tokyo, and IBM Research Lab Bangalore, respectively.

I am grateful to my doctoral committee (DC) members: Prof. John Ebenezer Augustine, Dr. Venkata Koppula, Prof. Krishna M. Sivalingam, and Dr. Aishwarya Thiruvengadam – for their valuable guidance and feedback. A special thanks to Mrs. Jeyanthi Arumugam for her support with administrative matters, making numerous processes much smoother. I would also like to acknowledge Google Research for funding my travel to EUROCRYPT 2023 and CRYPTO 2024.

Beyond academics, IIT Madras has gifted me not just friendships, but bonds that feel like home. Among them, Anshu holds a special place. She has been there from my very first steps in cryptography – guiding me through my first proof, my first research paper, and many more firsts that followed. But beyond equations and theorems, she has been a constant – a friend whose presence brought comfort and strength. Akhil has been another anchor through this journey. From deep cryptographic discussions to impromptu conversations about music and drama, he has been a steady presence, always ready with unfiltered opinions and unwavering support. I cannot imagine my PhD life without the warmth and laughter that Anshu and Akhil brought into it. I also cherish the memories I have shared with Nibedita, Aditya and Ishika – late-night snack runs, discovering amazing food across Chennai and Bengaluru, shopping detours that were supposed to be *quick*, and conversations that effortlessly stretched into hours. Our shared meals and small getaways brought so much joy and lightness to an otherwise overwhelming journey. I would also like to thank Antonio and Rohin for various fun outings and coffee breaks.

I am grateful to the IIT Madras campus – my silent companion through this journey. From cycling through its quiet roads at night with friends – to long conversations under the open sky, the campus has been a space for reflection and connection. Some of my most cherished moments have been walking on its scenic paths – sometimes in deep discussions with Anshu, sometimes alone, lost in thought, seeking clarity and calmness.

Finally, my deepest gratitude goes to my family – my parents, my brother, and my

husband – for being my foundation throughout this journey. They have been my biggest cheerleaders, standing by me through moments of self-doubt, and giving me the strength to keep going. My brother has been one of the calmest, most thoughtful human in my life. With endless patience, he has listened to multiple instances of my overcooked jargon, pulled me out of more self-dug mental pits than I can count, and quietly shielded me from anything that might distract me, just so I could stay focused. He has always encouraged me to look at things more positively and shared books for my growth (many of which I *do* plan to finish!). My husband, too, has been a steady and loving presence – kind, understanding, and always there. His quiet support and care have meant more to me than I can put into words.

For every suggestion that shaped my thoughts, every correction that deepened my understanding, and every gesture of faith that lifted me – thank you. Your impact is woven into this journey.

ABSTRACT

KEYWORDS functional encryption; multi-input functional encryption; attribute-based encryption; predicate encryption; indistinguishability obfuscation; laconic obfuscation; pseudorandom functionality; evasive LWE; broadcast, trace and revoke ; Turing Machines

As digital systems grow more interconnected, there is an increasing need for encryption mechanisms that do more than simply protect data – they must also control *who* can access *what* and *under what conditions*. Traditional public-key encryption follows an "all-or-nothing" paradigm, where possession of a secret key grants full access to the message. This model is inadequate for settings such as cloud computing or data-sharing platforms, where different users may require access to different parts of the data. For instance, a manager may require access to financial records but not legal documents within the same encrypted dataset. To address this, advanced encryption primitives such as *Attribute-Based Encryption* (ABE) and *Functional Encryption* (FE) enable fine-grained access control, allowing decryption only when specific conditions are satisfied (ABE), or enabling recovery of only the output of a function on encrypted data (FE).

While these advanced primitives offer appealing forms of access control, realizing them from well-understood and quantum-safe assumptions – like the Learning With Errors (LWE) – has resisted satisfying solutions despite significant effort. This thesis explores a new pathway using the evasive LWE assumption (Wee, Eurocrypt 2022 and Tsabary, Crypto 2022), a recent strengthening of LWE that has enabled progress on long-standing open problems in the area. Our results include the following advances:

1. We construct an optimal broadcast, trace, and revoke (TR) system that allows content to be securely distributed to many users, supports revocation of users from future communications, and enables tracing of unauthorized redistribution. A desirable feature of this notion is that of embedded identities – where user identities are embedded in

their secret keys, eliminating the need for a public index-identity mapping and enhancing anonymity. We study the notion in both secret tracing (where the tracing key is private) and public tracing (where the tracing key is public) with embedded identities. In the public tracing setting, we provide a construction relying on standard polynomial-hardness assumptions, namely FE and special forms of ABE. In the secret tracing setting, we achieve a TR scheme with asymptotically optimal parameters for ciphertexts, public keys, and secret keys using Lockable Obfuscation (LO), a key-policy ABE from Boneh et al. [43] based on LWE, and a ciphertext-policy ABE for P constructed by Wee [169] from evasive and tensor LWE.

2. We give the first ABE scheme for Turing machines supporting unbounded collusion from lattice assumptions. This allows policies expressed as general-purpose programs that run on unbounded-length inputs and support input-specific runtime – far beyond what circuit-based models can handle. Our construction uses LWE, evasive LWE, and a new assumption which we call the "circular tensor LWE" assumption. Towards our ABE for Turing machines, we obtain the first ciphertext-policy ABE for circuits of unbounded depth and size from the same assumptions.

3. We introduce and construct the first compact FE scheme for pseudorandom functionalities, under standard LWE and private-coin evasive LWE. Intuitively, a pseudorandom functionality means that the output of the circuit is indistinguishable from uniform for *every* input seen by the adversary. We demonstrate the power of our tool by achieving optimal key-policy and ciphertext-policy ABE for unbounded-depth circuits, improving upon prior work by: (i) replacing the circular evasive LWE used by Hsieh, Lin, and Luo [122] with plain evasive LWE, (ii) removing the need for circular tensor LWE from our prior work, and (iii) achieving asymptotically optimal parameters.

4. Finally, we compile our pseudorandom FE scheme into the first multi-input functional encryption (MIFE), where multiple users encrypt their inputs independently and a

functional key can compute a function jointly on them, and indistinguishability obfuscation ($i\mathcal{O}$), which hides everything about a program except its input-output behavior, for pseudorandom functionalities. Our MIFE scheme relies on LWE and (private-coin) evasive LWE for constant arity functions, and a strengthening of evasive LWE for polynomial arity. Thus, we obtain the first MIFE and $i\mathcal{O}$ schemes for a nontrivial functionality from conjectured post-quantum assumptions.

These results demonstrate that evasive LWE offers a powerful ground for building expressive, efficient, and (conjectured) quantum-secure cryptographic primitives. Finally, we discuss recent attacks on evasive LWE and their repercussions. Following the first online posting of our prFE work, a series of counterexamples to evasive LWE emerged [19, 121, 85], showing that the intuition behind evasive LWE – specifically, that it avoids the zeroizing regime – is not universally valid. However, we show that by taking suitable precautions, it is possible to recover the security of our constructions. We view these attacks as a constructive step toward a deeper understanding of evasive LWE. In our opinion, constructions from disciplined new assumptions are important to make meaningful progress on long-standing problems that have resisted solutions from standard assumptions despite much effort.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATION

$[m, n]$, for $m, n \in \mathbb{N}$	$\{i \ : \ m \leq i \leq n\}$
$[n]$, for $n \in \mathbb{N}$	$\{1, \ldots, n\}$
M (bold capital letter)	Matrix **M**
v (bold small letter)	Vector **v**
$\mathcal{D}_{\Lambda,\sigma}$	Discrete Gaussian distribution over lattice Λ with std. deviation σ and mean $\mathbf{0}$
$\log x$	Base 2 logarithm of x
$\mathsf{negl}(\lambda)$	A negligible function of λ
$\|\mathbf{v}\|$	ℓ_∞ norm of vector **v**
$\mathrm{poly}(\lambda)$	Polynomial in λ
PPT	Probabilistic Polynomial Time
$\mathbf{1}_{\ell \times n}$(resp. $\mathbf{0}_{\ell \times n}$)	A matrix of dimensions $\ell \times n$, having each entry as 1 (resp. 0)
$\mathbf{1}_\ell$(resp. $\mathbf{0}_\ell$)	Vector $(1, \ldots, 1) \in \mathbb{Z}^\ell$(resp. $(0, \ldots, 0) \in \mathbb{Z}^\ell$)
$\mathbf{x}\|\mathbf{y}$ (resp. $\mathbf{X}\|\mathbf{Y}$)	Horizontal concatenation of vectors **x** and **y** (resp. matrices **X** and **Y**)
$D_1 \approx_c D_2$	Distributions D_1 and D_2 are computationally indistinguishable
$D_1 \approx_s D_2$	Distributions D_1 and D_2 are statistically indistinguishable
$D_1 \equiv D_2$	Distributions D_1 and D_2 are perfectly indistinguishable

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Encryption is a fundamental tool for securing data in an increasingly interconnected world, allowing users to share sensitive information over insecure networks or storage systems. However, traditional public-key encryption enforces an "all-or-nothing" decryption model, where possession of a decryption key grants full access to the plaintext. As cloud computing, distributed systems, and data-driven applications continue to expand, concerns about privacy and controlled access to sensitive information have become more pressing. Standard encryption mechanisms fall short in scenarios requiring selective access – such as an enterprise system where employees should access records based on their roles, departments, or clearance levels, without revealing the rest of the database, or a content provider who wants to restrict users to only the portions of a dataset relevant to their subscription level. To overcome these limitations, advanced cryptographic primitives such as *Attribute-Based Encryption* (ABE) [157, 116] and *Functional Encryption* (FE) [157, 46] provide fine-grained access control, enabling decryption only under specified conditions (ABE), or permitting recovery of specific functions of the encrypted data (FE).

While these primitives offer powerful capabilities, constructing them for expressive class of functionalities from well-understood and quantum-safe assumptions has remained a significant challenge. Ideally, cryptographic security is most compelling when it rests on standard computational assumptions. The ideal paradigm ensures that breaking a cryptographic scheme is no easier than solving a well-studied hard problem – such as factoring, discrete logarithms, or learning with errors (LWE). This not only instills confidence in security but also creates a meaningful feedback loop: any successful attack

on the scheme advances our understanding of the underlying computational hardness. However, not all cryptographic constructions fit neatly into this framework. Several advanced primitives, such as functional encryption for general circuits, have remained out of reach when relying solely on well-established lattice assumptions, despite significant efforts by the community. This highlights the need for principled new assumptions that can push the boundaries of theoretical cryptography.

Evasive LWE. A promising development in this direction is the *evasive* LWE assumption, introduced independently by Wee [169] and Tsabary [163]. This assumption was designed to support the security of cryptographic primitives – such as witness encryption and broadcast encryption – that had long resisted reductions to standard lattice-based hardness assumptions. While techniques from lattice-based cryptography have yielded candidate constructions for these primitives, formal security proofs under standard assumptions remained elusive. Evasive LWE was proposed to bridge this gap, offering a principled strengthening of standard LWE that enables security proofs for a broader class of functionalities.

To recall, the standard LWE assumption posits that for a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, and an error vector $\mathbf{e}$ drawn from a discrete Gaussian distribution, the distribution $(\mathbf{A}, \mathbf{A}^\intercal \mathbf{s} + \mathbf{e})$ is computationally indistinguishable from $(\mathbf{A}, \mathbf{u})$, where $\mathbf{u}$ is uniform in $\mathbb{Z}_q^m$.

The evasive LWE assumption strengthens this by allowing the adversary additional auxiliary information beyond standard LWE samples. Roughly, it asserts that if the *precondition*

$$\left(\mathbf{B},\ \mathbf{P},\ \ \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}},\ \ \mathbf{s}^\intercal \mathbf{P} + \mathbf{e}_{\mathbf{P}},\ \ \mathsf{aux}\right) \approx_c \left(\mathbf{B},\ \mathbf{P},\ \ \$,\ \ \$,\ \ \mathsf{aux}\right)$$

holds (where \$ denotes uniform), then so does the corresponding *postcondition*

$$\left(\mathbf{B},\ \mathbf{P},\ \ \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}},\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux}\right) \approx_c \left(\mathbf{B},\ \mathbf{P},\ \ \$,\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux}\right).$$

Here, $\mathbf{B}^{-1}(\mathbf{P})$ refers to a low-norm matrix $\mathbf{K}$ satisfying $\mathbf{BK} = \mathbf{P} \bmod q$.

This models a setting where the adversary learns LWE samples (e.g., $\mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e_B}$) and a trapdoor preimage $\mathbf{K}$, enabling computation of $\mathbf{s}^\mathsf{T}\mathbf{P} + \mathbf{e_B}\mathbf{K}$. The core heuristic behind evasive LWE is that the term $\mathbf{s}^\mathsf{T}\mathbf{P} + \mathbf{e_B}\mathbf{K}$ is *flooded* by the high-entropy component $\mathbf{s}^\mathsf{T}\mathbf{P}$ and hence appears pseudorandom. The structure in $\mathbf{e_B}\mathbf{K}$ becomes irrelevant under this masking, rendering the trapdoor useless. This shifts the setting away from the "zeroizing" regime, where small-norm relations over the integers enable attacks.

The evasive LWE assumption is broadly categorized into two variants: the *public-coin* and *private-coin* formulations. In the public-coin variant, proposed by Wee [169], the sampler Samp outputs matrices $\mathbf{B}$, $\mathbf{P}$, and auxiliary information aux, where aux includes all the random coins used by Samp. This version suffices for constructing primitives such as optimal broadcast encryption [169] and multi-authority ABE [166]. Vaikuntanathan, Wee, and Wichs [164] introduced the *private-coin* variant, in which the sampler's randomness is hidden from the adversary. This stronger formulation has enabled a broader range of powerful constructions, including witness encryption and null-iO [163, 164], unbounded-depth KP-ABE [122] and CP-ABE [13], adaptively sound SNARGs [146, 131], and functional encryption and iO for pseudorandom functionalities [12, 14, 65].

The hope behind evasive LWE was to facilitate progress on long-standing, presumably "intermediate" cryptographic goals – such as constructing broadcast encryption from lattices. Its simplicity and generality made it a compelling foundation for achieving concrete advances in primitives that had resisted significant progress for over a decade. This thesis explores the broadening of the cryptographic landscape enabled by evasive LWE, demonstrating how this assumption can be leveraged to construct expressive, efficient, and provably secure primitives in the (conjectured) post-quantum regime.

1.2 OVERVIEW OF THE THESIS

In this section, we briefly outline the contributions presented in this thesis, following the chronological order in which the works were developed.

*Broadcast, Trace, and Revoke (*TR*) with Embedded Identities.* In a broadcast, trace and revoke system an encryptor can specify a list $L \subseteq N$ of revoked users so that (i) users in L can no longer decrypt ciphertexts, (ii) ciphertext size is independent of L, (iii) a pirate decryption box supports tracing of compromised users. The "holy grail" of this line of work is a construction which resists unbounded collusions, achieves all parameters (including public and secret key) sizes independent of $|L|$ and $|N|$, and is based on polynomial hardness assumptions. A desirable feature in this notion is the ability to embed user identities directly into secret keys [151]. This eliminates the need to store an index-to-identity mapping and improves user anonymity, particularly in the public trace setting. Our contribution in this domain is as follows.

1. *Public Trace Setting:* We provide a construction which achieves optimal parameters, supports embedding identities (from an exponential space), relies on polynomial hardness assumptions, namely compact functional encryption (FE) and a key-policy attribute based encryption (ABE) with special efficiency properties, and enjoys adaptive security with respect to the revocation list. The prior work [151] achieved optimal parameters and embedded identities, relied on the stronger tool of indistinguishability obfuscation and achieved only selective security with respect to the revocation list.

2. *Secret Trace Setting:* We provide the first construction with optimal ciphertext, public and secret key sizes and embedded identities from any assumption outside Obfustopia. Our construction relies on Lockable Obfuscation which can be constructed using LWE [111, 172] and two ABE schemes: the kpABE scheme by [43] which relies on LWE and the cpABE scheme by [169] which was constructed using the new assumptions called *(public-coin) evasive and tensor* LWE.

   Moreover, by relying on subexponential security of LWE, both our constructions can also support a *super-polynomial* sized revocation list, so long as it allows efficient representation and membership testing. Ours is the first work to achieve this, to the best of our knowledge.

Attribute-Based Encryption for Turing Machines. In an ABE scheme, a ciphertext encodes a message m and a public attribute $\mathbf{x}$, while a secret key embeds a function f;

decryption reveals m if and only if $f(\mathbf{x}) = 1$. Two common variants are key-policy (kpABE), where f is in the key, and ciphertext-policy (cpABE), where f is in the ciphertext. Typically, f is represented as a circuit.

While there has been significant progress in constructing ABE for general circuits [116, 107, 43, 26, 169, 122], the circuit model imposes rigid input sizes and worst-case evaluation time. To overcome these issues, several works [165, 102, 32, 15, 136, 103, 104, 17, 18, 142] have explored ABE for uniform computation models. Without relying on iO or compact FE, the best known constructions in this regime support non-deterministic log-space Turing machines from pairings [142]. In the post-quantum setting, the strongest result [17] supports non-deterministic finite automata under LWE, but only in the symmetric-key setting. We give the first ABE scheme for Turing machines with unbounded collusion resistance from lattice assumptions. The encryptor encodes an attribute $\mathbf{x}$, a time bound t, and a message m into the ciphertext. The secret key embeds a Turing machine M, and decryption returns m if and only if $M(\mathbf{x}) = 1$.

We achieve the first (conjectured) post-quantum (i) ABE for NL from LWE, (public-coin) evasive LWE, and tensor LWE; and (ii) an ABE for all Turing machines from LWE, (private-coin) evasive LWE, and a new assumption we call circular tensor LWE, which incorporates circularity into tensor LWE. As a stepping stone for ABE for TM, we also obtain the first cpABE for circuits of unbounded depth and size from the same assumptions.

FE and family for pseudorandom functionality. Functional encryption (FE) allows decryption of $f(\mathbf{x})$ from a ciphertext for input $\mathbf{x}$ and a secret key for function f, revealing nothing else. FE is a powerful theoretical tool and can be used to construct advanced primitives, most notably indistinguishability obfuscation ($i\mathcal{O}$), which is considered "crypto-complete" [34, 90, 129]. To obtain $i\mathcal{O}$ from FE, the scheme must satisfy *compactness*, meaning ciphertexts should be sublinear in the size of the

supported circuits. There has been substantial research effort in the community for instantiating FE (or directly iO) from well-understood assumptions, leading to a sequence of exciting results [128, 129, 5, 20, 171, 96, 82, 129, 130, 154]. The breakthrough work of Jain, Lin and Sahai [129] finally obtained the first construction of compact FE for P from standard assumptions. This has been subsequently improved by [130, 154]. However, these constructions rely heavily on pairings, which are quantum insecure and limit the diversity of underlying assumptions.

In the lattice world, several candidates for compact FE and iO exist [5, 20, 171, 96, 82, 120, 126], but they either rely on heuristics or the underlying assumptions have been broken. This raises a central open question: *Can we construct compact functional encryption for any nontrivial functionality from simple lattice assumptions?*

We introduce and construct the first compact FE scheme for pseudorandom functionalities, under standard LWE and private-coin evasive LWE. Intuitively, a pseudorandom functionality means that the output of the circuit is indistinguishable from uniform for *every* input seen by the adversary. We demonstrate the power of our tool by achieving optimal key-policy and ciphertext-policy ABE for unbounded-depth circuits, improving upon prior work by: (i) replacing the circular evasive LWE used by Hsieh, Lin, and Luo [122] with plain evasive LWE, (ii) removing the need for circular tensor LWE from our prior work [13], and (iii) achieving asymptotically optimal parameters. Finally, we compile our pseudorandom FE scheme into the first multi-input functional encryption (MIFE), where multiple users encrypt their inputs independently and a functional key can compute a function jointly on them, and indistinguishability obfuscation (iO), which hides everything about a program except its input-output behavior, for pseudorandom functionalities. Our MIFE scheme relies on LWE and (private-coin) evasive LWE for constant arity functions, and a strengthening of evasive LWE for polynomial arity. Thus, we obtain the first MIFE and iO schemes for a nontrivial functionality from conjectured post-quantum assumptions. Our tools of MIFE and iO for pseudorandom functionalities

appear quite powerful and yield extremely simple constructions when used in applications. We believe they provide a new pathway for basing "extreme" cryptography, which has so far required full fledged $i\mathcal{O}$, on the presumably weaker evasive LWE in the post quantum regime.

Attack on Evasive and Repercussions. Evasive LWE has been studied in two regimes: the *private-coin* setting, where random coins used by the sampler is hidden, and the *public-coin* setting, where it is revealed to the adversary. While early counterexamples in the private-coin case [164] relied on contrived auxiliary information, no known attacks existed against the public-coin variants, including circular evasive LWE [122]. This lent credibility to evasive LWE as a "middle ground" assumption between standard LWE and stronger assumptions needed for $i\mathcal{O}$.

Recent works [19, 121, 85] present new attacks showing the vulnerabilities of evasive LWE (even in the public-coin setting), particularly when maliciously crafted samplers or contrived functionalities are allowed. Specifically, [19, 121] demonstrate that a non-black-box use of pseudorandom functions can violate the post-condition while maintaining the pre-condition of the initial-version of the assumption used by our prFE scheme. These attacks also apply to the circular, small-secret variant of evasive LWE [122]. The works [65, 19] show impossibility results for pseudorandom functionalities under contrived, self-referential function classes. However, these are analogous to known limitations in other idealized models like ROM [68] and VBB [34], and do not apply to natural functionalities used in our constructions. The attacks by [66] target settings where either $\mathbf{B}$ or $\mathbf{P}$ is hidden. In our case, $\mathbf{B}$ is public, and $\mathbf{P}$ is computable from auxiliary information which is public.

Countermeasures As discussed in [19], several refinements to the evasive LWE assumption can be made to avoid known counterexamples arising from malicious samplers. One natural approach leverages the fact that, in cryptographic constructions, the key generation

algorithm is an honest party that holds the master secret key. Since the sampler is invoked by the key generator, it is reasonable, and often realistic, to assume that the function representation used in key generation can be enforced to follow a canonical structure. Furthermore, [19] outlines refinements to the evasive LWE assumption that exclude such malicious samplers. We follow their approach and restrict the class of samplers used in our constructions to avoid these vulnerabilities.

Our Perspective. We view these attacks as an important step in clarifying the scope and limitations of evasive LWE. Rather than discarding the assumption outright, we believe it should be refined to disallow malicious samplers or pathological functionalities. Carefully designed "safe zones" can preserve its utility – especially for functionalities that remain out of reach from standard assumptions. Finally, we remark that proposing principled new assumptions, by its very nature, highly non-trivial and it is unrealistic to expect the perfect formulation in the very first attempt. In our judgment, a balance of caution and risk is beneficial in this context.

1.3 ORGANIZATION OF THESIS

The rest of the thesis is organized as follows: we provide some preliminaries, used commonly in multiple chapters, in the thesis, in Chapter 2. In Chapter 3 we present our constructions of broadcast, trace, and revoke with embedded identities in both public and secret trace settings. In Chapter 4 we construct the first lattice-based ABE for Turing machines with unbounded collusions and also obtain the first ciphertext-policy ABE for unbounded-depth circuits. Chapter 5 introduces and constructs the notion of FE for pseudorandom functionalities, leading to optimal ABE and predicate encryption schemes. In Chapter 6 we define and construct the notion of multi-input FE and indistinguishability obfuscation ($i\mathcal{O}$) for pseudorandom functionalities, and demonstrates several applications. We also discuss the recent attacks on evasive and possible countermeasures in Chapter 7. We conclude the thesis in Chapter 8.

CHAPTER 2

PRELIMINARIES

In this chapter, we provide some preliminaries commonly used in this thesis. Additional preliminaries are given in respective chapters.

2.1 LATTICES AND DISCRETE GAUSSIANS

In this section, we recall some facts on lattices and discrete Gaussian distributions.

Definition 2.1 (Lattice)**.** An m-dimensional lattice Λ is a discrete additive subgroup of R^m. For an integer $n < m$ and a rank n matrix $\mathbf{B} \in \mathsf{R}^{m \times n}$, $\Lambda(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$ is the lattice generated by integer linear combinations of columns of matrix $\mathbf{B}$. The matrix $\mathbf{B}$ is called a *basis* of the lattice.

Definition 2.2 (Integral lattice)**.** An *m-dimensional integral lattice* Λ is a full-rank subgroup of $\mathbb{Z}^m$.

Among these lattices are the "q-ary" lattices defined as follows: for any integer $q \geq 2$ and any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define

$$\Lambda_q^{\perp}(\mathbf{A}) \quad \coloneqq \quad \left\{\mathbf{e} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{e} = \mathbf{0} \bmod q\right\}.$$

For a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define the following coset of $\Lambda_q^{\perp}(\mathbf{A})$:

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) \quad \coloneqq \quad \left\{\mathbf{e} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{e} = \mathbf{u} \bmod q\right\}.$$

We have $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^{\perp}(\mathbf{A}) + \mathbf{t}$ for any $\mathbf{t}$ such that $\mathbf{A} \cdot \mathbf{t} = \mathbf{u} \bmod q$.

Definition 2.3 (Gaussian distribution)**.** For any vector $\mathbf{c} \in \mathsf{R}^m$ and any real $s > 0$, the (spherical) Gaussian function with standard deviation parameter s and center $\mathbf{c}$ is defined

as:

$$\forall \mathbf{x} \in \mathsf{R}^m, \rho_{s,\mathbf{c}}(\mathbf{x}) = \mathsf{Exp}\left(-\frac{\pi\|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right).$$

The Gaussian distribution is $\mathcal{D}_{s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/s^m$.

The (spherical) *discrete Gaussian distribution* over a lattice $\Lambda \subseteq \mathsf{R}^m$, with standard deviation parameter $s > 0$ and center $\mathbf{c}$ is defined as:

$$\forall \mathbf{x} \in \Lambda, \mathcal{D}_{\Lambda,s,\mathbf{c}} = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)},$$

where $\rho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{s,\mathbf{c}}(\mathbf{x})$. When $\mathbf{c} = \mathbf{0}$, we omit the subscript $\mathbf{c}$.

2.1.1 Lattice Trapdoors

Let us consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. For all $\mathbf{V} \in \mathbb{Z}_q^{n\times m'}$, we let $\mathbf{A}^{-1}(\mathbf{V})$ be an output distribution of $\mathsf{SampZ}(\gamma)^{m\times m'}$ conditioned on $\mathbf{A}\cdot\mathbf{A}^{-1}(\mathbf{V},\gamma) = \mathbf{V}$; where $\mathsf{SampZ}(\gamma)$ is a sampling algorithm for the truncated discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\gamma > 0$ whose support is restricted to $z \in \mathbb{Z}$ such that $|z| \leq \sqrt{n}\gamma$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}^{-1}(\mathbf{V},\gamma)$ in time $\mathrm{poly}(n, m, m', \log q)$ for any $\mathbf{V}$. We slightly overload notation and denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$.

The following properties had been established in a long sequence of works [98, 71, 6, 7, 148, 57].

Lemma 2.1 (Properties of Trapdoors)**.** *Lattice trapdoors exhibit the following properties.*

1. *Given $\mathbf{A}_\tau^{-1}$, one can obtain $\mathbf{A}_{\tau'}^{-1}$ for any $\tau' \geq \tau$.*
2. *Given $\mathbf{A}_\tau^{-1}$, one can obtain $[\mathbf{A}\|\mathbf{B}]_\tau^{-1}$ and $[\mathbf{B}\|\mathbf{A}]_\tau^{-1}$ for any $\mathbf{B}$.*
3. *There exists an efficient procedure $\mathsf{TrapGen}(1^n, 1^m, q)$ that outputs $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ for some $m = O(n\log q)$ and is 2^{-n}-close to uniform, where $\tau_0 = \omega(\sqrt{n\log q \log m})$.*

Useful Lemmata.

Lemma 2.2 (tail and truncation of $\mathcal{D}_{\mathbb{Z},\gamma}$)**.** *There exists $B_0 \in \Theta(\sqrt{\lambda})$ such that*

$$\mathsf{Pr}[x \leftarrow \mathcal{D}_{\mathbb{Z},\gamma} : |x| > \gamma B_0(\lambda)] \leq 2^{-\lambda} \quad \textit{for all} \quad \gamma \geq 1 \textit{ and } \lambda \in \mathbb{N}.$$

Lemma 2.3 (Smudging Lemma [166])**.** *Let λ be a security parameter. Take any $a \in \mathbb{Z}$ where $|a| \leq B$. Suppose $\gamma \geq B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}\}$ is $\mathsf{negl}(\lambda)$.*

Lemma 2.4 (Leftover Hash Lemma)**.** *Fix some $n, m, q \in \mathbb{N}$. The leftover hash lemma states that if $m \geq 2n \log q$, then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{x} \leftarrow \{0,1\}^m$ and $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ the statistical distance between $(\mathbf{A}, \mathbf{A}\cdot\mathbf{x})$ and $(\mathbf{A}, \mathbf{y})$ is negligible. More concretely, it is bounded by $q^n\sqrt{2^{1-m}}$.*

2.1.2 Hardness Assumptions

Assumption 2.5 (The LWE Assumption)**.** Let $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda) > 2$ be integers and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$. We say that the LWE(n, m, q, χ) hardness assumption holds if for any PPT adversary $\mathcal{A}$ we have

$$|\mathsf{Pr}[\mathcal{A}(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal) \to 1] - \mathsf{Pr}[\mathcal{A}(\mathbf{A}, \mathbf{v}^\intercal) \to 1]| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the choice of the random coins by the adversary $\mathcal{A}$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^m$. We also say that LWE$(n, m, q, \chi)$ problem is subexponentially hard if the above probability is bounded by $2^{-n^\epsilon} \cdot \mathsf{negl}(\lambda)$ for some constant $0 < \epsilon < 1$ for all PPT $\mathcal{A}$.

As shown by previous works [155, 57], if we set $\chi = \mathsf{SampZ}(\gamma)$, the LWE$(n, m, q, \chi)$ problem is as hard as solving worst case lattice problems such as gapSVP and SIVP with approximation factor $\mathrm{poly}(n) \cdot (q/\gamma)$ for some $\mathrm{poly}(n)$. Since the best known algorithms for 2^k-approximation of gapSVP and SIVP run in time $2^{\tilde{O}(n/k)}$, it follows that the above LWE(n, m, q, χ) with noise-to-modulus ratio 2^{-n^ϵ} is likely to be (subexponentially) hard for some constant ϵ.

'p[];/-=]

Assumption 2.6 (Evasive LWE)**.** [169, 22, 19] Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be parameters for Gaussian distributions. For Samp that outputs

$$\mathbf{S} \in \mathbb{Z}_q^{m' \times n}, \mathbf{P} \in \mathbb{Z}_q^{n \times t}, \mathsf{aux} \in \{0,1\}^*$$

on input 1^λ and for PPT adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$, we define the following advantage functions:

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_0(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) = 1] - \mathsf{Pr}[\mathcal{A}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1] \tag{2.1}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \mathsf{Pr}[\mathcal{A}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1] \tag{2.2}$$

where

$$(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda),$$

$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m},$$

$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m' \times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m' \times t},$$

$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m' \times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m' \times t}$$

$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m \log(q)}).$$

We say that the *evasive* LWE (EvLWE) assumption with respect to the sampler class $\mathcal{SC}$ holds if for every PPT $\mathsf{Samp} \in \mathcal{SC}$ and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \quad \text{and} \quad \mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda). \tag{2.3}$$

We conjecture that for reasonable class of samplers, the evasive LWE assumption holds. In particular, we conjecture that our sampler $\mathrm{Samp}_{\mathsf{prFE}}(1^\lambda)$ used for the security proof of

our prFE for natural class of functions should be in the secure class of samplers $\mathcal{SC}$ for which the evasive LWE holds.

Remark 1. In the above definition, all the LWE error terms are chosen from the same distribution $D_{\mathbb{Z},\chi}$. However, in our security proof, we often consider the case where some of LWE error terms are chosen from $D_{\mathbb{Z},\chi}$ and others from $D_{\mathbb{Z},\chi'}$ with different $\chi \gg \chi'$. The evasive LWE assumption with such a mixed noise distribution is implied by the evasive LWE assumption with all LWE error terms being chosen from $D_{\mathbb{Z},\chi}$ as above definition, since if the precondition is satisfied for the latter case, that for the former case is also satisfied. To see this, it suffices to observe that we can convert the distribution from $D_{\mathbb{Z},\chi'}$ into that from $D_{\mathbb{Z},\chi}$ by adding extra Gaussian noise.

In multiple of our security proofs, we may require the auxiliary information to include terms dependent on $\mathbf{S}$. Furthermore, we may want to prove the pseudorandomness of such auxiliary information. The following lemma from [22] enables this. In the lemma, we separate the auxiliary information into two parts aux_1 and aux_2, where aux_1 is typically the part dependent on $\mathbf{S}$. The lemma roughly says that aux_1 is pseudorandom in the post condition distribution, if it is pseudorandom in the precondition distribution.

Lemma 2.7 (Lemma 3.4 in [22])**.** *Let $n, m, t, m', q \in \mathbb{N}$ be parameters and λ be a security parameter. Let χ and χ' be Gaussian parameters. Let* Samp *be a PPT algorithm that takes as input 1^λ and outputs*

$$\mathbf{S} \in \mathbb{Z}_q^{m' \times n}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \mathcal{S} \times \{0,1\}^* \text{ and } \mathbf{P} \in \mathbb{Z}_q^{n \times t}$$

for some set $\mathcal{S}$. Furthermore, we assume that there exists a public deterministic poly-time algorithm Reconstruct *that allows to derive $\mathbf{P}$ from aux_2, i.e. $\mathbf{P} = \mathsf{Reconstruct}(\mathsf{aux}_2)$.*

We introduce the following advantage functions:

$$\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda) \stackrel{def}{=} \Pr[\mathsf{Adv}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \Pr[\mathsf{Adv}(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2) = 1] \tag{2.4}$$

$$\mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}}(\lambda) \stackrel{def}{=} \mathsf{Pr}[\mathsf{Adv}(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2) = 1] - \mathsf{Pr}[\mathsf{Adv}(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2) = 1] \tag{2.5}$$

where

$$(\mathbf{S}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2), \mathbf{P}) \leftarrow \mathsf{Samp}(1^\lambda),$$

$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$$

$$\mathbf{C}_0 \leftarrow \mathbb{Z}_q^{m' \times m}, \mathbf{C}' \leftarrow \mathbb{Z}_q^{m' \times t}, \mathbf{c} \leftarrow \mathcal{S}$$

$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m' \times m}, \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m' \times t}$$

$$\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}) \text{ with standard deviation } O(\sqrt{m \log(q)}).$$

*Then, under the Evasive-*LWE *(cited above in Assumption 2.6) with respect to a sampler* $\mathsf{Samp} \in \mathcal{SC}$*, for a sampler class* $\mathcal{SC}$*, if* $\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda)$ *is negligible for any PPT adversary* Adv*, so is* $\mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}}(\lambda)$ *for any PPT adversary* Adv*.*

2.2 ATTRIBUTE BASED ENCRYPTION

We define both ciphertext policy attribute-based encryption (cpABE) and key policy attribute-based encryption (kpABE) in a unified form below.

Let $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a relation where $\mathcal{X}$ and $\mathcal{Y}$ denote "ciphertext attribute" and "key attribute" spaces, respectively. Ideally, we would like to have an ABE scheme that handles the relation R directly, where we can encrypt w.r.t any ciphertext attribute $x \in \mathcal{X}$ and can generate a secret key for any key attribute $y \in \mathcal{Y}$. However, in many cases, we are only able to construct a scheme that poses restrictions on the ciphertext attribute space and key attribute space. To capture such restrictions, we introduce a parameter prm and consider subsets of the domains $\mathcal{X}_{\mathsf{prm}} \subseteq \mathcal{X}$ and $\mathcal{Y}_{\mathsf{prm}} \subseteq \mathcal{Y}$ specified by it and the function R_{prm} defined by restricting the function R on $\mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}}$.

An attribute-based encryption (ABE) scheme for $R = \{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0, 1\}\}_{\mathsf{prm}}$

and a message space $\mathcal{M}$ is defined by the following algorithms.

- $\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk.

- $\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}_X$. The encryption algorithm takes as input a master public key mpk, a ciphertext attribute $X \in \mathcal{X}_{\mathsf{prm}}$, and a message $\mu \in \mathcal{M}$. It outputs a ciphertext ct_X.

- $\mathsf{KeyGen}(\mathsf{msk}, Y) \to \mathsf{sk}_Y$. The key generation algorithm takes as input the master secret key msk and a key attribute $Y \in \mathcal{Y}_{\mathsf{prm}}$. It outputs a private key sk_Y.

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}_X, X) \to \mu$ or $\perp$. The decryption algorithm takes as input the master public key mpk, a private key sk_Y, private key attribute $Y \in \mathcal{Y}_{\mathsf{prm}}$, a ciphertext ct_X and ciphertext attribute $X \in \mathcal{X}_{\mathsf{prm}}$. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

Definition 2.4 (Correctness)**.** An ABE scheme for relation family R is correct if for all prm, $X \in \mathcal{X}_{\mathsf{prm}}, Y \in \mathcal{Y}_{\mathsf{prm}}$ such that $R(X, Y) = 0$, and for all messages $\mu \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}), \\ \mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y), \\ \mathsf{ct}_X \leftarrow \mathsf{Enc}(\mathsf{mpk}, X, \mu) : \\ \mathsf{Dec}\Big(\mathsf{mpk}, \mathsf{sk}_Y, Y, \mathsf{ct}_X, X\Big) \neq \mu \end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 2.5 ($\mathsf{Sel\text{-}IND}$ security for ABE)**.** For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_{\mathsf{prm}} : \mathcal{X}_{\mathsf{prm}} \times \mathcal{Y}_{\mathsf{prm}} \to \{0, 1\}\}_{\mathsf{prm}}$ and a message space $\mathcal{M}$ and an adversary $\mathcal{A}$, let us define $\mathsf{Sel\text{-}IND}$ security game $\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ as follows.

1. $\mathcal{A}$ outputs prm and the challenge ciphertext attribute $X^\star \in \mathcal{X}_{\mathsf{prm}}$.

2. **Setup phase:** On input $1^\lambda, \mathsf{prm}$, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and gives mpk to $\mathcal{A}$.

3. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order. $\mathcal{A}$ can make unbounded many key queries, but can make only single challenge query.

a) **Key Queries:** $\mathcal{A}$ chooses an input $Y \in \mathcal{Y}_{\mathsf{prm}}$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, Y)$.

b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$ to the challenger. The challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct}_{X^\star} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

We require that $R(X^\star, Y) = 1$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.

4. **Output phase:** $\mathcal{A}$ outputs a guess bit b' as the output of the experiment.

We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) := \left|\mathsf{Pr}[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 0] - \mathsf{Pr}[\mathsf{Exp}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = 1 | b = 1]\right|.$$

The ABE scheme ABE is said to satisfy $\mathsf{Sel\text{-}IND}$ security (or simply *selective security*) if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

We can consider the following stronger version of the security where we require the ciphertext to be pseudorandom.

Definition 2.6 ($\mathsf{Sel\text{-}INDr}$ security for ABE)**.** We define $\mathsf{Sel\text{-}INDr}$ security game similarly to $\mathsf{Sel\text{-}IND}$ security game except that the adversary $\mathcal{A}$ chooses single message μ instead of (μ_0, μ_1) at the challenge phase and the challenger returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu)$ if $b = 0$ and a random ciphertext $\mathsf{ct} \leftarrow \mathcal{CT}$ from a ciphertext space $\mathcal{CT}$ if $b = 1$. Here, we assume that uniform sampling from the ciphertext space $\mathcal{CT}$ is possible without any parameter other than the security parameter λ. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies $\mathsf{Sel\text{-}INDr}$ security if the quantity is negligible.

We also consider the very selective notion of security.

Definition 2.7 ($\mathsf{VerSel\text{-}IND}$ security for ABE)**.** We define $\mathsf{VerSel\text{-}IND}$ security game similarly to $\mathsf{Sel\text{-}IND}$ security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute $X^\star$ in the beginning of the security game. We define the advantage $\mathsf{Adv}^{\mathsf{VerSel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies

VerSel-IND security if the quantity is negligible.

Definition 2.8 (VerSel-INDr security for ABE)**.** We define VerSel-IND security game similarly to Sel-INDr security game except that the adversary $\mathcal{A}$ outputs the key queries $Y_1, \ldots, Y_Q$, where Q is the number of key queries made by $\mathcal{A}$, along with the challenge ciphertext attribute $X^\star$ in the beginning of the security game. We define the advantage $\mathsf{Adv}^{\mathsf{VerSel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies VerSel-INDr security if the quantity is negligible.

Definition 2.9 (Ada-IND security for ABE)**.** We define Ada-IND security game similarly to Sel-IND security game except that the adversary $\mathcal{A}$ can choose the challenge ciphertext attribute $X^\star$ adaptively. We define the advantage $\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies Ada-IND security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation.

Ciphertext-policy Attribute Based encryption (cpABE)**.** We define cpABE for circuit class $\{C_{\ell(\lambda),d(\lambda)}\}_\lambda$ by specifying the relation. Here, $C_{\ell(\lambda),d(\lambda)}$ is a set of circuits with binary output whose input length is $\ell(\lambda)$ and the depth is at most $d(\lambda)$. Note that we do not pose any restriction on the size of the circuits. We define $A^{\mathsf{cpABE}}_\lambda = C_{\ell(\lambda),d(\lambda)}$ and $B^{\mathsf{cpABE}}_\lambda = \{0,1\}^\ell$. Furthermore, we define the relation $R^{\mathsf{cpABE}}_\lambda$ as

$$R^{\mathsf{cpABE}}_\lambda(C, \mathbf{x}) = C(\mathbf{x}).$$

Key-policy Attribute Based encryption (kpABE)**.** To define kpABE for circuits, we simply swap key and ciphertext attributes in cpABE for circuits. More formally, to define kpABE for circuits, we define $A^{\mathsf{kpABE}}_\lambda = \{0,1\}^\ell$ and $B^{\mathsf{kpABE}}_\lambda = C_{\ell(\lambda),d(\lambda)}$. We also define $R^{\mathsf{kpABE}}_\lambda : A^{\mathsf{kpABE}}_\lambda \times B^{\mathsf{kpABE}}_\lambda \to \{0,1\}$ as

$$R^{\mathsf{kpABE}}_\lambda(\mathbf{x}, C) = C(\mathbf{x}).$$

The above relations also holds for circuit class $\{C_{\ell(\lambda)}\}_\lambda$ which is the set of circuits with binary output whose input length is $\ell(\lambda)$ and the depth is unbounded.

2.3 PSEUDORANDOM FUNCTION

Definition 2.10 (Pseudorandom functions (PRF))**.** A pseudorandom function (PRF) family $\mathcal{F} = \{\mathsf{F}^K\}_{K \in \mathcal{K}}$ with a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$ is a function family that consists of functions $\mathsf{F}^K : \mathcal{X} \to \mathcal{Y}$. Let $\mathcal{R}$ be a set of functions consisting of all functions whose domain and range are $\mathcal{X}$ and $\mathcal{Y}$ respectively. A PRF family $\mathcal{F}$ is said to be secure if for any PPT adversary $\mathcal{A}$, the following condition holds,

$$|\mathsf{Pr}[\mathcal{A}^{\mathsf{F}^K(\cdot)}(1^\lambda) = 1] - \mathsf{Pr}[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $K \leftarrow \mathcal{K}$ and $R \leftarrow \mathcal{R}$.

CHAPTER 3

BROADCAST, TRACE AND REVOKE WITH OPTIMAL PARAMETERS FROM POLYNOMIAL HARDNESS

3.1 INTRODUCTION

Traitor Tracing. Traitor tracing (TT) schemes were first proposed by Chor, Fiat, and Naor [74] to enable content providers to trace malicious users who exploit their secret keys to construct illegal decryption boxes. More formally, a TT system is a public key encryption system comprising N users for some large polynomial N. Each user $i \in [N]$ is provided with a unique secret key sk_i for decryption, and there is a common public key pk which is used by the content distributor to encrypt content. If any collection of users attempts to create and sell a new decoding box that can be used to decrypt the content, then the tracing algorithm, given black-box access to any such pirate decoder, is guaranteed to output an index $i \in [N]$ of one of the corrupt users, which in turn allows to hold them accountable. The literature has considered both public and secret tracing, where the former requires knowledge of a secret key to run the trace procedure and the latter does not suffer from this restriction.

Broadcast Encryption. Broadcast Encryption [86] (BE) introduced by Fiat and Naor, is also an N user system which supports an encrypted broadcast functionality. In BE, a content provider can transmit a single ciphertext over a broadcast channel so that only an authorized subset $S \subseteq N$ of users can decrypt and recover the message. More formally, each user $i \in [N]$ is provided with a unique decryption key sk_i and a ciphertext ct_m for a message m also encodes an authorized list S so that sk_i decrypts ct_m if and only if $i \in S$. Evidently, public key encryption provides a trivial construction of BE with ciphertext of size $O(N)$ – thus, the focus in such schemes is to obtain short ciphertext,

ideally logarithmic in N.

Broadcast, Trace and Revoke. Naor and Pinkas [150] suggested a meaningful interleaving of these two functionalities so that traitors that are identified by the TT scheme can be removed from the set of authorized users in a BE scheme. To capture this, they defined the notion of "Broadcast, Trace and Revoke" (or simply "Trace and Revoke", which we denote by TR) where the content provider in a broadcast encryption scheme includes a list L of revoked users in the ciphertext, and sk_i works to decrypt ct_L if $i \notin L$. Moreover, it is required that revocation remain compatible with tracing, so that if an adversary builds a pirate decoder that can decrypt ciphertexts encrypted with respect to L, then the tracing algorithm should be able to output a corrupt non-revoked user who participated in building the illegal decoder. Trace and revoke systems provide a functionality which is richer than a union of BE and TT, since the traitor traced by the latter must belong to the set of non-revoked users for the guarantee to be meaningful. As such, TT schemes have been challenging to construct even given TT and BE schemes.

The Quest for Optimal Parameters. All the above primitives have been researched extensively over decades, resulting in a long sequence of beautiful constructions, non-exhaustively [44, 50, 48, 49, 151, 135, 112, 113, 27, 169]. A central theme in this line of work is to achieve optimal parameters, namely optimal sizes for the ciphertext, public key and secret key (and understanding tradeoffs thereof), while still supporting unbounded collusion resistance. Towards this, the powerful hammer of indistinguishability obfuscation ($i\mathcal{O}$) [34] yielded the first feasibility results for traitor tracing [49] as well as trace and revoke [151] while multilinear maps [89, 80] led to the first construction for broadcast encryption [48]. Though there has been remarkable progress in the construction of $i\mathcal{O}$ from standard assumptions, with the breakthrough work of Jain, Lin and Sahai [129, 130] finally reaching this goal, $i\mathcal{O}$ is an inherently subexponential assumption [93] because the challenger is required to check whether two circuits are functionally equivalent, which can take exponential time in general. Indeed,

all known constructions of $i\mathcal{O}$ assume subexponential hardness of the underlying algebraic assumptions. To address this limitation, a sequence of works [93, 15, 56, 92, 139] has sought to replace $i\mathcal{O}$ by polynomially hard assumptions such as functional encryption in different applications.

Optimal TT, BE and TR from Polynomial Assumptions: For traitor tracing, the first construction from standard assumptions was finally achieved by the seminal work of Goyal, Koppula and Waters [112] in the secret trace setting, from the Learning With Errors (LWE) assumption. For broadcast encryption, this goal was achieved by Agrawal and Yamada [27] from LWE and the bilinear GGM. In the standard model, Agrawal, Wichs and Yamada [24] provided a construction from a non-standard knowledge assumption on pairings, while Wee [169] provided a construction from a new assumption on lattices, called Evasive and Tensor LWE. For trace and revoke, the only construction without $i\mathcal{O}$ that achieves collusion resistance and optimal parameters is by Goyal, Vusirikala and Waters (GVW) [115] from *positional witness encryption* (PWE) which is a polynomial hardness assumption. However, their construction incurs an exponential loss in the security proof, requiring the underlying PWE to satisfy subexponential security. Moreover, although PWE is not an inherently subexponential assumption as are $i\mathcal{O}$ and witness encryption (WE), we do not currently know of any *constructions* of PWE that rely on standard polynomial hardness assumptions. In particular, [129, 130] do *not* imply PWE from polynomial hardness.

Pathway via Secret Tracing. Both the $i\mathcal{O}$ and PWE based constructions of TR [151, 115] achieve public tracing. Taking a lesson from TT, where optimal parameters were achieved from standard assumptions only in the secret trace setting [112], a natural approach towards optimal TR from better assumptions is to weaken the tracing algorithm to be secret key. This approach has been explored in a number of works – the current best parameters are achieved by Zhandry [174] who obtains the best known tradeoff in ciphertext, public key and secret key size. In particular, Zhandry [174] showed that all

parameters can be of size $O(N^{1/3})$ by relying on the bilinear generic group model (GGM). Note that the generic group model is a strong assumption, and indeed a construction secure in this model cannot be considered as relying on standard assumptions, since several non-standard assumptions on pairings are secure in the GGM. Prior to [174], Goyal et al. [114] provided a construction from LWE and Pairings, but their overall parameters are significantly worse – while their ciphertext can be arbitrarily small, $O(N^{\epsilon})$, their public key is $O(N)$ and secret key is $O(N^{c})$ for some large constant c[1].

Thus, a central open question in TR is:

Can we construct collusion resistant Trace and Revoke with optimal parameters from concrete polynomial assumptions?

Embedding Identities. Traditionally, it was assumed that tracing the index $i \in [N]$ of a corrupt user is enough, and there is an external mapping, maintained by the content distributor or some other party which associates the number i to the identity of the user, i.e. name, national identity number and such, which is then used to ensure accountability. The work of Nishimaki, Wichs and Zhandry (henceforth NWZ) [151] argued that this assumption is problematic since it implies that a user must trust the content provider with her confidential information. Storing such a map is particularly worrisome in the setting of public tracing since the user either cannot map the recovered index to an actual person, or the index-identity map must be stored publicly.

NWZ provided an appealing solution to the above conundrum – they suggest that identifying information be embedded in the key of the user, so that if a coalition of traitors constructs a pirate decoder, the tracing algorithm can directly retrieve the identifying

[1] Zhandry [174] states that the secret size in [114] is $O(N^2)$ but in fact the exponent is much larger due to the usage of arithmetic computations in NC^1, which blows up the circuit size associated with the ABE secret keys.

information from one of the keys that was used to construct the decoder and no one needs to keep any records associating users to indices. Notably, the identities can live in an *exponential* sized space, which introduces significant challenges in the tracing procedure. Indeed, handling an exponential space in the tracing procedure is the key contribution of NWZ. They also provided constructions of traitor tracing as well as trace and revoke with embedded identities, denoted by EITT and EITR respectively, from various assumptions.

3.1.1 Prior Work: Embedded Identity Trace and Revoke.

In the *public* trace setting, the only work that achieves embedded identity trace and revoke (EITR) with full collusion resistance is that of NWZ. However, while it takes an important first step, the construction by NWZ suffers from the following drawbacks:

1. *Reliance on Subexponential Hardness Assumption.* The construction relies on indistinguishability obfuscation [34], which appears to be an inherently subexponential assumption as discussed above.

2. *Selective Security in Revocation List:* Despite relying on adaptive security of functional encryption, the notion of security achieved by their construction is selective – the adversary must announce the revocation list before making any key requests or seeing the challenge ciphertext.

In the *secret* trace setting, the work of Kim and Wu [135] achieves EITR from the subexponential Learning With Errors (LWE) assumption. However, their construction incurs a ciphertext size that grows with the size of the revocation list. Additionally, while they can achieve adaptive security with respect to the revocation list, this is either by incurring an exponential loss in the security proof, or by assuming sub-exponential security for an ingredient scheme.

3.1.2 Our Results

In our work , we provide the first constructions with optimal parameters from polynomial assumptions, which additionally support embedded identities from an exponential space. We detail our contributions below.

Work	$\|\mathsf{CT}\|$	$\|\mathsf{SK}\|$	$\|\mathsf{PK}\|$	Trace Space	Sel/Adp	Asspn	Identities
[151]	1	1	1	Exp	Selective	Subexp (iO)	Yes
[115]	1	1	1	Poly	Adaptive	Subexp (subexp PWE)	No
This	1	1	1	Exp	Adaptive	Poly (FE and Special ABE)	Yes

Table 3.1: State of the art with Public Traceability.

Public Trace Setting. We provide a construction of Trace and Revoke with public tracing which overcomes the limitations of NWZ – (i) it relies on polynomial hardness assumptions, namely functional encryption and "special" attribute based encryption, both of which can be constructed using standard polynomial hardness assumptions [43, 129, 130] (ii) it enjoys adaptive security in the revocation list.

A detailed comparison with prior work is provided in Table 3.1.

Our Assumptions. Functional Encryption (FE) and Attribute Based Encryption (ABE) are generalizations of Public Key Encryption. In FE, a secret key corresponds to a circuit C and a ciphertext corresponds to an input x from the domain of C. Given a function key sk_C and a ciphertext ct_x, the decryptor can learn $C(x)$ and nothing else. It has been shown that FE implies $i\mathcal{O}$ [29, 40] albeit with exponential loss. The aforementioned work of Jain, Lin and Sahai [129, 130] provides a construction of compact FE from polynomial hardness assumptions, namely LPN, PRG in NC_0 and pairings. ABE is a special case of FE in which the input can be divided into a public and private part (x, m) and the circuit C in the secret key sk_C is only evaluated on the public part x in the ciphertext $\mathsf{ct}_{x,m}$. The private message m is revealed by decryption if and only if $C(x) = 1$. While FE implies ABE in general, we require our underlying ABE to satisfy special

efficiency properties, which is not generically implied by FE. However, the desired ABE can be instantiated using the construction of Boneh et al. [43] which is based on LWE.

Secret Trace Setting. In the secret trace setting, we achieve the optimal size of $O(\log N)$ for ciphertext, public and secret key by relying on Lockable Obfuscation (LO) [111, 172] and two special ABE schemes – one, the key-policy scheme with special efficiency properties by Boneh et al. [43] which is based on LWE, and two, a ciphertext-policy ABE for P which was recently constructed by Wee [169] using the new evasive and tensor LWE assumptions. Along the way, we show that a small modification to the TR construction by Goyal et al. [114] yields a ciphertext of size $O(\log N)$ as against their original $O(N^{\epsilon})$, from LWE and pairings. However, this construction retains the large public and secret keys of their construction, which depend at least linearly on N. Our results are summarized in Table 3.2.

Our Assumptions. We remark that while FE has now been constructed from standard assumptions [129, 130], the reliance of these constructions on pairings makes it insecure in the post-quantum regime. From lattices, constructions of FE rely on strong, non-standard assumptions which are often subject to attack [5, 21, 171, 96, 82, 126]. Hence, there is an active effort in the community [169, 163, 164] to construct advanced primitives from the hardness of weaker assumptions in the lattice regime. The new assumptions by Wee, also independently discovered by Tsabary [163], are formulated for designing ciphertext-policy ABE which is much weaker than FE since ABE is an all or nothing primitive in contrast to FE. As such, these are believed to be much weaker than lattice based assumptions that have been introduced in the context of FE or $i\mathcal{O}$. In particular, based on the current state of art, evasive LWE is required even for broadcast encryption in the lattice regime, and is therefore necessary for the generalization of broadcast encryption studied in this work.

Super-polynomial Revoke List. Lastly, by relying on subexponential security of LWE, both our constructions can support a *super-polynomial* sized revocation list, so long as it

Work	|CT|	|SK|	|PK|	Trace Space	Asspn	Identities
[114]	N^{ϵ}	N^{c}	N	Poly	LWE and Pairings	No
[174]	N^{a}	N^{1-a}	N^{1-a}	Poly	GGM Pairings	No
[135]	L	1	1	Exp	Subexp LWE	Yes
This work[11]	1	1	1	Exp	Evasive Tensor LWE	Yes
Modified [114]	1	N^{c}	N	Poly	LWE and Pairings	No

Table 3.2: State of the Art with Secret Traceability.

allows efficient representation and membership testing. Ours is the first work to achieve this, to the best of our knowledge.

3.1.3 Technical Overview

We proceed to give an overview of our techniques. We begin by defining the notion of revocable predicate encryption (RPE) in both the public and secret setting, then describe the ideas used to instantiate this primitive. Finally we outline how to upgrade public/secret RPE to build trace and revoke with embedded identities with public/secret tracing.

Revocable Predicate Encryption. NWZ introduced the notion of revocable functional encryption (RFE) and used it to construct EITR with public tracing. Subsequently, Kim and Wu [135] adapted this notion to the secret key setting, under the name of revocable predicate encryption (RPE) and used it to construct EITR with secret tracing. In this work, we extend Kim and Wu's notion of RPE to the public key setting and use it to construct EITR with public tracing. Our notion of RPE in the public setting is similar to but weaker than RFE[2] – it only supports "all or nothing" decryption in contrast to

[2]Syntactically, RPE is "ciphertext-policy" while RFE is "key-policy", i.e. the function is emdedded in the ciphertext in RPE as against the key in RFE.

RFE. This weaker notion nevertheless suffices to construct EITR and moreover admits constructions from weaker assumptions.

In RPE, the key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}$ and an attribute $x \in \mathcal{X}$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$. The encryption algorithm takes as input the encryption key ek, a function f, a message $m \in \mathcal{M}$, and a revocation list $L \subseteq \mathcal{L}$. It outputs a ciphertext ct. Decryption recovers m if $f(x) = 1$ and $\mathsf{lb} \notin L$. In the public variant of RPE, ek is a public key, while in the secret variant, ek is a secret key. In the secret variant, the scheme is also required to support a public "broadcast" functionality, i.e. there exists a public encryption algorithm that allows anyone to encrypt a message with respect to the "always-accept" policy, i.e. a policy that evaluates to true for all inputs. This is analogous to the primitive of "mixed FE" introduced by [112].

In terms of security, we require RPE to satisfy message hiding and function hiding. At a high level, message hiding stipulates that an adversary cannot distinguish between encryptions of (f, m_0) and (f, m_1) as long as every key query for (lb, x) satisfies $f(x) = 0$ or $\mathsf{lb} \in L$. Function hiding stipulates that an adversary cannot distinguish between encryptions of (f_0, m) and (f_1, m) as long as every key query for (lb, x) satisfies $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Before we describe our constructions, we highlight the chief difficulties that are inherent to designing RPE:

1. *Independence of parameter sizes from $|L|$.* A key requirement in TR schemes is that the ciphertext size should be independent of the length of the revocation list L – this constraint must also be satisfied by the underlying RPE, in both the secret and public setting. In our work, we insist that even the public and secret keys satisfy $|L|$ independence. This constraint is inherited from broadcast encryption, and is challenging to satisfy. Further, note that L must be unbounded – its length cannot be fixed during setup, which introduces additional difficulties.

2. *Encrypted Computation.* While the revocation list L need not be hidden by the ciphertext, the function f[3] in the ciphertext is required to be hidden, as formalized

[3]For the informed reader, this function encodes the "index" and function hiding corresponds to "index

by our function hiding requirement. Yet, this hidden function must participate in computing $f(x)$ where x is provided in the key. This requirement makes TR schemes worryingly close to collusion resistant functional encryption, an "obfustopia" primitive which we want to avoid in the secret trace setting.

Constructing Public Revocable Predicate Encryption. We proceed to describe the main ideas in constructing public RPE.

Overview of NWZ. The work of NWZ addresses the challenge of making the ciphertext size independent of $|L|$ by using a somewhere statistically binding (SSB) hash and hides the function f by using a functional encryption scheme, where f is encrypted in the ciphertext. However, they must additionally rely on iO – at a high level, this is because they require the *decryptor* to compute the SSB opening π and then run SSB verification on it (details of how SSB algorithms work are not relevant for this overview). In turn, the reason they need the decryptor to compute the opening π is because this needs both the set L and the label lb, which are available only to the decryptor – note that the encryptor has only L and the key generator has only lb. Now, since the decryptor has to compute π and run SSB verification, and since the program that computes SSB verification has some secrets, the decryptor is allowed to obtain obfuscation of this program. To implement this idea, they nest iO inside a compact FE scheme so that FE decryption outputs an iO which is then run by the decryptor on openings that it computes.

Trading iO for ABE. Above, note that the usage of iO is caused by the usage of SSB, which in turn is used to compress L. However, compression of a list has been achieved by much weaker primitives than iO in the literature of broadcast encryption – in particular, the construction of optimal broadcast encryption by Agrawal and Yamada uses the much weaker primitive of ABE (with special efficiency properties) to achieve this. However, ABE does not permit hiding anything other than a message, in particular, an ABE

hiding" in the literature.

ciphertext cannot encrypt our function f since we desire f to participate in computation. ABE only permits computation on public values, and using ABE to encode f would force f to be public which we cannot allow.

In order to get around this difficulty, we leverage the power of functional encryption (FE), which permits encrypted computation and exactly fills the gap over ABE that we require. A natural candidate for RPE would be to simply use FE to encrypt f, L and m, and encode x and lb in the secret key for a functionality which tests that $\mathsf{lb} \notin L$, that $f(x) = 1$ and outputs m if so. Indeed, this approach using FE is folklore, and was explicitly discussed by NWZ. Yet, they end up with a construction that additionally uses SSB, iO, a puncturable PRF and secret key encryption scheme because of the requirement of size independence from $|L|$ – we do not have candidates for FE with ciphertext size independent of the public attributes. In short, ABE gives us L compactness (in some cases by encoding L in the secret key [43] and in some cases by encoding L in the ciphertext [33]) but does not hide f, whereas FE gives the opposite.

Synthesis of ABE *and* FE. We address this conundrum by combining the two primitives in a way that lets us get the best of both. In particular, we use ABE to check that $\mathsf{lb} \notin L$ and use FE to compute $f(x)$. Evidently, the two steps cannot be performed independently in order to resist mix and match attacks so we use nesting, i.e. we use FE to generate ABE ciphertexts. Here, care is required, because ABE encryption takes L as input and done naively, this strategy will again induce a size dependence on L. We address this challenge by using the special ABE by Boneh et al. [43] which enjoys succinct secret keys and encoding L in the ABE secret key. In more detail, we let the RPE encryption generate $\mathsf{ABE.sk}(C_L)$ for a circuit C_L which takes as input lb and checks that $\mathsf{lb} \notin L$. Additionally, it generates an FE ciphertext for the function f and message m. The RPE key generator computes an FE key for a function which has (lb, x) hardwired and takes as input a function f, checks whether $f(x) = 1$ and if so, generates a fresh ABE ciphertext with attribute lb and message m. Thus the decryptor can first

compute FE decryption to recover the ABE ciphertext $\mathsf{ABE.ct}(\mathsf{lb}, m)$ and then use ABE decryption with $\mathsf{ABE.sk}(C_L)$ to output m if and only if $\mathsf{lb} \notin L$. It is easy to verify that this construction achieves optimal parameters – this is because ABE has optimal parameters and we used FE only for a simple functionality that does not involve L.

Putting it all Together. The above description is over-simplified and ignores technical challenges such as how to leverage indistinguishability based security of FE, how to generate the randomness used for ABE encryption and such others – we refer the reader to Section 3.5 for details. However, even having filled in these details, we get only a selectively secure RPE. Substantial work and several new ideas are required for adaptive security, as we discuss next.

Adaptive Security. Next, we outline our ideas to achieve adaptive security, namely where the revocation list L is chosen adaptively by the adversary. Note that to avoid complexity leveraging, we are required to rely only on the selective security of the underling ABE – this creates multiple technical difficulties which are resolved by very carefully using specific algebraic properties of our ingredients.

Leveraging Late Generation of ABE. Our first observation is that full adaptive security of ABE may be unnecessary, since in our construction of RPE, the generation of the ABE instance is deferred until the generation of the challenge ciphertext, at which time the set of revoked users is known. This intuition turns out to be true, but via a complicated security proof as we outline next. Below, we consider the case of function hiding in the RPE ciphertext, the case of message hiding is similar.

Recall that function hiding says that two ciphertexts encoding (f_b, m, L), where $b \in \{0, 1\}$ should be indistinguishable so long as for any requested key $\mathsf{sk}_{\mathsf{lb},x}$ it holds that $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. Note that the adversary is permitted to query for keys that allow decryption of the ciphertexts, i.e. $f_0(x) = f_1(x) = 1$.

Embedding ABE *CTs in* FE *keys.* In order to use ABE security to prove RPE security, a first (by now standard) step is to use the "trapdoor technique" [70, 28, 59], which allows us to hardwire ABE ciphertexts into FE secret keys. In the security game with the ABE challenger, the reduction submits the label lb associated with each RPE secret key as its challenge attribute and embeds the returned ABE ciphertext into the FE key. Here we immediately run into a difficulty, since in the RPE setting some ABE ciphertexts are decryptable by the adversary and we cannot leverage ABE security. Moreover, we cannot even hope to guess which keys will correspond to decryptable ABE ciphertexts since there are an unbounded polynomial number of key queries in the RPE security game. The same difficulty is faced by NWZ and is the main reason why their construction does not achieve adaptive security in the revocation list.

Polynomial Function Space Suffices for TR. To overcome this hurdle, we leverage the serendipitous fact that for the purpose of constructing TR, it suffices to construct RPE whose function space (recall that functions are encoded in the ciphertext) is only of polynomial size. This observation, which was implicitly present in [113], is abstracted and used explicitly in our proof. In particular, we can assume that the reduction algorithm knows the challenge functions (f_0, f_1) at the beginning of the game, since it can simply guess them. Now, given the secret key query (lb, x), the reduction checks whether $f_0(x) = f_1(x)$. If yes, then there is no need to use ABE security, for the ABE ciphertexts in this case will encode the same message, and will hence be independent of the challenge bit. On the other hand, if $f_0(x) \neq f_1(x)$, then we have by the admissibility condition that $\mathsf{lb} \in L$, even when L is not known. In this case, the reduction can use the security of the ABE without any difficulty.

Additional Hurdles Stemming from ABE *Selective Security.* We now highlight another challenge in the proof. For concreteness, let us consider the second key query $(\mathsf{lb}^{(2)}, x^{(2)})$, which we assume is a pre-challenge query, and assume that $f_0(x^{(2)}) \neq f_1(x^{(2)})$. Hence, by the above discussion, we are required to use ABE security for the ciphertexts with

attribute $\mathsf{lb}^{(2)}$. However, according to the selective definition, the reduction is required to choose the challenge attribute at the very start of the game, without even seeing the public parameters. At the same time, the reduction is required to simulate the ABE ciphertext for the first key query, before receiving the second key query from the adversary, that is, without seeing the ABE parameters, leading to an apparent impasse.

We address this issue by considering the following two cases separately: for the first query $(\mathsf{lb}^{(1)}, x^{(1)})$, we have (1) $f_0(x^{(1)}) \neq f_1(x^{(1)})$ or (2) $f_0(x^{(1)}) = f_1(x^{(1)})$. In first case, it is tempting to think that one can simply use a hybrid argument to change the ABE ciphertext associated with each key query satisfying $f_0(x^{(i)}) \neq f_1(x^{(i)})$ for $i \in [2]$. However, this does not work as is, since the ABE ciphertext may leak information about the ABE public key. To address this, we rely on the pseudorandomness of ciphertexts in our ABE [43] due to which we are guaranteed that the ciphertext does not reveal any information about the public parameters, enabling the hybrid strategy above. To handle the second case, we change the way in which the ABE ciphertext for the first key is generated. In more detail, we stop hardwiring the the ABE ciphertext into the first key and instead generate it directly using ABE parameters. This removes the aforementioned problem since we no longer need to embed the ABE ciphertext or public key into the first FE key. To enable this idea, we introduce additional branch of trapdoor mode for the construction to separate the paths of computation for the cases $f_0(x) = f_1(x)$ and $f_0(x) \neq f_1(x)$. To handle post-challenge queries, we need to address additional challenges, which we do not describe here. We refer the reader to Section 3.5 for details.

Handling Super-polynomial Revocation List. Our construction (also the secret version, described next) organically supports super-polynomially large revocation list, something that was not known before, to the best of our knowledge. In more detail, let L be a list of super-polynomial size, such that L can be represented as a string of polynomial length and there exists a circuit C_L of polynomial size which takes as input some string lb and checks whether $\mathsf{lb} \in L$ or not. Note that any super-polynomially large list must have

efficient representation in order to even allow various algorithms to read it. Then, the key generation of [43] can naturally encode the circuit C_L as before and the construction works as before. A subtlety that arises with super-polynomial L is that when we deal with post challenge key queries in the proof, we have to deal with the ABE queries in the order of key first and ciphertext later. With polynomial size L, this does not pose a problem because when the adversary chooses L, all the labels for which we use ABE security are in L and we can perform a hybrid argument over these labels. However, this is not possible for super polynomial L, which requires to rely on subexponentially secure LWE. Please see Section 3.12 for details.

Instantiating Public RPE. Overall, armed with the above ideas, we get a public RPE from compact FE and efficient ABE supporting exponential sized identity space and adaptive security in the revocation list L. Currently, we only know how to instantiate our desired ABE from LWE [43], whereas FE can be instantiated in multiple different ways. A natural candidate would be the FE from standard assumptions [129, 130] which relies on pairings, LPN and low depth PRG – in this case, our RPE will require the extra assumption of LWE. Another option is to instantiate FE with a post-quantum candidate [96, 144, 171, 5, 82] from non-standard strengthenings of LWE – this has the advantage that the ABE does not incur any extra assumption in the final construction. For super-polynomial L, we need subexponential hardness of LWE in either pathway to instantiation, as discussed above.

Alternative Construction Based on Laconic OT. Here, we sketch an alternative construction of RPE based on laconic OT (LOT) [73] that works when the number N of possible labels is polynomially bounded (i.e., the identity space is of polynomial size). Since LOT is known to be possible from various assumptions, this diversifies the assumptions that we need to rely on. The basic idea is to replace ABE with LOT. In more detail, the encryptor chooses LOT parameters instead of ABE parameters and computes the digest of the list of recipients (or equivalently, the list of revoked users),

which is represented as a binary string of length N with 1 for non-revoked identities. The digest, whose size is independent of N, is then embedded into the FE ciphertext. Then, FE decryption yields LOT encryption of the message for the label $\mathsf{lb} \in [N]$, which is the label associated with the secret key, instead of ABE ciphertext. The LOT ciphertext is encrypted so that it can be decrypted only when the lb-th bit of the binary string representing the list of recipients is 1. We note that this idea does not extend for identities from exponentially large space and cannot therefore support embedded identities any more.

Revocable Predicate Encryption in Private Setting. For private revocable predicate encryption, our starting point is the work of Goyal et al. [114], who show how to combine "broadcast mixed FE" (called BMFE) together with ABE to achieve RPE (via a different abstraction which they call AugBE). They construct BMFE by adding the broadcast functionality to the primitive of mixed FE defined by [112]. They embed BMFE ciphertext into an ABE ciphertext to achieve RPE, where BMFE is constructed from LWE and ABE is instantiated using pairings.

Supporting Exponential Identity Space. To begin, we upgrade their notion of BMFE to support an exponential space of identities (which we refer to as labels) towards the goal of embedded identity trace and revoke. We refer to our notion as Revocable Mixed FE (denoted by RMFE) and construct it from LWE. Both [114] and our work start with a mixed FE scheme and add broadcast to it, but their construction builds upon the scheme based on constrained PRFs [72] while ours begins with the scheme based on Lockable Obfuscation (LO), also from [72]. Our construction of RMFE deviates significantly from theirs, and achieves significantly better secret key size – $O(\log N)$ as against $O(N)$ – in addition to supporting exponential instead of polynomial space. We describe this construction next.

Mixed FE. The notion of mixed FE was introduced by Goyal, Koppula and Waters in the context of traitor tracing [112]. Identifying and constructing this clever primitive is the key insight that enables [112] to construct traitor tracing with optimal parameters from LWE. Mixed FE is, as the name suggests, a mix of public and secret key FE. Thus, it has a secret as well as a public encryption procedure. The secret encryption procedure takes as input a function f and computes ct_f. This is decryptable by a key sk_x to recover $f(x)$. The adversary can make one query to the encryption oracle in addition to getting the challenge ciphertext for challenge (f_0, f_1). It can also make an unbounded number of key requests so long as $f_0(x) = f_1(x)$. The public encryption algorithm computes a ciphertext for the "always accept" function, i.e. a function which evaluates to 1 for any input x. It is required that the public ciphertext be indistinguishable from the secret ciphertext.

One of the constructions of mixed FE suggested by [72] uses a secret key FE scheme (SKFE) to construct the secret encryption algorithm and leverages the power of lockable obfuscation (LO) to construct the public encryption procedure. Recall that in a lockable obfuscation scheme [111, 172] there exists an obfuscation algorithm Obf that takes as input a program C, a message m and a (random) "lock value" α and outputs an obfuscated program $\tilde{P}$. One can evaluate the obfuscated program on any input x to obtain as output m if $P(x) = \alpha$ and $\perp$ otherwise. Intuitively, the idea of [72] is to wrap the FE ciphertext using LO and to define the public key encryption algorithm as outputting a simulated version of the LO obfuscated circuit, which is publicly sampleable.

In more detail, the construction works as follows. The secret key for a user with input x is an SKFE secret key $\mathsf{SKFE.sk}(x)$. The secret ciphertext of MFE for function f is constructed as follows.

1. First, SKFE ciphertext $\mathsf{SKFE.ct}(H_{f,\alpha})$ is generated, where α is a freshly chosen random value and $H_{f,\alpha}$ is a circuit that takes as input x and outputs α if $f(x) = 0$ and 0 otherwise.

2. Then, LO with lock value α and any message $m \neq \perp$ is used to obfuscate the circuit SKFE.Dec(SKFE.ct($H_{f,\alpha}$), $\cdot$), namely the circuit that takes as input an SKFE secret key and decrypts the hardwired ciphertext using this.

The decryption result of MFE is defined as 1 if the evaluation result of the LO circuit on the given input SKFE secret key is $\perp$ and 0 otherwise. Correctness follows from correctness of SKFE and LO. In particular, if $f(x) = 0$, then SKFE decryption outputs α, which unlocks the LO to give m, otherwise $\perp$. By definition, MFE decryption will output 1 if LO outputs $\perp$ which happens when $f(x) = 1$, and 0 otherwise.

Revocable Mixed FE. RMFE augments MFE so that the encryption algorithms (both secret and public) now include a revocation list L and the secret key additionally includes a label lb. A secret key $\mathsf{sk}_{\mathsf{lb},x}$ decrypts a secret ciphertext $\mathsf{ct}_{f,L}$ to recover $f(x)$ if $\mathsf{lb} \notin L$ and 1 otherwise. For a public ciphertext ct_L, the output of decryption is always 1 regardless of which secret key is being used. For security, we need two properties: function hiding and mode hiding. For function hiding, we require that a secret ciphertext $\mathsf{ct}_{f_0,L}$ is indistinguishable from $\mathsf{ct}_{f_1,L}$ if for all queries, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. For mode hiding, we require that a secret ciphertext $\mathsf{ct}_{f,L}$ is indistinguishable from a public ciphertext ct_L. Recall that L is not required to be hidden, but we require that the parameters do not depend on $|L|$.

To extend MFE to RMFE, we retain the idea of letting the secret ciphertext be an LO obfuscated circuit and public ciphertext be the simulated LO. To incorporate the list L, we must ensure that the LO lock value α is recovered only when $f(x) = 0$ and $\mathsf{lb} \notin L$. To do so, we consider two subsystems such that one system outputs partial decryption result α_1 only when $f(x) = 0$ and the second system outputs partial decryption result α_2 only when $\mathsf{lb} \notin L$ such that $\alpha = \alpha_1 + \alpha_2$. We must ensure that α_1 and α_2 are user specific decryption results to avoid collusion attacks.

Note that the second subsystem, which entails L, should be constructed so that the hardwired values inside the circuit do not depend on $|L|$, but still control access to the

value α_2 depending on L. To satisfy these apparently conflicting requirements, we make use of the unique algebraic properties of the ABE construction by Boneh et al [43], as described below. For the first subsystem, we use SKFE.

In more detail, our candidate scheme is as follows.

1. **Secret Key:** The RMFE secret key consists of $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and $\mathsf{SKFE.ct}((x, K, R))$ where K and R are user specific random strings, lb is used as an attribute and K is the plaintext for ABE encryption.

2. **Ciphertext:** To generate RMFE ciphertext, the secret key encryption procedure is as follows:

   - It first generates $\mathsf{ABE.sk}(C_L)$, where C_L is a circuit that takes as input a label lb and outputs 1 only when $\mathsf{lb} \notin L$.

   - It also generates $\mathsf{SKFE.sk}(H_{f,\alpha})$, where $H_{f,\alpha}$ takes as input (x, K, R) and outputs $K \oplus \alpha$ if $f(x) = 0$ and R if $f(x) = 1$.

   - Now, consider the circuit $CC[\mathsf{ABE.sk}(C_L), \mathsf{SKFE.sk}(H_{f,\alpha})]$, which takes as an input the pair (ABE.ct, SKFE.ct), decrypts both ABE and SKFE ciphertexts using their respective keys, and then outputs the XOR between the decryption results.

   - The final ciphertext is an LO of $CC[\mathsf{ABE.sk}(C_L), \mathsf{SKFE.Enc}(H_{f,K,\alpha})]$ with lock value α and any arbitrary message $m \neq \perp$.

By key compactness of [43], the size of $\mathsf{ABE.sk}(C_L)$ is independent of $|L|$. A subtle point here is that ABE decryption is happening inside the LO and this depends on L. If the LO must process L, then the size of the LO and hence ciphertext blows up with L! Fortunately, the algebraic structure of the ABE scheme we use [43] again comes to our rescue. At a high level, ABE decryption can be divided into an "L-dependent" step which results in a short processed ciphertext, followed by an "L-independent" step. Importantly, the L-dependent step does not depend on the ABE secret key which is hardwired in the LO and hence inaccessible, and can hence be performed *outside* the LO by the decryptor! The resultant short processed ciphertext can then be provided as input to the LO preventing the problematic size blowup.

RMFE *Proof Overview.* Next we outline some of the ideas developed for the security proof. For ease of understanding, we limit ourselves to the simpler setting where the adversary does not have access to the encryption oracle. This restriction can be removed using combinatorial tricks, similar to [72]. For security, we must argue two properties – mode indistinguishability and function hiding. The former can be established by relying on security of SKFE and LO analogously to the MFE proof in [72]. Hence, we focus on function hiding for the rest of the overview, which is subtle and requires several new ideas.

For function hiding, we must make use of the security of ABE and SKFE. Intuitively, security of SKFE guarantees that the values encoded in SKFE ciphertexts and secret keys are hidden, beyond what is revealed by decryption.[4] First note that given a key for (lb, x) such that $f_0(x) = f_1(x)$, no information about the challenge bit is revealed by decryption, since the decryption results of SKFE are the same for both cases. The case with $f_0(x) \neq f_1(x)$ is more challenging. Let us assume $f_0(x) = 0$ and $f_1(x) = 1$. In this case, the decryption result of the challenge ciphertext is R or $K \oplus \alpha$ depending on the value of the challenge bit. Since both are random strings, it is tempting to conclude that they do not reveal any information of the challenge bit.

However, in reality, information about K is encoded in the ciphertext $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and creates a correlation which must be handled. Indeed, a computationally unbounded attacker can learn the challenge bit by breaking open the ABE ciphertext, recovering K and then correlating it with the decryption result of SKFE. Hence, security of ABE must play a role and fortunately, we show that security of ABE suffices to overcome this difficulty. Recall that our security definition of RMFE requires that if $f_0(x) \neq f_1(x)$, then it should hold that $\mathsf{lb} \in L$. This means that the ciphertext $\mathsf{ABE.ct}(\mathsf{lb}, K)$ is computationally indistinguishable from $\mathsf{ABE.ct}(\mathsf{lb}, 0)$, since the only ABE secret key available to the adversary is $\mathsf{ABE.sk}(C_L)$. Now, in the adversary's view, both $K \oplus \alpha$

[4] We note that we need message *and* function hiding security for the underlying SKFE, while [72] only needs message hiding security.

and R are random strings that are independent from other parameters. Therefore, the adversary cannot obtain any information of the challenge bit from the decryption result in this case as well. For more details, please see Section 3.6.

Comparison with the BMFE *by Goyal et al. [114].* We observe that both our RMFE as well as the BMFE by [114] rely solely on LWE. However, our secret key is $\mathsf{ABE.ct}(\mathsf{lb}, K)$ and $\mathsf{SKFE.ct}((x, K, R))$, which has optimal size, being clearly independent of N and L. In contrast their secret key depends linearly on N. We also observe that our RMFE can support an exponentially large space of identities, while their BMFE does not.

Combining RMFE *and* ABE *to get* RPE. Finally, we nest our RMFE inside an outer ABE scheme to obtain RPE. This step is very similar to [114], but we need to use a different ABE scheme. In particular, in the construction of RPE in [114], a key policy ABE (kpABE) is used to encrypt the message m with attributes as the RMFE ciphertext along with the list L. The RPE secret key for (x, lb) is a kpABE secret key for a the RMFE decryption circuit $\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}, \cdot, \cdot)$.

An obvious difficulty here is that encoding the attribute $(L, \mathsf{RMFE.ct})$ in the ABE ciphertext can cause the ciphertext size to depend on the size of L. To avoid this blowup, [114] use a special kpABE which has the property that the ciphertext size is independent of the size of the attribute. They instantiate this kpABE with the scheme [33] which uses pairings[5]. However, we cannot use [33, 160] because of the following two reasons:

1. First, the ABE scheme by [33] only supports NC_1. However, our circuit $\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}, \cdot, \cdot)$ does not fit into NC_1[6].

2. Furthermore, even if the above problem could be resolved, using [33] is problematic since their ABE has secret and public keys at least as large as $O(|L|)$. While the scheme of [114] also suffers from this blow-up, our goal is to obtain short keys, independent of $|L|$.

[5] In fact, one could instead use the kpABE constructed by [160]. This enjoys the same efficiency properties and is based on the standard DLIN assumption as against the q-type assumption of [33].

[6] The informed reader may wonder whether we can solve this issue by using preprocessing as in [114] but this does not work due to technical reasons.

The first problem cannot be resolved even if we use the ABE schemes for circuits [107, 43], since their ciphertext size also depends on $|L|$. To instantiate our ABE, we use recent construction of compact cpABE from evasive and tensor LWE [169], whose parameter sizes depend only on the input length of the circuit and are independent of its size. Armed with the above ideas, we suggest the following RPE:

1. The encryption algorithm of RPE, given m, f, L computes RMFE ciphertext encoding (f, L) and then computes $\mathsf{cpABE.Enc}(\mathsf{RMFE.Dec}(\mathsf{RMFE.ct}, \cdot, L), m)$.

2. The key generation algorithm RPE given (lb, x), computes RMFE secret key for (lb, x) and outputs $\mathsf{cpABE.sk}(\mathsf{RMFE.sk})$.

Correctness of RPE follows from correctness of cpABE and RMFE while optimality of parameters follows from the efficiency of the underlying schemes. In particular, observe that all parameters are independent of $|L|$. Also note that evasive and tensor LWE are required only to instantiate cpABE with the desired efficiency. If future work standardizes the assumptions underlying the cpABE, our construction will inherit these assumptions. For more details, we refer the reader to Section 3.7.

Instantiating Secret RPE. Currently, the only two suitable ABE schemes that we know to instantiate our compiler are the LWE based kpABE by Boneh et al. [43] and the evasive and tensor LWE based cpABE by Wee [169]. These two ABEs give us a secret RPE scheme supporting exponential identities and with optimal parameters, from evasive and tensor LWE. Note that this construction does *not* achieve adaptive security in the revoke list. Nevertheless, it is the first construction of optimal RPE, even without embedded identities, from any assumption outside Obfustopia. Note that the usage of a non-standard assumption outside of obfustopia (in particular, only from lattice techniques) is somewhat inherent given that even broadcast encryption *without* tracing requires non-standard assumption if we instantiate it only from lattices. We are hopeful that future improvements in cpABE will yield a construction from completely standard assumptions.

Trace and Revoke with Optimal Ciphertext from LWE **and Pairings.** Along the way, we observe that the broadcast and trace construction provided by Goyal et al. [114], without embedded identities, can be easily modified to achieve at least optimal ciphertext size, from the same assumptions. At a high level, they construct a broadcast mixed FE from LWE with optimal ciphertext size and then nest this inside the kpABE by [33], which enjoys ciphertext size independent of the attribute length, and can support computation in NC_1. Since their BMFE decryption does not fit into NC_1, they preprocess the ciphertext so that part of the decryption is performed "outside", namely, they group $\log N$ matrix tuples into c groups of $(\log N)/c$ tuples each. Then they precompute all possible $2^{(\log N)/c} = N^{1/c}$ subset-products within each group. Due to this, BMFE decryption only needs to multiply together c of the preprocessed matrices, which can be done in NC_1 so long as c is constant. Unfortunately, this step increases their ciphertext size to $O(N^\epsilon)$ for any $\epsilon > 0$ though the BMFE ciphertext size was optimal.

We observe that they are "under-using" the ciphertext size independence of [33] – in particular, while the attribute length has indeed been blown up to $O(N^\epsilon)$, this does not affect the ciphertext size of [33]. Moreover, while the attribute must also be provided outside in the clear, this part can be compressed, i.e. the preprocessing which expands the attribute to size N^ϵ can be performed by the decryptor directly by grouping and multiplying matrices as described above, and there is no need for the encryptor to provide this expanded form. Thus, their scheme tweaked with this simple modification already achieves ciphertext of optimal size, though with large secret key $O(N^c)$ for some large constant c.

Trace and Revoke from Revocable Predicate Encryption. It remains to show how to construct the final goal of trace and revoke with embedded identities. As discussed earlier, we follow [151, 135] and use the abstraction of RPE to build trace and revoke. However, to embed identities in our trace and revoke schemes, we deviate from these

works and instead build upon ideas developed by [113] (henceforth GKW) in the context of traitor tracing.

*Embedded Identity Traitor Tracing (*EITT*) by* GKW. The work of Goyal, Koppula and Waters [113] provided an alternative approach for embedding identities in traitor tracing schemes. A well known approach for constructing Traitor Tracing systems suggested by Boneh, Sahai and Waters [38] is via the intermediate primitive of *Private Linear Broadcast Encryption* (PLBE), which allows to construct a tracing algorithm that performs a linear search over the space of users to recover the traitor. Since the number of users was polynomial, this algorithm could be efficient. However, if we allow arbitrary identities to correspond to user indices then the space over which this search must be performed becomes exponential even if the number of users is polynomial, and the trace algorithm is no longer efficient. The main new idea in NWZ that enables them to handle exponentially large identity spaces is to replace a linear search over indices by a clever generalization of binary search, which efficiently solves an "oracle jump problem" which in turn suffices for tracing.

Goyal, Koppula and Waters (GKW) provided an alternate route to the problem of embedding identities. Instead of using PLBE and generalizing the search procedure, they instead extend the definition of PLBE to support embedded identities, denoted by EI-PLBE, and then used this to get a full fledged EITT scheme. This approach has the notable advantage that even if the space of identities is exponential, it can use the fact that the number of users is only *polynomial* and hence rely on only *selective* security of the underlying primitives. In particular, they demonstrate a "nested" tracing approach, where the tracing algorithm works in two steps: first, it outputs a set of indices that correspond to the users that are traitors, and then it uses each index within this set to recover the corresponding identity. Additionally, GKW provide a sequence of (increasingly stronger) TT primitives with embedded identities, namely, indexed EITT, bounded EITT and finally unbounded EITT where unbounded EITT satisfies the most general notion of

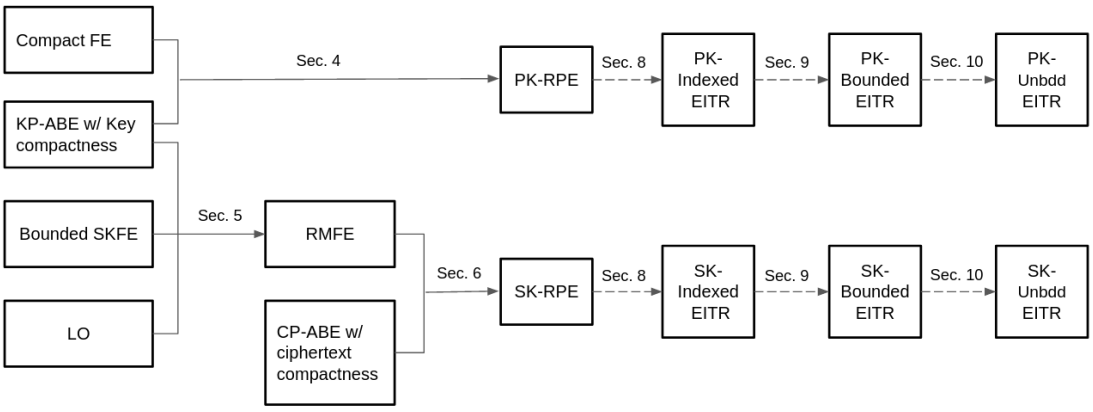

Figure 3.1: Overview of our constructions.

embedded identity traitor tracing. They also provide generic transformations between these notions, which allows to focus on the weakest notion for any new instantiation.

*Embedded Identity Trace and Revoke (*EITR*).* We adapt the approach of GKW and show how to use their nested approach to trace embedded identities even in the more challenging setting of trace and revoke. As in their case, this lets us use polynomial hardness assumptions in obtaining EITR, in contrast to NWZ. We also define indexed, bounded and unbounded EITR and provide transformations between them. Our definitions as well as transformations are analogous to GKW albeit care is required to incorporate the revoke list L in each step and adapt the definitions and proofs of security accordingly. We then construct indexed EITR using RPE, and obtain unbounded EITR via our generic conversions.

We note that our framework unifies the approaches of Kim and Wu [135] who used the framework of RPE in the context of TR and that of GKW who used the framework of EI-PLBE in the context of TT, to obtain EITR. This unification yields a clean abstraction which can be used for both public and secret key settings. We believe this framework is of independent interest. We refer the reader to Sections 3.8, 3.9, 3.10 and 3.11 for details. An overview of our constructions is provided in Figure 3.1.

Organization of the Chapter. We provide some additional preliminaries in Section 3.2. We define RPE in Section 3.4 and provide a construction of public-key RPE in Section 3.5. We give our construction of RMFE in Section 3.6. Then we construct secret-key RPE using RMFE in Section 3.7. We define different versions of trace and revoke with embedded identities in Section 3.8 and construct indexed-EITR in Section 3.9, bounded-EITR in Section 3.10 and unbounded-EITR in Section 3.11. Our goal is unbounded-EITR and other variants are introduced as intermediate goals. Secret and public tracing unbounded-EITR can be obtained by applying the conversions in Section 3.9, 3.10, and 3.11 to the secret-key and public key RPE, respectively. We show how to extend our constructions for super-polynomial sized revocation list in Section 3.12.

3.2 PRELIMINARIES

3.2.1 Symmetric Key Encryption

Definition 3.1 (Syntax)**.** A symmetric key encryption scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in[\mathbb{N}]}$ and key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in[\mathbb{N}]}$ and ciphertext space $\mathcal{CT}_{\mathsf{SKE}}$ has the following syntax:

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ and outputs a secret key sk.

$\mathsf{Enc}(\mathsf{sk}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m'$. The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs a message $m' \in \mathcal{M}_\lambda$.

Correctness: A SKE scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\mathsf{Pr}\left[\begin{array}{ll} & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ m' = m : & \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m); \\ & m' = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}). \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

Security: A SKE scheme is said to have pseudorandom ciphertext if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\mathsf{Pr}\left[\begin{array}{ll} & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ \beta' = \beta : & \\ & \beta' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk},\cdot),\mathsf{Enc}^\beta(\mathsf{sk},\cdot)}. \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the $\mathsf{Enc}(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns $\mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{Enc}(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns ct_β, where $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

3.2.2 Functional Encryption

Here, we recall the definition of public-key and secret-key functional encryption.

Public-Key Functional Encryption

Consider a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, with input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and output space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, i.e, $\mathcal{F}_\lambda = \{f : \mathcal{M}_\lambda \to \mathcal{Y}_\lambda\}$. A public-key functional encryption scheme FE for $\mathcal{F}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and outputs a public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_\lambda$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{m}) \to \mathsf{ct}$. The encryption algorithm takes as input the public key mpk and a message $\mathbf{m} \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y$. The decryption algorithm takes as input a functional secret key sk_f and a ciphertext ct and outputs $y \in \mathcal{Y}_\lambda$.

Definition 3.2 (Correctness)**.** A functional encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $\mathbf{m} \in \mathcal{M}_\lambda$, and for every function $f \in \mathcal{F}_\lambda$, we have

$$\mathsf{Pr}\left[f(\mathbf{m}) \leftarrow \mathsf{Dec}\big(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, \mathbf{m})\big) \right] \geq 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the probability is taken over the random coins of all algorithms.

Definition 3.3 (Selective Message Privacy)**.** A functional encryption scheme over a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ is said to have selective message privacy (or simply is selectively secure) if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have

$$\mathsf{Pr}\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{ct}) = b : \begin{array}{l} (\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}; \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ b \leftarrow \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{m}_b) \end{array} \right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where each key query for a function $f \in \mathcal{F}_\lambda$, queried by $\mathcal{A}$ to the KeyGen oracle must satisfy the condition that $f(\mathbf{m}_0) = f(\mathbf{m}_1)$.

Compactness : We say that a FE scheme is compact if the running time of the encryption algorithm only depends on the security parameter and the input message length. In particular, it is independent of the complexity of the function class supported by the scheme.

Definition 3.4 (Fully Compact FE)**.** A functional encryption scheme, $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and

function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $\mathcal{F}_\lambda$ is a finite collection of functions, is said to be fully compact if the running time of the encryption algorithm $\mathsf{FE.Enc}$, on input $\mathsf{FE.mpk}$ and a message $\mathbf{m} \in \mathcal{M}_\lambda$, is $\mathsf{poly}(\lambda, |\mathbf{m}|)$.

Jain, Lin and Sahai [129, 130] provided the first construction of FE with sublinear compactness, from standard assumptions. In more detail:

Lemma 3.1 ([130, 94, 30])**.** *There exists a public-key functional encryption scheme for polynomial sized circuits having selective security (as per Definition 3.3) and full compactness (as per Definition 3.4) with encryption time* $\mathrm{poly}(\lambda, |\mathbf{m}|)$*, where* $\lambda \in \mathbb{N}$ *is the security parameter,* $\mathbf{m}$ *is the input message, assuming LPN, DLIN and existence of boolean PRGs in* NC^0.

Secret Key Functional Encryption

A secret key functional encryption scheme is the same as the public key functional encryption scheme, except that the setup algorithm only outputs msk and the encryption algorithm takes the master secret key msk as input, instead of the master public key mpk.

Definition 3.5 ($\mathsf{Ada\text{-}IND}$ Function-Message Privacy ([59], adapted))**.** A $\mathsf{SKFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme over an input space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is said to be $\mathsf{Ada\text{-}IND}$ function and message private if for any PPT algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$|\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}(\lambda, 0) = 1] - \mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda)$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$, the $\mathsf{Exp}_{\mathcal{A}}(\lambda, b)$ is defined as follows

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}_b(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)}$, where the oracle $\mathsf{KeyGen}_b(\mathsf{msk}, \cdot, \cdot)$ on input (f_0, f_1) outputs $\mathsf{KeyGen}(\mathsf{msk}, f_b)$ and $\mathsf{Enc}_b(\mathsf{msk}, \cdot, \cdot)$ on input $(\mathbf{m}_0, \mathbf{m}_1)$ outputs $\mathsf{Enc}(\mathsf{msk}, \mathbf{m}_b)$.
3. Output b'.

and $\mathcal{A}$ is admissible only if for all the key queries $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ and encryption queries $(\mathbf{m}_0, \mathbf{m}_1) \in \mathcal{M} \times \mathcal{M}$, it must hold that $f_0(\mathbf{m}_0) = f_1(\mathbf{m}_1)$.

Note: We refer to the t-bounded Ada-IND function and message private SKFE scheme where Def. 3.5 holds only if at most t queries are made to the $\mathsf{KeyGen}_b(\mathsf{msk}, \cdot, \cdot)$ oracle.

Remark 2. We can construct a SKFE scheme satisfying t-bounded Ada-IND message privacy (without function privacy, i.e., $f_0 = f_1$ should hold in the security game) from one-way functions [31]. This implies a SKFE scheme satisfying t-bounded Ada-IND function-message privacy, using the conversion results from [59].

We also need a 2-bounded semi-adaptive simulation based function-message private SKFE scheme, defined next.

Definition 3.6 (Simulation Based Function-Message Privacy)**.** A secret-key functional encryption scheme is said to satisfy t-bounded semi-adaptive simulation based function and message privacy, if for all PPT stateful algorithm $\mathcal{A}$, there exists PPT stateful algorithms $\mathsf{Sim}^{\mathsf{SK}}, \mathsf{Sim}^{\mathsf{CT}}$ such that:

$$\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}^{\mathsf{SK}},\mathsf{Sim}^{\mathsf{CT}}}(1^\lambda)\}_{\lambda\in\mathbb{N}}$$

where the real and ideal experiments of stateful algorithms $\mathcal{A}, \mathsf{Sim}^{\mathsf{SK}}, \mathsf{Sim}^{\mathsf{CT}}$ are as follows:

$\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{A},\mathsf{Sim}}(1^\lambda)$
$\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$
For $i \in [t]$:	For $i \in [t]$:
$\quad \mathcal{A} \rightarrow f_i \in \mathcal{F}$;	$\quad \mathcal{A} \rightarrow f_i \in \mathcal{F}$;
$\quad \mathcal{A} \leftarrow \mathsf{sk}_{f_i} = \mathsf{KeyGen}(\mathsf{msk}, f_i)$	$\quad \mathcal{A} \leftarrow \mathsf{sk}_{f_i} = \mathsf{Sim}^{\mathsf{SK}}(\mathsf{msk}, 1^{\lvert f_i \rvert})$
Repeat polynomially many times:	Repeat polynomially many times:
$\quad \mathcal{A} \rightarrow \mathbf{m} \in \mathcal{M}$;	$\quad \mathcal{A} \rightarrow \mathbf{m} \in \mathcal{M}$;
$\quad \mathcal{A} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{m})$	$\quad \mathcal{A} \leftarrow \mathsf{Sim}^{\mathsf{CT}}(\mathsf{msk}, \{f_i(\mathbf{m})\}_{i\in[t]})$
$\mathcal{A} \rightarrow b$; Output b	$\mathcal{A} \rightarrow b$; Output b

Lemma 3.2. *There exists a 2-bounded semi-adaptive simulation based function and message private* SKFE *scheme, assuming the existence of one-way functions.*

Proof. Let $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ be a 2-bounded Ada-IND function-message private SKFE scheme. We construct a 2-bounded semi-adaptive simulation based function-message private $\mathsf{SKFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme as follows:

1. $\mathsf{Setup}(1^\lambda)$: Sample $\mathsf{msk} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$.
2. $\mathsf{KeyGen}(\mathsf{msk}, f)$: $\mathsf{sk} \leftarrow \Pi.\mathsf{KeyGen}(\mathsf{msk}, E[f, 0])$. Here, $E[f, \mathsf{mode}]$ is defined as $E[f, \mathsf{mode}](\mathbf{m}, y_1, y_2) = y_{\mathsf{mode}}$ for $\mathsf{mode} \in \{0, 1, 2\}$, where $y_0 := f(\mathbf{m})$.
3. $\mathsf{Enc}(\mathsf{msk}, \mathbf{m})$: $\mathsf{ct} \leftarrow \Pi.\mathsf{Enc}(\mathsf{msk}, (\mathbf{m}, \perp, \perp))$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: $\mathbf{m}' \leftarrow \Pi.\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$.

Simulators for encryption and keygen are described as follows:

$\mathsf{Sim}^{\mathsf{CT}}(\mathsf{msk}, \{f_1(\mathbf{m}), f_2(\mathbf{m})\})$ outputs $\Pi.\mathsf{Enc}(\mathsf{msk}, (\perp, f_1(\mathbf{m}), f_2(\mathbf{m})))$.

$\mathsf{Sim}^{\mathsf{SK}}(\mathsf{msk}, 1^{|f|})$ outputs $\Pi.\mathsf{KeyGen}(\mathsf{msk}, E[1^{|f|}, b])$ for the b-th key query, for $b \in \{1, 2\}$.

Security:

Firstly, observe that the simulator thus constructed is valid since its output does not depend on the function f or input $\mathbf{m}$. Then proof of security follows directly from Ada-IND security of Π, because for any set of queries consisting of two functions f_1 and f_2 and polynomial many inputs $\{\mathbf{m}_i\}$, $f_b(\mathbf{m}_i) = g_b(\mathbf{z}_i)$ for $b \in \{1, 2\}$, where g_b is the circuit defined as $(E[1^{|f_b|}, b])$ and $\mathbf{z}_i = (\perp, f_1(\mathbf{m}_i), f_2(\mathbf{m}_i))$ used by the simulators to generate the simulated keys and ciphertexts, respectively. ∎

3.3 ATTRIBUTE BASED ENCRYPTION

For the definition of key-policy (KPABE) and ciphertext-policy ABE , we refer to Section 2.2.

Boneh et al. [43] provided a construction of kpABE which satisfies key compactness and ciphertext succinctness. The following theorem summarizes the efficiency properties of their construction.

Theorem 3.3 (Properties of [43])**.** *There exists a key-policy ABE scheme* $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Enc}, \mathsf{kpABE.Dec})$ *for function class* $C_{\ell,d}$ *which is selectively secure under the LWE assumption and has the following properties. In particular:*

1. *Key Compactness. We have* $|\mathsf{ABE.sk}_C| \leq \mathrm{poly}(\lambda, d)$ *for any* $C \in C_{\ell,d}$*, where* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ *and* $\mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}, C)$*. In particular, the length of the secret key is independent of the attribute length* ℓ *and the size of the circuit* C*.*

2. *Parameters Succinctness. We have* $|\mathsf{ABE.mpk}|, |\mathsf{ABE.msk}| \leq \mathrm{poly}(\lambda, d, \ell)$ *and* $|\mathsf{ABE.ct}| \leq \mathrm{poly}(\lambda, d, \ell) + |\mu|$ *for any* $\mathbf{x} \in \mathcal{X}_\lambda$ *and* $\mu \in \mathcal{M}_\lambda$*, where* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$ *and* $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, \mu)$*.*

3. *Online-Offline Decryption. The decryption algorithm* $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}_\mathbf{x}, \mathbf{x})$ *can be divided into two parts, which we call*

   - $\mathsf{Dec}^{\mathsf{off}}(\mathsf{mpk}, C, \mathbf{x}) \rightarrow \mathsf{off}$*, which performs the heavier computation that involves the circuit* C *and attribute* $\mathbf{x}$ *offline without knowing the ciphertext* $\mathsf{ct}_\mathbf{x}$ *or the secret key* sk_C *to get a "help"* off*. We have that the size of* off *is* $\mathrm{poly}(\lambda, \ell, d)$ *and the depth of the circuit* $\mathsf{Dec}^{\mathsf{off}}(\cdot,\cdot,\cdot)$ *is bounded by* $\mathrm{poly}(\lambda, \ell, \mathsf{depth}(C))$*, where* $\mathsf{depth}(C)$ *is the depth of the circuit* C*.*

   - $\mathsf{Dec}^{\mathsf{on}}(\mathsf{sk}_C, \mathsf{ct}_\mathbf{x}, \mathsf{off})$ *takes the help* off *generated offline along with the secret key* sk_C *and the ciphertext* $\mathsf{ct}_\mathbf{x}$ *and outputs the underlying message* μ*. We note that this part does not take* C *as input and in particular, the size of the circuit* $\mathsf{Dec}^{\mathsf{on}}(\cdot,\cdot,\cdot)$ *is* $\mathrm{poly}(\lambda, \ell, d)$*, which is independent from the size of the circuit* C*.*

We will provide the detail of the kpABE scheme given by [43] in Sec. 3.3.1. There, we

will show that the scheme satisfies the online-offline decryption property defined above. We will also use the cpABE scheme given by [169]. The following theorem summarizes the properties of the scheme.

Theorem 3.4 (Properties of [169])**.** *There exists a ciphertext policy ABE scheme* $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Enc}, \mathsf{cpABE.Dec})$ *for function class* $C_{\ell,d}$*, which is very selectively secure under the evasive and tensor LWE assumption and has the following properties. In particular:*

1. *Ciphertext Compactness. We have* $|\mathsf{cpABE.ct}| \leq \mathrm{poly}(\lambda, d) + |\mu|$ *for any* $C \in C_{\ell,d}$ *and* $\mu \in \mathcal{M}_\lambda$*, where* $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ *and* $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C, \mu)$*. In particular, the size of the ciphertext is independent from the size of the circuit* C *and its input length.*

2. *Parameters Succinctness. We have* $|\mathsf{cpABE.mpk}|, |\mathsf{cpABE.msk}|, |\mathsf{cpABE.sk}_{\mathbf{x}}| \leq \mathrm{poly}(\lambda, \ell, d)$ *for any* $\mathbf{x} \in \{0,1\}^\ell$*, where* $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ *and* $\mathsf{cpABE.sk} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathbf{x})$.

3.3.1 Key-Policy ABE by Boneh et al. [43]

We will use the kpABE scheme proposed by Boneh et al. [43]. We provide the description of the scheme in the following while showing that the decryption algorithm can be divided into two phases as stated in Theorem 3.3. We note that the presentation here is largely based on [27].

Lattice Evaluation. The following is an abstraction of the evaluation procedure in previous LWE based FHE and ABE schemes. We follow the presentation by Tsabary [162].

Lemma 3.5 (Fully Homomorphic Computation [43])**.** *There exists a pair of deterministic algorithms* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *with the following properties.*

- $\mathsf{EvalF}(\mathbf{B}, F) \to \mathbf{H}_F$*. Here,* $\mathbf{B} \in \mathbb{Z}_q^{n \times m\ell}$ *and* $F : \{0,1\}^\ell \to \{0,1\}$ *is a circuit.*

- $\mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B}) \to \widehat{\mathbf{H}}_{F,\mathbf{x}}$*. Here,* $\mathbf{x} \in \{0,1\}^\ell$ *and* $F : \{0,1\}^\ell \to \{0,1\}$ *is a circuit with depth* d*. We have* $[\mathbf{B} - \mathbf{x} \otimes \mathbf{G}]\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G}$ *mod* q*, where we*

denote $[x_1\mathbf{G}\| \cdots \|x_k\mathbf{G}]$ *by* $\mathbf{x} \otimes \mathbf{G}$. *Furthermore, we have* $\|\mathbf{H}_F\|_\infty \leq m^{O(d)}$ *and* $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \leq m^{O(d)}$.

- *The running time of* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *is bounded by* $\mathrm{poly}(n, m, \log q, d)$.

Note that the last item implies that the circuits computing EvalF and EvalFX can be implemented with depth $\mathrm{poly}(n, m, \log q, d)$.

Key-Policy ABE by Boneh et al. [43]

The scheme supports the circuit class $C_{\ell(\lambda),d(\lambda)}$, which is the set of all circuits with input length $\ell(\lambda)$ and depth at most $d(\lambda)$ with arbitrary $\ell(\lambda) = \mathrm{poly}(\lambda)$ and $d(\lambda) = \mathrm{poly}(\lambda)$. In our case, we will set $d(\lambda) = \omega(\log \lambda)$. In the description below, we focus on the case where the message space is $\{0, 1\}$ for simplicity. To encrypt a long message, we run the construction in parallel to encrypt an SKE key $K \in \{0, 1\}^\lambda$ and then use it to encrypt the message.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda), m = m(\lambda)$, noise distributions χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified later. It then proceeds as follows.

1. Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
2. Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_\ell) \leftarrow (\mathbb{Z}_q^{n\times m})^\ell$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
3. Output the master public key $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and the master secret key $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$.

$\mathsf{KeyGen}(\mathsf{msk}, C)$: The key generation algorithm takes as input the master public key mpk, the master secret key msk, and a circuit $C \in C_{\ell,d}$ and proceeds as follows.

1. Set $F := \neg C$ to be the same circuit as C except that the output bit is flipped. [7]
2. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.

[7] While we follow the standard definition of ABE and require the decryption to be possible when $C(x) = 1$, it is when $C(\mathbf{x}) = 0$ in [43]. To fill the gap, we flip the output bit.

3. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\top \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\top)$.

4. Output the secret key $\mathsf{sk}_C := \mathbf{r}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$: The encryption algorithm takes as input the master public key mpk, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$ and proceeds as follows.

1. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{e}_{i,b} \leftarrow \widetilde{\chi^m}$ for $i \in [\ell]$ and $b \in \{0,1\}$, where $\widetilde{\chi^m}$ is the distribution obtained by first sampling $\mathbf{x} \leftarrow \chi^m$ and $\mathbf{S} \leftarrow \{-1,1\}^{m\times m}$ and then outputting $\mathbf{Sx}$.

2. Compute

$$\text{For all } i \in [\ell],\ b \in \{0,1\}, \psi_{i,b} := \mathbf{s}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{e}_{i,b} \in \mathbb{Z}_q^m$$
$$\psi_{2\ell+1} := \mathbf{sA} + \mathbf{e} \in \mathbb{Z}_q^m,\ \psi_{2\ell+2} := \mathbf{su}^\top + e_0 + \mu\lceil q/2\rceil \in \mathbb{Z}_q,$$

3. Output the ciphertext $\mathsf{ct}_\mathbf{x} := (\{\psi_{i,x_i}\}_{i\in[\ell]}, \psi_{2\ell+1}, \psi_{2\ell+2})$, where x_i is the i-th bit of $\mathbf{x}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}_\mathbf{x}, \mathbf{x})$: The decryption algorithm takes as input the master public key mpk, a secret key sk_C for a circuit C, and a ciphertext $\mathsf{ct}_\mathbf{x}$ for an attribute $\mathbf{x}$ and proceeds as follows. The decryption algorithm can be divided into offline and online phase $\mathsf{Dec}^{\mathsf{off}}$ and $\mathsf{Dec}^{\mathsf{on}}$, respectively, as defined below.

$\mathsf{Dec}^{\mathsf{off}}(\mathsf{mpk}, C, \mathbf{x})$:

1. Compute $C(\mathbf{x})$ and $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalF}(F, \mathbf{x}, \mathbf{B})$, where $F = \neg C$.

2. Output $\mathsf{off} = (C(\mathbf{x}), \widehat{\mathbf{H}}_{F,\mathbf{x}})$.

$\mathsf{Dec}^{\mathsf{on}}(\mathsf{sk}_C, \mathsf{ct}_\mathbf{x}, \mathsf{off})$:

1. Parse off as $(C(\mathbf{x}), \widehat{\mathbf{H}}_{F,\mathbf{x}})$ and $\mathsf{ct}_\mathbf{x} \to (\{\psi_{i,x_i} \in \mathbb{Z}_q^m\}_{i\in[\ell]}, \psi_{2\ell+1} \in \mathbb{Z}_q^m, \psi_{2\ell+2} \in \mathbb{Z}_q)$, and $\mathsf{sk}_C = \mathbf{r} \in \mathbb{Z}^{2m}$. If any of the component is not in the corresponding domain or $C(\mathbf{x}) = 0$, output $\perp$.

2. Concatenate $\{\psi_{i,x_i}\}_{i\in[\ell]}$ to form $\psi_\mathbf{x} = (\psi_{1,x_1}, \ldots, \psi_{\ell,x_\ell})$.

3. Compute
$$\psi' := \psi_{2\ell+2} - [\psi_{2\ell+1} \| \psi_{\mathbf{x}} \widehat{\mathbf{H}}_{F,\mathbf{x}}] \mathbf{r}^\top.$$

4. Output 0 if $\psi' \in [-B, B]$ and 1 if $[-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$.

Parameters and Security. We choose the parameters for the scheme as follows for concreteness:

$$d = \log n \log\log n, \quad n = \tilde{\Theta}(\lambda^c), \quad m = n^{1.1} \log q, \quad q = n^{O(\log^3 n)},$$
$$\chi = \mathsf{SampZ}(3\sqrt{n}), \quad \tau_0 = n \log q \log m, \quad \tau = n^{O(\log^2 n)} \quad B = n^{O(\log^2 n)},$$

where c is some constant (e.g., $c = 1$).

It is easy to see that the efficiency requirement stated in Theorem 3.3 directly follows from the above parameter settings and Lemma 3.5.

Theorem 3.6 (Adapted from [43])**.** *The above scheme satisfies* Sel-INDr *security (Definition 2.6) if we assume the LWE assumption with $n^{O(\log^3 n)}$ approximation factor.*

3.3.2 Lockable Obfuscation

We define lockable obfusctaion [111, 172] below. Consider a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and output space $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$, i.e, $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}$. A lockable obfuscation scheme for $\mathcal{F}$ consists of algorithms Obf and Eval with the following syntax:

$\mathsf{Obf}(1^\lambda, f, \alpha) \to \tilde{f}$. The obfuscation algorithm takes as input the security parameter λ, a function $f \in \mathcal{F}_\lambda$ and a lock value $\alpha \in \mathcal{Y}_\lambda$. It outputs an obfuscated program $\tilde{f}$.

$\mathsf{Eval}(\tilde{f}, x) \to 1 \cup \{\perp\}$. The evaluation algorithm takes as input the obfuscated program $\tilde{f}$ and an input $x \in \mathcal{X}_\lambda$. It outputs 1 or $\perp$.

Definition 3.7 (Correctness)**.** A lockable obfuscation scheme is said to be correct if it satisfies the following properties:

1. For all $\lambda \in \mathbb{N}, f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$ and $\alpha \in \mathcal{Y}_\lambda$ such that $f(x) = \alpha$, we have

$$\mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, f, \alpha), x\right) = 1$$

2. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}, f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$ and $\alpha \in \mathcal{Y}_\lambda$ such that $f(x) \neq \alpha$, we have

$$\mathsf{Pr}[\mathsf{Eval}\left(\mathsf{Obf}(1^\lambda, f, \alpha), x\right) = \perp] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins used during obfuscation.

Definition 3.8 (Security)**.** A lockable obfuscation scheme is said to be secure if there is a PPT simulator Sim such that for all $f \in \mathcal{F}_\lambda$, we have

$$\mathsf{Obf}(1^\lambda, f, \alpha) \approx_c \mathsf{Sim}(1^\lambda, 1^{|f|})$$

where $\alpha \leftarrow \mathcal{Y}_\lambda$ and the probability is taken over the randomness of the obfuscator and simulator Sim.

Theorem 3.7 ([111, 172])**.** *There exists lockable obfuscation for all circuits with lock space* $\{0,1\}^\lambda$ *from the LWE assumption.*

3.3.3 Laconic Oblivious Transfer

We define laconic oblivious transfer (LOT) [73] below. A LOT scheme consists of four algorithms crsGen, Hash, Send and Receive with the following syntax:

$\mathsf{crsGen}(1^\lambda) \to \mathsf{crs}$. It takes the security parameter λ as input and outputs a common reference string crs.

$\mathsf{Hash}(\mathsf{crs}, D) \to (\mathsf{digest}, \hat{D})$. It takes as input a common reference string crs and a database $D \in \{0,1\}^*$ and outputs a digest digest of the database and a state $\hat{D}$.

$\mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1) \to e$. It takes as input a common reference string crs, a digest digest, a database location $L \in \mathbb{N}$ and two messages m_0 and m_1 of length λ, and outputs a ciphertext e.

$\mathsf{Receive}^{\hat{D}}(\mathsf{crs}, e, L) \to m$. This is a RAM algorithm with random read access to $\hat{D}$. It takes as input a common reference string crs, a ciphertext e, and a database location $L \in \mathbb{N}$. It outputs a message m.

Definition 3.9 (Correctness)**.** A LOT scheme is said to be correct if for any database D of size at most $M = \mathrm{poly}(\lambda)$ for any polynomial function $\mathrm{poly}(\cdot)$, any memory location $L \in [M]$, and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, we have

$$\Pr\left[m = m_{D[L]} : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda) \\ (\mathsf{digest}, \hat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D) \\ e \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1) \\ m \leftarrow \mathsf{Receive}^{\hat{D}}(\mathsf{crs}, e, L) \end{array} \right] = 1,$$

where the probability is taken over the random choices made by crsGen and Send.

Definition 3.10 (Sender Privacy Against Semi-Honest Receivers)**.** There exists a PPT simulator LOTSim such that the following holds. For any database D of size at most $M = \mathrm{poly}(\lambda)$ for any polynomial function $\mathrm{poly}(\cdot)$, any memory location $L \in [M]$, and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, let $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda)$ and $\mathsf{digest} \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$. Then it holds that

$$(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1)) \approx_c (\mathsf{crs}, \mathsf{LOTSim}(D, L, m_{D[L]})).$$

Theorem 3.8 ([73, 83, 58, 84])**.** *There exists laconic oblivious transfer where the length of* digest *is a fixed polynomial in security parameter λ, independent of the size of the database assuming either the Computational Diffie-Hellman assumption or the Factoring assumption or the Learning with Errors assumption. Moreover, the algorithm* Hash *runs in time $|D| \cdot \mathrm{poly}(\log |D|, \lambda)$,* Send *and* $\mathsf{Receive}$ *run in time* $\mathrm{poly}(\log |D|, \lambda)$ *for any database D of size at most* $\mathrm{poly}(\lambda)$ *for any polynomial function* $\mathrm{poly}(\cdot)$.

3.4 REVOCABLE PREDICATE ENCRYPTION

We define revocable predicate encryption (RPE), in both public and secret key setting. Since the two notions differ only in the encryption algorithm, we present them here in a unified way.

Definition 3.11. A RPE scheme for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in[\mathbb{N}]}$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in[\mathbb{N}]}$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda\in[\mathbb{N}]}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in[\mathbb{N}]}$ has the following probabilistic polynomial time algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes the security parameter λ as input and it outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x) \to \mathsf{sk}_{\mathsf{lb},x}$[8]. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}_\lambda$ and an attribute $x \in \mathcal{X}_\lambda$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$.

$\mathsf{Enc}(\mathsf{ek}, f, m, L) \to \mathsf{ct}$. The encryption algorithm takes as input the encryption key ek, a function f, a message $m \in \mathcal{M}_\lambda$, and a revocation list $L \subseteq \mathcal{L}_\lambda$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, \mathsf{ct}, L) \to m^{\mathsf{Pr}}ime$. The decryption algorithm takes the secret key $\mathsf{sk}_{\mathsf{lb},x}$, a ciphertext ct, and a revocation list L and it outputs $m^{\mathsf{Pr}}ime \in \mathcal{M}_\lambda \cup \{\perp\}$.

In *public-key RPE*, we take $\mathsf{ek} = \mathsf{mpk}$ in the Enc algorithm, and in *secret-key RPE*, we take $\mathsf{ek} = \mathsf{msk}$. Furthermore, there is an additional algorithm in the secret key setting defined as follows:

$\mathsf{Broadcast}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. On input the master public key, a message m, and a revocation list $L \subseteq \mathcal{L}_\lambda$, the broadcast algorithm outputs a ciphertext ct.

[8] We want to point out that the secret key $\mathsf{sk}_{\mathsf{lb},x}$ does not hide the corresponding label lb and attribute x and we assume these to be included in the secret key.

Definition 3.12 (Correctness)**.** A revocable predicate encryption scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, label $\mathsf{lb} \in \mathcal{L}_\lambda$, attributes $x \in \mathcal{X}_\lambda$, predicates $f \in \mathcal{F}_\lambda$ such that $f(x) = 1$, all messages $m \in \mathcal{M}_\lambda$ and any set of revoked users $L \subseteq \mathcal{L}_\lambda$ such that $\mathsf{lb} \notin L$, if we set $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x)$, then the following holds

$$\Pr\left[\ \mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, \mathsf{ct}, L) = m\ \right] \geq 1 - \mathsf{negl}(\lambda),$$

for $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, f, m, L)$ (*Encryption correctness*) and $\mathsf{ct} \leftarrow \mathsf{Broadcast}(\mathsf{mpk}, m, L)$ (*Broadcast correctness*).

Security. In the following security definitions, we assume for simplicity that the adversary does not make key queries for same input (lb, x) more than once.

Definition 3.13 (q-query Message Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query message hiding property if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\ \beta^{\mathsf{Pr}ime} = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{ek}, f, m_\beta, L); \\ \beta^{\mathsf{Pr}ime} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\ \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption oracle $\mathsf{Enc}(\mathsf{ek}, \cdot, \cdot, \cdot)$, and $\mathcal{A}$ is admissible if and only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, either $f(x) = 0$ or $\mathsf{lb} \in L$.

Definition 3.14 (q-query Selective Message Hiding)**.** This is the same as the Def 3.13 except that $\mathcal{A}$ outputs the revocation list L in the beginning of the game, before the Setup algorithm is run.

Definition 3.15 (q-query Very Selective Message Hiding)**.** This is the same as the Def 3.14 except that $\mathcal{A}$ outputs all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle in the beginning of the game, before the Setup algorithm is run.

Definition 3.16 (q-query Function Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query function hiding property if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\beta^{\mathsf{Pr}ime} = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f_0, f_1, m, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{ek}, f_\beta, m, L); \\ \beta^{\mathsf{Pr}ime} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{ek},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption oracle $\mathsf{Enc}(\mathsf{ek}, \cdot, \cdot, \cdot)$, and $\mathcal{A}$ is admissible if and only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Definition 3.17 (q-query Selective Function Hiding)**.** This is the same as the Def 3.16 except that $\mathcal{A}$ outputs the revocation list L in the beginning of the game, before the Setup algorithm is run.

The following security notion is defined only for secret-key RPE scheme.

Definition 3.18 (q-query Selective Broadcast Security)**.** Let $q(\cdot)$ be any fixed polynomial. A RPE scheme satisfies q-query selective broadcast security if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every PPT adversary $\mathcal{A}$, for every $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$ and any subset of revoked users $L \subseteq \mathcal{L}_\lambda$, the following holds

$$\Pr\left[\beta^{\mathsf{Pr}ime} = \beta : \begin{array}{l} L \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ f, m \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{msk}, f, m, L); \\ \mathsf{ct}_1 \leftarrow \mathsf{Broadcast}(\mathsf{mpk}, m, L); \\ \beta^{\mathsf{Pr}ime} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the encryption $\mathsf{Enc}(\mathsf{msk}, \cdot, \cdot, \cdot)$ oracle and $\mathcal{A}$ is admissible if and only if $f(x) = 1, \forall x \in \mathcal{X}_\lambda$.

Remark 3. In the *public-key* RPE scheme, the adversary $\mathcal{A}$ can itself simulate the encryption oracle $\mathsf{Enc}(\mathsf{ek}, \cdot, \cdot, \cdot)$, as $\mathsf{ek} = \mathsf{mpk}$ in this setting. Therefore, in public-key setting, we refer to the security definitions without imposing the q-query bound on the encryption oracle.

Remark 4. We note that when the message space is binary, function space $\mathcal{F}_\lambda$ is polynomially small and q is a constant, the weaker security definitions where adversary outputs the challenge function f, the challenge message m and the SK-Enc query functions $\{\bar{f}_i\}_{i\in[q]}$ at the beginning of the game, before the $\mathsf{Setup}(1^\lambda)$ algorithm is run, is equivalent to the definitions where the adversary outputs $f, m, \{\bar{f}_i\}_{i\in[q]}$ adaptively. First, the functions can be guessed with polynomial loss. Furthermore, if we restrict the message space to be binary, we can guess the challenge message as well. To extend the message space, we can encrypt each bit by parallel systems.

3.5 PUBLIC-KEY RPE FROM FE AND LWE

3.5.1 Construction

In this section we provide our construction of a public key RPE scheme $\mathsf{RPE} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ from polynomial hardness assumptions. We assume that $|\mathcal{F}_\lambda|$ and $|\mathcal{M}_\lambda|$ are bounded by some polynomial in λ. The restriction on $|\mathcal{F}_\lambda|$ is sufficient for our purpose and the restriction on $|\mathcal{M}_\lambda|$ can be removed by running the scheme in parallel.

Our construction uses the following building blocks:

1. A Sel-INDr secure key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.Enc}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Dec})$ for circuit class $C_{\ell(\lambda),d(\lambda)}$ with parameter succinctness and key compactness (Theorem 3.3). Here $\ell(\lambda)$ is the input length and is the length of labels in our setting and the depth of the circuit is $d(\lambda) \in \omega(\log \lambda)$ to support unbounded revocation list. The message space of the scheme kpABE is $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ and $\mathcal{CT}_{\mathsf{kpABE}}$ denotes the ciphertext space. We assume that uniform sampling from $\mathcal{CT}_{\mathsf{kpABE}}$ is efficiently possible without any

parameter.

2. A (fully) compact, selectively secure, public-key functional encryption scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ that supports polynomial sized circuits. We assume that the message space is sufficiently large so that it can encrypt an ABE master public key, a (description of) function $f \in \mathcal{F}_\lambda$, a PRF key, two secret keys of SKE, and a trit $\mathsf{mode} \in \{0, 1, 2\}$.

3. A PRF $F : \{0,1\}^\lambda \times \mathcal{X} \to \{0,1\}^t$ where t is the length of the randomness used in kpABE encryption (Def. 2.10)

4. A symmetric key encryption schemes $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with pseudorandom ciphertexts (Def. 3.1) We let $\mathcal{CT}_{\mathsf{SKE}}$ denote the ciphertext space of SKE.[9] We assume that uniform sampling from $\mathcal{CT}_{\mathsf{SKE}}$ is efficiently possible without any parameter.

We describe our construction below.

$\mathsf{RPE.Setup}(1^\lambda) \to (\mathsf{RPE.mpk}, \mathsf{RPE.msk})$. The setup algorithm does the following:

- Generate $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- Output $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and $\mathsf{RPE.msk} = \mathsf{FE.msk}$.

$\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x) \to \mathsf{RPE.sk}_{\mathsf{lb},x}$. The key generation algorithm does the following:

- Sample random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
- Construct a circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ which has the label lb, attribute x, γ_1, γ_2 and δ hardwired, as defined in Figure 3.2.
- Compute $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
- Output $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$.

$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m, L) \to \mathsf{RPE.ct}$. The encryption algorithm does the following:

[9] We note that we use the same ciphertext space for simplicity even though messages with different lengths are going to be encrypted. To have the same ciphertext space, we can, for example, pad short messages to be some fixed length, which is possible when the message length is bounded by some polynomial.

- Parse $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$.
- Sample a F key $K \leftarrow \{0,1\}^\lambda$.
- Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
- Compute $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m, K, 0, \perp, \perp))$.
- Construct a circuit C_L, with revocation list L hardwired defined as follows: On input a label $\mathsf{lb} \in \mathcal{L}_\lambda$,

$$C_L(\mathsf{lb}) = 1 \text{ if and only if } \mathsf{lb} \notin L. \tag{3.1}$$

  Compute $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.
- Output $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$.

$\mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},x}, \mathsf{RPE.ct}, L) \to m'$. The decryption algorithm does the following:

- Parse $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ and $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$.
- Compute $\mathsf{ct}' = \mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct})$.
- Construct circuit C_L from L and compute $m' = \mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, C_L, \mathsf{ct}', \mathsf{lb})$.
- Output m'.

Correctness. We now show that the above construction is correct via the following theorem.

Theorem 3.9. *Suppose* FE *and* kpABE *schemes are correct. Then the above construction satisfies the encryption correctness (Def. 3.12).*

Proof. Firstly, for any label lb, attribute x, and function f such that $f(x) = 1$, we have $\mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct}) = \mathsf{kpABE.ct}_{\mathsf{lb}}$, where $\mathsf{kpABE.ct}_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, m; F(K, (\mathsf{lb}, x)))$, by the correctness of FE and the definition of $\mathsf{Re\text{-}Enc}$. We then observe that $C_L(\cdot)$ can be implemented with

Function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$

Hardwired values: A label lb, an attribute x, and SKE ciphertexts γ_1, γ_2, and δ.

Inputs: A kpABE master public key $\mathsf{kpABE.mpk}$, a function $f \in \mathcal{F}_\lambda$, a message $m \in \mathcal{M}_\lambda$, a F key K, a trapdoor mode $\mathsf{mode} \in \{0, 1, 2\}$ and ske keys $\mathsf{ske.key}_1$ and $\mathsf{ske.key}_2$.

Output : A kpABE ciphertext.

1. Parse the input as $(\mathsf{ABE.mpk}, f, m, K, \mathsf{mode}, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$.
2. Set $\tilde{m} = \begin{cases} m & \text{if } f(x) = 1 \\ 0 & \text{if } f(x) = 0. \end{cases}$
3. Compute $\mathsf{kpABE.ct}_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.
4. Compute $\mathsf{flag} = \mathsf{ske.Dec}(\mathsf{ske.key}_2, \delta)$.
5. Compute $\mathsf{out}_i = \mathsf{ske.Dec}(\mathsf{ske.key}_i, \gamma_i)$ for $i \in \{1, 2\}$.
6. If $\mathsf{mode} = 0$, output $\mathsf{kpABE.ct}_{\mathsf{lb}}$.
7. If $\mathsf{mode} = 1$, output out_1.
8. If $\mathsf{mode} = 2$, output $\begin{cases} \mathsf{out}_2 & \text{if } \mathsf{flag} = 1 \\ \mathsf{kpABE.ct}_{\mathsf{lb}} & \text{if } \mathsf{flag} = 0. \end{cases}$

Figure 3.2: Function to compute kpABE ciphertexts.

depth $O(\log(|L| \cdot |\mathsf{lb}|)) = O(\log \mathrm{poly}(\lambda)) = O(\log \lambda) \leq d$ and thus $C_L(\cdot) \in C_{\ell,d}$. Then if $\mathsf{lb} \notin L$ we have $C_L(\mathsf{lb}) = 1$ and hence from the correctness of the kpABE scheme it follows that $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, C_L, \mathsf{kpABE.ct}_{\mathsf{lb}}, \mathsf{lb}) = m$. So the decryption correctly recovers the message when $f(x) = 1$ and $\mathsf{lb} \notin L$. ■

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that the size of each parameter is independent from $|L|$. We note that $|f|$ refers to the description size of the function, not the circuit size that implements the function. When $|x|$ is very long and f has succinct description, the former can be much shorter

than the latter.

1. Public key size $|\mathsf{RPE.mpk}|$: We have $|\mathsf{RPE.mpk}| = |\mathsf{FE.mpk}|$. Since we assumed that FE is fully compact (Def. 3.4), the length of $\mathsf{FE.mpk}$ only depends on the input length of $\mathsf{Re\text{-}Enc}$. We have that the input length is $|\mathsf{ABE.mpk}| + |f| + |m| + |K| + |\mathsf{mode}| + 2|\mathsf{SKE.key}| = |\mathsf{ABE.mpk}| + |f| + O(\lambda)$. We have $|\mathsf{ABE.mpk}| \leq \mathrm{poly}(\lambda, |\mathsf{lb}|, d) = \mathrm{poly}(\lambda, |\mathsf{lb}|)$ by Theorem 3.3. The total length is therefore $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$.

2. Secret key size $|\mathsf{RPE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RPE.sk}_{\mathsf{lb},x}| = |\mathsf{FE.sk}_{\mathsf{lb},x}|$. Since the size of the latter is polynomially dependent on the size of $\mathsf{Re\text{-}Enc}$, we evaluate its size. We can see that the size of $\mathsf{Re\text{-}Enc}$ is polynomial in the total length of the input and the hardwired values. The length of the input is bounded by $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$ as analyzed in the above item. The length of the hardwired values are $|\mathsf{lb}| + |x| + |\gamma_1| + |\gamma_2| + |\delta|$. We have $|\gamma_1| + |\gamma_2| + |\delta| = 3|\gamma_2|$[10] and $|\gamma_2| = \mathrm{poly}(\lambda, \mathsf{kpABE.ct}_{\mathsf{lb}}) = \mathrm{poly}(\lambda, \mathsf{lb}, d) = \mathrm{poly}(\lambda, \mathsf{lb})$. Therefore, the size of $\mathsf{Re\text{-}Enc}$ is $\mathrm{poly}(\lambda, |x|, |f|, |\mathsf{lb}|)$ and so is the size of the secret key.

3. Ciphertext size $|\mathsf{RPE.ct}|$: We have $|\mathsf{RPE.ct}| = |\mathsf{kpABE.mpk}| + |\mathsf{kpABE.sk}_L| + |\mathsf{FE.ct}|$. We have $|\mathsf{ABE.mpk}| = \mathrm{poly}(\lambda, |\mathsf{lb}|)$ as we showed in the first item. We also have $|\mathsf{kpABE.sk}_L| \leq \mathrm{poly}(\lambda, d) \leq \mathrm{poly}(\lambda)$ by the key compactness of kpABE (3.3). By similar analysis to the first item, full compactness of FE implies $|\mathsf{FE.ct}| \leq \mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$. Therefore, the overall length of the ciphertext is $\mathrm{poly}(\lambda, |f|, |\mathsf{lb}|)$.

3.5.2 Security

Now we prove that the above construction of RPE satisfies both function hiding and message hiding security.

Function Hiding

Theorem 3.10. *Assume that F is a secure* PRF, SKE *is correct and secure,* FE *and* kpABE *are secure as per definitions 3.3 and 2.6, respectively. Furthermore, assume $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$ and $|\mathcal{M}_\lambda| \leq \mathrm{poly}(\lambda)$. Then the* RPE *constructed above is function hiding (Def. 3.16).*

Proof. Recall that for function hiding we need $\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_0, m, L) \approx_c$

[10] We assumed that $|\gamma_1| = |\gamma_2| = |\delta|$. See Footnote 9.

$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_1, m, L)$, where for all the key queries (lb, x), either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

The proof proceeds via a sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hyb_0. This is the real world with $\beta = 0$, i.e. the challenge ciphertext is computed using the function f_0. We write the complete game here to set up the notations and easy reference in later hybrids.

1. The adversary outputs the challenge functions f_0 and f_1 and the challenge message m[11].
2. The challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to the adversary. The challenger then responds to different queries from $\mathcal{A}$ as follows:
3. **Key Queries :** For each key query (lb, x), the challenger does the following:
   - Samples random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Defines the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in Figure 3.2 and computes $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
   - Returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to the adversary.
4. **Challenge Query:** When the adversary outputs the revocation list L for the challenge query, the challenger does the following:
   - Samples a F key K.
   - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
   - Computes $\mathsf{FE.ct}$ as
   $$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)).$$
   - Defines C_L as in Eq. (3.1) and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

[11] To keep the proofs and notations simple, we let f_0, f_1, m to be given selectively. This is sufficient to achieve security as in Def 3.16, as mentioned in Remark 4.

- Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to the adversary.

5. In the end, the adversary outputs a bit β'.

Hyb_1. This hybrid is same as the previous hybrid except the following changes:

- The challenger generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ in the beginning of the game after the adversary outputs the challenge functions f_0 and f_1 and the challenge message m. It also samples a SKE secret key $\mathsf{SKE.key}_1$ and a F key K.

- For each key query (lb, x), γ_1 is computed differently. In particular, the challenger does the following

  - Sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$ and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

  - Sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hyb_2. This hybrid is same as the previous hybrid except the following changes:

- The challenger samples two SKE secret keys $\mathsf{SKE.key}_1$ and $\mathsf{SKE.key}_2$ in the beginning of the game (after receiving f_0, f_1, m from $\mathcal{A}$).

- For each key query (lb, x), γ_2 and δ are computed differently from the previous hybrid. In particular, the challenger does the following:

  - Sets $\mathsf{flag} = \begin{cases} 1 & \text{if } f_0(x) \neq f_1(x) \\ 0 & \text{otherwise.} \end{cases}$

  - Sets $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. We note that $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ and γ_1 are computed as in the previous hybrid.

Hyb_3. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge

ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hyb_4. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as follows:

- If $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, where $r \leftarrow \{0,1\}^t$.
- Else, if $\mathsf{flag} = 0$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

Hyb_5. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)).$$

Hyb_6. This hybrid is same as the previous hybrid except that for each key query (lb, x), γ_1 is set differently as $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hyb_7. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is sampled uniformly from $\mathcal{CT}_{\mathsf{kpABE}}$, i.e., $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

Hyb_8. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, K, 2, \perp, \mathsf{SKE.key}_2)).$$

We note that in the hybrids hereafter, we rewind the changes made in the preceding hybrids.

Hyb_9. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$ for $r \leftarrow \{0,1\}^t$, where $\tilde{m}$ is now defined as m if $f_1(x) = 1$ and 0 otherwise.

Hyb_{10}. This hybrid is same as the previous hybrid except that for all the key queries (lb, x), the challenger sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hyb_{11}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hyb_{12}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

Hyb_{13}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_1, m, K, 0, \perp, \perp)).$$

Hyb_{14}. This hybrid is same as the previous hybrid except that γ_2 and δ_2 are set as $\gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hyb_{15}. This hybrid is same as the previous hybrid except that γ_1 is set as $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Note that this is the real world with $\beta = 1$, i.e., f_1 is encrypted in the challenge ciphertext.

Indistinguishability of hybrids We now show that the consecutive hybrids are indistinguishable.

Claim 3.11. Assume that SKE is secure, then $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of SKE scheme with advantage ϵ. The reduction is as follows.

1. The SKE challenger samples $\mathsf{SKE.key}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the SKE security game with $\mathcal{B}$.

2. $\mathcal{B}$ invokes $\mathcal{A}$, which then outputs the challenge functions f_0 and f_1 and the challenge message m.

3. $\mathcal{B}$ generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{A}$.
   $\mathcal{B}$ also generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and samples a F key K.

4. **Key Queries :** Whenever $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:

   - It sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$ and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

   - It sends $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ as the challenge message to the SKE challenger. The SKE challenger returns $\mathsf{ct}_{\hat{\beta}}$ to $\mathcal{B}$, where $\mathsf{ct}_0 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and $\mathsf{ct}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

   - It sets $\gamma_1 = \mathsf{ct}_{\hat{\beta}}$, samples random values $\gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and computes the FE key for the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

5. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following:

   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp))$.

   - Defines C_L as in Eq. 3.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

- Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the SKE challenger.

We observe that if the SKE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_0, else Hyb_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_0) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1)| = \epsilon$ (by assumption).

■

Claim 3.12. Assume that SKE is secure. Then $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. The proof follows the same steps as that for the claim 3.11 and hence omitted. ■

Claim 3.13. Assume that FE satisfies selective security (Def. 3.3) and SKE is correct. Then $\mathsf{Hyb}_2 \approx_c$ and Hyb_3.

Proof. We show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of FE with the same advantage ϵ. $\mathcal{B}$ is defined as follows:

1. Upon being invoked by the FE challenger, $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge functions f_0 and f_1 and the challenge message m.

2. $\mathcal{B}$ samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, a F key K and generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.

3. It sets $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ and sends (μ_0, μ_1) to the FE challenger as challenge messages.

4. The FE challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, samples $\hat{\beta} \leftarrow \{0, 1\}$. It then computes $\mathsf{FE.ct}_{\hat{\beta}} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, \mu_{\hat{\beta}})$ and sends $(\mathsf{FE.mpk}, \mathsf{FE.ct}_{\hat{\beta}})$ to $\mathcal{B}$.

5. $\mathcal{B}$ sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends RPE.mpk to $\mathcal{A}$.

6. **Key Queries :** When $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:

- It sets $\tilde{m} = \begin{cases} m & \text{if } f_0(x) = 1 \\ 0 & \text{if } f_0(x) = 0 \end{cases}$ and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

- It sets $\mathsf{flag} = \begin{cases} 1 & \text{if } f_0(x) \neq f_1(x) \\ 0 & \text{otherwise} \end{cases}$.

- It computes $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else samples $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

- It defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in Figure 3.2 and sends a key query $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ to the FE challenger. The FE challenger returns the secret key $\mathsf{FE.sk}_{\mathsf{lb},x}$ to $\mathcal{B}$.

- $\mathcal{B}$ returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

7. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following:

   - It defines C_L as in Eq. (3.1) and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct}_{\hat{\beta}})$ to $\mathcal{A}$.

8. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the FE challenger.

We observe that if FE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_2, else Hyb_3 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_2) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_3)| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$

Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for the Re-Enc functions defined for each key query (lb, x) by $\mathcal{A}$. Next, we observe that for any function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 0, \perp, \perp)$, and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) \quad = \quad \mathsf{kpABE.ct}_{\mathsf{lb}}$$

$$= \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$$

$$\begin{aligned}\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \mathsf{ske.Dec}(\mathsf{ske.key}_1, \gamma_1)\\ &= \mathsf{kpABE.ct}'_{\mathsf{lb}}\\ &= \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))\end{aligned}$$

Thus, for all the keys queried to the FE challenger, $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) = \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1)$. This establishes the admissibility of $\mathcal{B}$. ■

Claim 3.14. Assume that F is secure, then $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. We show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against F security with the same advantage ϵ. The reduction $\mathcal{B}$ is defined as follows:

1. The F challenger samples a F key K and a bit $\hat{\beta} \leftarrow \{0, 1\}$ and starts the game with $\mathcal{B}$.

2. $\mathcal{B}$ then invokes $\mathcal{A}$ which outputs the challenge functions f_0 and f_1 and the challenge message m.

3. $\mathcal{B}$ samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends RPE.mpk to $\mathcal{A}$.

4. **Key Queries :** When $\mathcal{A}$ issues a key query (lb, x), $\mathcal{B}$ does the following:

   - If $\mathsf{flag} = 0$, it sends an evaluation query for input (lb, x) to the F challenger and gets back $F(K, (\mathsf{lb}, x))$. It then computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

   - If $\mathsf{flag} = 1$, it sends (lb, x) as a challenge query to the F challenger and gets back $r_{\hat{\beta}}$, where $r_0 = F(K, (\mathsf{lb}, x))$ and $r_1 \leftarrow \{0, 1\}^t$. It then computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r_{\hat{\beta}})$.

   - Computes $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

- Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and computes the FE key for the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

5. **Challenge Query :** When $\mathcal{A}$ outputs the revocation list L for the challenge ciphertext, $\mathcal{B}$ does the following

   - Computes $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2))$.

   - Defines C_L as in Eq. 3.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ sends β' to the SKE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_3, else Hyb_4 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_3) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_4)|$ $= \epsilon$ (by assumption).

■

Claim 3.15. Assume that FE is selectively secure (as per Def 3.3) and SKE is correct. Then $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

Proof. We show that if $\mathcal{A}$ wins with non-negligible advantage ϵ in distinguishing the two hybrids, then there exists an adversary $\mathcal{B}$ against FE security with the same advantage ϵ. The steps of the reduction are similar as in the proof of the Claim 3.13, with $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$. Hence, here we only argue the admissibility of $\mathcal{B}$ in the FE security game.

Admissibility of $\mathcal{B}$

Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for

the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ functions, where each function is defined corresponding to a key query (lb, x) by $\mathcal{A}$. Here, $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}}), \delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Also $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$ if $\mathsf{flag} = 0$; else $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$ where $r \leftarrow \{0,1\}^t$.

Next, we observe that for any function $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)$, and $\mu_1 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

$$\begin{aligned}
\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) &= \mathsf{ske.Dec}(\mathsf{ske.key}_1, \gamma_1) \\
&= \mathsf{kpABE.ct}'_{\mathsf{lb}} \\
&= \begin{cases} \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x))) & \text{if } \mathsf{flag} = 0 \\ \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r) & \text{if } \mathsf{flag} = 1 \end{cases}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \begin{cases} \mathsf{kpABE.ct}_{\mathsf{lb}} & \text{if } \mathsf{flag} = 0 \\ \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) = \mathsf{kpABE.ct}'_{\mathsf{lb}} & \text{if } \mathsf{flag} = 1 \end{cases} \\
&= \begin{cases} \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x))) & \text{if } \mathsf{flag} = 0 \\ \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r) & \text{if } \mathsf{flag} = 1 \end{cases}
\end{aligned}$$

Thus, $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0)$=$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1)$. This establishes the admissibility of $\mathcal{B}$. ■

Claim 3.16. Assume that SKE is secure, then $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

Proof. The proof is similar to the proof for the claim 3.11 and hence omitted. ■

Claim 3.17. Assume that kpABE is Sel-INDr secure (Def. 2.6), then $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$.

Proof. To argue indistinguishability between the two hybrids, we define an intermediate hybrid Hyb_{6a} as follows - this hybrid is same as Hyb_6 except that for each such pre-challenge key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. The proof then follows from the following two claims:

Claim 3.18 (1)**.** Assuming kpABE is Sel-INDr secure (Def. 2.6), $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_{6a}$.

Proof. Let Q_{fpre} be the number of pre-challenge key queries with $\mathsf{flag} = 1$[12]. Then, we further define the following sub hybrids: for $i = 0$ to Q_{fpre}, define $\mathsf{Hyb}_{6.i}$ which is same as Hyb_6, except that for the first i key queries with $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Thus, $\mathsf{Hyb}_{6.0} = \mathsf{Hyb}_6$ and $\mathsf{Hyb}_{6.Q_{\mathsf{fpre}}} = \mathsf{Hyb}_{6a}$. Next, we show that for all $i \in [Q_{\mathsf{fpre}}]$, $\mathsf{Hyb}_{6.i-1} \approx_c \mathsf{Hyb}_{6.i}$. In particular, we show that if $\mathcal{A}$ distinguishes between the two hybrids with non-negligible advantage ϵ, then there exists a PPT algorithm $\mathcal{B}$ against Sel-INDr security of kpABE with the same advantage ϵ.

Observe that the two hybrids differ only in the value of $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ used in the computation of $\mathsf{RPE.sk}_{\mathsf{lb},x}$ for the i-th key query (lb, x) with $\mathsf{flag} = 1$; in the former hybrid we have $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, while in the latter hybrid, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Now, we define the reduction $\mathcal{B}$.

1. $\mathcal{B}$ firstly invokes $\mathcal{A}$ and gets f_0, f_1 and m.
2. $\mathcal{B}$ then samples two SKE secret keys $\mathsf{SKE.key}_1, \mathsf{SKE.key}_2$, a F key K and generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to $\mathcal{A}$. It then answers different queries from $\mathcal{A}$ as follows:
3. **Key Queries**: For each key query (lb, x), $\mathcal{B}$ does the following:
   - Sets $\mathsf{flag} = 0$, if $f_0(x) = f_1(x)$; else $\mathsf{flag} = 1$. It computes $\delta = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and samples $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Computes γ_2 as follows:

[12] Note that we can upper bound Q_{fpre} as $Q_{\mathsf{fpre}} \leq |L|$.

– If $\mathsf{flag} = 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

– Else, if $\mathsf{flag} = 1$, then let this be the j-th key query with $\mathsf{flag} = 1$. Then,

* For $j < i$, $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. (Note that this does not require $\mathsf{kpABE.mpk}$).

* For $j = i$, $\mathcal{B}$ does the following:

· Sends lb and $\tilde{m}$ as the challenge attribute and message, respectively, to the kpABE challenger.

· The kpABE challenger samples $\hat{\beta} \leftarrow \{0,1\}$ and computes $\mathsf{kpABE.ct} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$, if $\hat{\beta} = 0$, else samples $\mathsf{kpABE.ct} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. The kpABE challenger sends $\{\mathsf{kpABE.mpk}, \mathsf{kpABE.ct}\}$ to $\mathcal{B}$.

· $\mathcal{B}$ then computes $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct})$.

* For $j > i$, $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$.

- Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, computes an FE key for this circuit and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

4. **Challenge Query:** When $\mathcal{A}$ outputs the revocation list L for the challenge query, $\mathcal{B}$ does the following:

- Computes
$\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2))$.

- Defines C_L as in Eq. 3.1 and sends a key query for the circuit C_L to the kpABE challenger. The kpABE challenger returns $\mathsf{kpABE.sk}_L$.

- Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

5. In the end, $\mathcal{A}$ outputs its guess bit β'. $\mathcal{B}$ sends β' to the kpABE challenger as its guess bit.

We observe that if the kpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hyb}_{6,i-1}$, else $\mathsf{Hyb}_{6,i}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{6,i-1}) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{6,i})| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$: Observe that $\mathcal{B}$ issues a single kpABE key query which is for circuit C_L and the challenge attribute is the label lb corresponding to a key query (lb, x) by $\mathcal{A}$ for which $\mathsf{flag} = 1$. Hence, by the admissibility condition of $\mathcal{A}$, $\mathsf{lb} \in L$ and hence $C_L(\mathsf{lb}) = 0$. ■

Claim 3.19 (2)**.** Assume kpABE is $\mathsf{Sel\text{-}INDr}$ secure (Def. 2.6). Then, $\mathsf{Hyb}_{6a} \approx_c \mathsf{Hyb}_7$.

Proof. To prove the claim, we again consider sub hybrids, $\mathsf{Hyb}_{6a.i}$ for $i = 0$ to $|L|$, defined as follows: let $L_{[1:j]}$ be the set of first j labels in the revocation list L. Then, $\mathsf{Hyb}_{6a.i}$ is same as Hyb_{6a} except the following changes: for any post-challenge key query (lb, x) such that $\mathsf{lb} \in L_{[1:i]}$ and $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Thus, $\mathsf{Hyb}_{6a.0} = \mathsf{Hyb}_{6a}$ and $\mathsf{Hyb}_{6a.|L|} = \mathsf{Hyb}_7$. Next we prove the following claim:

Claim 3.20. Assume kpABE is $\mathsf{Sel\text{-}INDr}$ secure (Def. 2.6). Then for $i \in [|L|]$, $\mathsf{Hyb}_{6a.i-1} \approx_c \mathsf{Hyb}_{6a.i}$.

Proof. Let lb_i be the i-th label in L. Then we observe that if there is no post-challenge key query (lb, x) such that $\mathsf{flag} = 1$ and $\mathsf{lb} = \mathsf{lb}_i$ then the two hybrids are identical. Else, we show that if $\mathcal{A}$ can distinguish between the two hybrids with non negligible advantage ϵ then there exists a PPT algorithm $\mathcal{B}$ against $\mathsf{Sel\text{-}INDr}$ security of kpABE security with the same advantage ϵ. $\mathcal{B}$ is defined as follows:

1. $\mathcal{B}$ firstly invokes $\mathcal{A}$ and gets f_0, f_1 and m.
2. It then samples a SKE secret key $\mathsf{SKE.key}_2$, a F key K and generates FE keys as $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends it to $\mathcal{A}$. It then answers different queries from $\mathcal{A}$ as follows:
3. **Pre-challenge Key Queries:** For pre-challenge key query (lb, x), $\mathcal{B}$ does the following:
   - Sets $\mathsf{flag} = 0$ if $f_0(x) = f_1(x)$; else $\mathsf{flag} = 1$ and computes $\delta = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$. It also samples $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Computes γ_2 as follows: if $\mathsf{flag} = 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$; else $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

(Note that this does not require $\mathsf{kpABE.mpk}$).

- Defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, computes an FE key for this circuit and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

4. **Challenge Query:** When $\mathcal{A}$ outputs the revocation list $L = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L|}\}$ for the challenge ciphertext, $\mathcal{B}$ does the following:

   - Sends lb_i to the kpABE challenger as the challenge attribute. The kpABE challenger samples $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and $\hat{\beta} \leftarrow \{0,1\}$, and sends $\mathsf{kpABE.mpk}$ to $\mathcal{B}$.

   - Constructs circuit C_L as defined in the construction and sends a key query for C_L to the kpABE challenger and gets $\mathsf{kpABE.sk}_L$ in response.

   - Computes
     $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2))$

   - Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to $\mathcal{A}$.

5. **Post-challenge Key Queries:** For each post-challenge key query (lb, x), $\mathcal{B}$ computes flag, δ and γ_1 as defined for the hybrid and defines $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$, where γ_2 is computed as follows:

   - if $\mathsf{flag} = 0$, $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$. Else,

     – if $\mathsf{lb} \in L_{[1:i-1]}$, $\gamma_2 = \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

     – if $\mathsf{lb} = \mathsf{lb}_i$, $\mathcal{B}$ sends challenge query with message $\tilde{m}$ to the kpABE challenger. The kpABE challenger returns a ciphertext $\mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}_i, \tilde{m}; r)$, if $\hat{\beta} = 0$; else $\mathsf{kpABE.ct} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$. Then $\mathcal{B}$ computes $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct})$. [13]

     – if $\mathsf{lb} \notin L_{[1:i]}$, then $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; r)$.

   $\mathcal{B}$ computes an FE key for $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ and returns it as the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs its guess bit β'. $\mathcal{B}$ sends β' to the kpABE challenger as its guess bit.

We observe that if the kpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hyb}_{6a,i-1}$, else

[13] In case multiple queries with $\mathsf{lb} = \mathsf{lb}_i$ are made, we need to simulte the ciphertext multiple times. In that case, we rely on multi-challenge version of $\mathsf{Sel\text{-}IND}r$, which is easily seen to be equivalent with the single challenge version.

$\mathsf{Hyb}_{6a,i}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ = $|\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)|$ = $|\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{6a,i-1}) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{6a,i})| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$: Observe that $\mathcal{B}$ issues a single ABE key query, which is for circuit C_L and the challenge attribute is $\mathsf{lb}_i \in L$. Hence, by the design of C_L, $C_L(\mathsf{lb}_i) = 0$ as desired. ■

■

■

Claim 3.21. Assume that FE is secure, then $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$.

Proof. We show that if $\mathcal{A}$ wins with non-negligible advantage ϵ in distinguishing the two hybrids, then there exists an adversary $\mathcal{B}$ against FE security with the same advantage ϵ. The steps of the reduction are similar to the proof of Claim 3.13, with $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f_1, m, K, 2, \perp, \mathsf{SKE.key}_2)$. Hence, here we only argue the admissibility of $\mathcal{B}$ in the FE security game.

Admissibility of $\mathcal{B}$:

Firstly, we observe that the only key queries that $\mathcal{B}$ issues to the FE challenger are for the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ functions, where each function is defined corresponding to a key query (lb, x) by $\mathcal{A}$. Here, $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$, $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$; else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$, where $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$ for $\mathsf{flag} = 1$. Next, we observe that for any $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function, $\mu_0 = (\mathsf{kpABE.mpk}, f_0, m, K, 2, \perp, \mathsf{SKE.key}_2)$, and

$\mu_1 = (\mathsf{kpABE.mpk}, f_1, m, K, 2, \perp, \mathsf{SKE.key}_2)$ the following holds true from the definition of Re-Enc and correctness of SKE decryption,

- when flag $= 0$, i.e $f_0(x) = f_1(x)$, $\tilde{m}$ in $\mathsf{kpABE.ct}_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}; F(K, (\mathsf{lb}, x)))$, computed inside the $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function is same for both f_0 and f_1, and hence same on both the inputs μ_0 and μ_1. So,

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) = \mathsf{kpABE.ct}_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 0).$$

$$\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) = \mathsf{kpABE.ct}_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 0).$$

- When flag $= 1$,i.e $f_0(x) \neq f_1(x)$, by definition of $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ we have

$$\begin{aligned} \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) &= \mathsf{ske.Dec}(\mathsf{ske.key}_2, \gamma_2) \\ &= \mathsf{kpABE.ct}'_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 1\ ). \end{aligned}$$

$$\begin{aligned} \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1) &= \mathsf{ske.Dec}(\mathsf{ske.key}_2, \gamma_2) \\ &= \mathsf{kpABE.ct}'_{\mathsf{lb}} \quad (\text{since } \mathsf{mode} = 2 \text{ and } \mathsf{flag} = 1). \end{aligned}$$

This establishes the admissibility of $\mathcal{B}$. ■

The rest of the hybrids, Hyb_9 to Hyb_{15}, are simply unwinding the previous hybrids and their proofs of indistinguishability are same as their corresponding counterparts in the first set of hybrids and hence, omitted. ■

Message Hiding

Theorem 3.22. *Assume that F is a secure* PRF, SKE *is correct and secure,* FE *and* kpABE *are secure as per definitions 3.3 and 2.6, respectively. Furthermore, assume $|\mathcal{F}_\lambda| \leq \text{poly}(\lambda)$ and $|\mathcal{M}_\lambda| \leq \text{poly}(\lambda)$. Then the construction for* RPE *satisfies message hiding property as defined in Def. 3.13.*

Proof. Recall that for message hiding we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m_1, L),$$

where for all the key queries (lb, x), either $f(x) = 0$ or $\mathsf{lb} \in L$. The proof is given via a similar sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$ as in the proof of Theorem 3.10. The hybrids are defined as follows:

Hyb_0. This is the real world with $\beta = 0$, i.e., the challenge ciphertext is computed using the message m_0. We write the complete game here to set up notations and easy reference in later hybrids.

1. The adversary outputs the challenge messages (m_0, m_1) and the challenge function f[14].

2. The challenger generates $(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sends $\mathsf{RPE.mpk}$ to the adversary.

3. **Key Queries :** When adversary issues a key query on (lb, x), the challenger does the following:

   - Samples random values $\gamma_1, \gamma_2, \delta \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
   - Defines the circuit $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ as in the Figure 3.2.
   - Computes $\mathsf{FE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.msk}, \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta])$.
   - Returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$ to the adversary.

4. **Challenge Query :** When the adversary outputs the revocation list L for the challenge ciphertext, the challenger does the following:

   - Samples a F key K.
   - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
   - Computes $\mathsf{FE.ct}$ as
   $$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, K, 0, \perp, \perp)).$$
   - Defines C_L as in Eq. 3.1 and computes $\mathsf{kpABE.sk}_L \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_L)$.

[14] To keep the proofs and notations simple, we let f, m_0, m_1 to be output selectively. This is sufficient to achieve security as in Def 3.13, as mentioned in Remark 4.

- Returns $\mathsf{RPE.ct} = (\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_L, \mathsf{FE.ct})$ to the adversary.

5. In the end, the adversary outputs a bit β'.

Hyb_1. This hybrid is same as the previous hybrid except the following changes:

- The challenger generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ in the beginning of the game after the adversary outputs (m_0, m_1) and f. It also samples a SKE secret key $\mathsf{SKE.key}_1$ and a F key K.

- For each key query (lb, x), γ_1 is computed differently. In particular, the challenger does the following:

  - Sets $\tilde{m}_0 = \begin{cases} m_0 & \text{if } f(x) = 1, \\ 0 & \text{if } f(x) = 0. \end{cases}$ and computes $\mathsf{kpABE.ct}'_{\mathsf{lb}} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; F(K, (\mathsf{lb}, x)))$.

  - Sets γ_1 as $\mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hyb_2. This hybrid is same as the previous hybrid except the following:

- The challenger samples two SKE secret keys $\mathsf{SKE.key}_1$ and $\mathsf{SKE.key}_2$.

- For each key query (lb, x), γ_2 and δ are computed differently as follows:

  - Set $\mathsf{flag} = \begin{cases} 1 & \text{if } f(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$

  - Set $\delta \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{flag})$ and $\gamma_2 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_2, \mathsf{kpABE.ct}'_{\mathsf{lb}})$ if $\mathsf{flag} = 1$, else $\gamma_2 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hyb_3. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hyb_4. This hybrid is same as the previous hybrid except that for each key query (lb, x) $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as follows:

- If $\mathsf{flag} = 1$, $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; r)$, where $r \leftarrow \{0,1\}^t$,
- Else, if $\mathsf{flag} = 0$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_0; F(K, (\mathsf{lb}, x)))$. (This is same as in the previous hybrid).

Hyb_5. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2)).$$

Hyb_6. This hybrid is same as the previous hybrid except that for all the key queries (lb, x), $\gamma_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

Hyb_7. This hybrid is same as the previous hybrid except that for each such key query (lb, x) where $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}} \leftarrow \mathcal{CT}_{\mathsf{kpABE}}$.

Hyb_8. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2)).$$

We note that the hybrids hereafter are unwinding the changes made in the previous hybrids.

Hyb_9. This hybrid is same as the previous hybrid except that for each key query (lb, x) with $\mathsf{flag} = 1$, $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_1; r)$, where $r \leftarrow \{0,1\}^t$.

Hyb_{10}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\gamma_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.key}_1, \mathsf{kpABE.ct}'_{\mathsf{lb}})$.

Hyb_{11}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, \perp, 1, \mathsf{SKE.key}_1, \mathsf{SKE.key}_2)).$$

Hyb_{12}. This hybrid is same as the previous hybrid except that for each key query (lb, x), $\mathsf{kpABE.ct}'_{\mathsf{lb}}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, \tilde{m}_1; F(K, (\mathsf{lb}, x)))$.

Hyb_{13}. This hybrid is same as the previous hybrid except that $\mathsf{FE.ct}$ in the challenge ciphertext is computed as

$$\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{kpABE.mpk}, f, m_1, K, 0, \perp, \perp)).$$

Hyb_{14}. This hybrid is same as the previous hybrid except that the γ_2, δ are now sampled uniformly from $\mathcal{CT}_{\mathsf{SKE}}$ for all the key queries.

Hyb_{15}. This hybrid is same as the previous hybrid except that γ_1 is now sampled uniformly from $\mathcal{CT}_{\mathsf{SKE}}$ for all the key queries. This is the real world where the message m_1 is encrypted in the challenge ciphertext.

Indistinguishability of hybrids: The indistinguishability between the consecutive hybrids is argued in the same way as that in the proof of Theorem 3.10. Therefore, here we give only a brief sketch.

$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ from SKE security. $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ due to FE security and SKE correctness and $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ follows from PRF security. $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$ follows from FE security and SKE correctness and $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$ follows again from the SKE security.

$\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$ follows from selective security of ABE. We observe that in both Hyb_6 and Hyb_7, $\mathsf{ABE.ct}'_{\mathsf{lb}}$ is not used when $\mathsf{flag} = 0$ and when $\mathsf{flag} = 1$, it is sampled from $\mathcal{CT}_{\mathsf{kpABE}}$ directly which can be efficiently done without using $\mathsf{kpABE.mpk}$. This lets the reduction go through. The steps of reduction are same as in the proof of Claim 3.17. $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$ follows again from the security of FE and SKE correctness. In particular, we observe that here the FE challenge messages are $\mu_0 = (\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2)$ and $\mu_1 = (\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2)$. For every $\mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta]$ function (corresponding to RPE key query (lb, x)) for which FE key is generated, we have the following:

- When $\mathsf{flag} = 0$, this implies $f(x) = 0$, which in turn implies that $\tilde{m}_0 = \tilde{m}_1 = 0$. Hence,

$$\begin{aligned}
& \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mathsf{lb}, 0; F(K, (\mathsf{lb}, x))) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1).
\end{aligned}$$

- When $\mathsf{flag} = 1$,

$$\begin{aligned}
& \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_0) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_0, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{SKE.Dec}(\mathsf{SKE.key}_2, \gamma_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mathsf{kpABE.mpk}, f, m_1, K, 2, \perp, \mathsf{SKE.key}_2) \\
= \; & \mathsf{Re\text{-}Enc}[\mathsf{lb}, x, \gamma_1, \gamma_2, \delta](\mu_1).
\end{aligned}$$

This satisfies the admissibility condition for FE security. The rest of the hybrids undo the changes made so far to get to the real world with $\beta = 1$ and the arguments for indistinguishability are same as their counterparts in the first set of hybrids. ■

3.5.3 Alternate Construction using LOT

Here, we consider an alternative construction using LOT. Compared to our construction in Sec. 3.5.1, the construction here can only handle the case where the number of users is

polynomially bounded and only achieves selective security. On the other hand, it can be based on FE and LOT rather than FE and kpABE with specific properties. Note that LOT can be based on more diverse assumptions than kpABE and this leads to an instantiation without LWE in particular.

The construction is similar to that in Sec. 3.5.1 except that we use LOT in place of ABE, which brings in the following changes in the KeyGen, Enc, and Dec algorithms:

- We use $\mathsf{LOT} = (\mathsf{LOT.crsGen}, \mathsf{LOT.Hash}, \mathsf{LOT.Send}, \mathsf{LOT.Receive})$ instead of kpABE.
- The function in Figure 3.2, for which FE key is generated now takes as input LOT objects crs and digest, instead of kpABE.mpk and computes $\mathsf{LOT.ct}_{\mathsf{lb}} = \mathsf{LOT.Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{lb}, 0, \tilde{m}; F(K, (\mathsf{lb}, x)))$, instead of $\mathsf{kpABE.ct}_{\mathsf{lb}}$.
- The encryption algorithm changes as follows:
  $\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f, m, L) \rightarrow \mathsf{RPE.ct}$. The encryption algorithm does the following:
  - Parse $\mathsf{RPE.mpk} = \mathsf{FE.mpk}$ and sample a F key $K \leftarrow \{0,1\}^\lambda$.
  - Generate $\mathsf{crs} \leftarrow \mathsf{LOT.crsGen}(1^\lambda)$.
  - Compute $(\mathsf{digest}, \hat{D}) \leftarrow \mathsf{LOT.Hash}(\mathsf{crs}, D)$, where D is a binary vector of length N (the number of users) and is 1 at positions corresponding to non-revoked labels, i.e. $D[\mathsf{lb}'] = 1$ iff $\mathsf{lb}' \notin L$.
  - Compute $\mathsf{FE.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{crs}, \mathsf{digest}, f, m, K, 0, \perp, \perp))$.
  - Output $\mathsf{RPE.ct} = (\mathsf{crs}, \mathsf{FE.ct})$.
- The algorithm for decryption also changes accordingly as follows:
  $\mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},x}, \mathsf{RPE.ct}, L) \rightarrow m'$. The decryption algorithm does the following:
  - Parse $\mathsf{RPE.ct} = (\mathsf{crs}, \mathsf{FE.ct})$ and $\mathsf{RPE.sk}_{\mathsf{lb},x} = \mathsf{FE.sk}_{\mathsf{lb},x}$.
  - Define D from L as described in the encryption algorithm and compute $(\mathsf{digest}, \hat{D}) \leftarrow \mathsf{LOT.Hash}(\mathsf{crs}, D)$.
  - Compute $\mathsf{LOT.ct}' = \mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct})$.
  - Compute $m' = \mathsf{LOT.Receive}^{\hat{D}}(\mathsf{crs}, \mathsf{LOT.ct}', \mathsf{lb})$.

– Output m'.

We note that the above construction works when the identity space is of polynomial size.

Next we sketch the correctness, efficiency and security of the above construction.

Correctnness. Firstly, for any label lb, attribute x, and function f such that $f(x) = 1$, we have $\mathsf{FE.Dec}(\mathsf{FE.sk}_{\mathsf{lb},x}, \mathsf{FE.ct}) = \mathsf{LOT.ct}_{\mathsf{lb}}$, where $\mathsf{LOT.ct}_{\mathsf{lb}} = \mathsf{LOT.Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{lb}, 0, m; F(K, (\mathsf{lb}, x)))$, by the correctness of FE and the definition of $\mathsf{Re\text{-}Enc}$. Next we observe that if $\mathsf{lb} \notin L$, then $D[\mathsf{lb}] = 1$ and hence from the correctness of LOT scheme it follows that $\mathsf{LOT.Receive}^{\hat{D}}(\mathsf{crs}, \mathsf{LOT.ct}_{\mathsf{lb}}, \mathsf{lb}) = m_{D[\mathsf{lb}]} = m_1 = m$.

So the decryption correctly recovers the message when $f(x) = 1$ and $\mathsf{lb} \notin L$.

Efficiency Here we argue that the above construction using LOT achieves optimal parameters. Namely, we show that the size of each parameter is independent from $|L|$. We note that $|f|$ refers to the description size of the function, not the circuit size that implements the function. When $|x|$ is very long and f has succinct description, the former can be much shorter than the latter.

1. Public key size $|\mathsf{RPE.mpk}|$: We have $|\mathsf{RPE.mpk}| = |\mathsf{FE.mpk}|$. Since we assumed that FE is fully compact (Def. 3.4), the length of $\mathsf{FE.mpk}$ only depends on the input length of $\mathsf{Re\text{-}Enc}$. We have that the input length is $|\mathsf{crs}| + |\mathsf{digest}| + |f| + |m| + |K| + |\mathsf{mode}| + 2|\mathsf{SKE.key}| = |\mathsf{crs}| + |\mathsf{digest}| + |f| + O(\lambda)$. We have $|\mathsf{crs}| + |\mathsf{digest}| = \mathrm{poly}(\lambda)$ by the efficiency of LOT (Theorem 3.8). The total length of the public key is therefore $\mathrm{poly}(\lambda, |f|)$.

2. Secret key size $|\mathsf{RPE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RPE.sk}_{\mathsf{lb},x}| = |\mathsf{FE.sk}_{\mathsf{lb},x}|$. Since the size of the latter is polynomially dependent on the size of $\mathsf{Re\text{-}Enc}$, we evaluate its size. We can see that the size of $\mathsf{Re\text{-}Enc}$ is polynomial in the total length of the input and the hardwired values. The length of the input is bounded by $\mathrm{poly}(\lambda, |f|)$ as analyzed in the above item. The length of the hardwired values are $|\mathsf{lb}| + |x| + |\gamma_1| + |\gamma_2| + |\delta|$. We have $|\gamma_1| + |\gamma_2| + |\delta| = 3|\gamma_2|$[15] and $|\gamma_2| = \mathrm{poly}(\lambda, \mathsf{LOT.ct}_{\mathsf{lb}}) = \mathrm{poly}(\log |D|, \lambda) = \mathrm{poly}(\lambda)$. Therefore, the size of $\mathsf{Re\text{-}Enc}$ is $\mathrm{poly}(\lambda, |f|) + |\mathsf{lb}| + |x|$ and so is the size of the secret key.

[15] We assumed that $|\gamma_1| = |\gamma_2| = |\delta|$. See Footnote 9.

3. Ciphertext size $|\mathsf{RPE.ct}|$: We have $|\mathsf{RPE.ct}| = |\mathsf{crs}| + |\mathsf{FE.ct}|$. We have $|\mathsf{crs}| = \mathrm{poly}(\lambda)$. Also, by similar analysis to the first item, full compactness of FE implies $|\mathsf{FE.ct}| \leq \mathrm{poly}(\lambda, |f|)$. Therefore, the overall length of the ciphertext is $\mathrm{poly}(\lambda, |f|)$.

Security We show that the above construction satisfies the function hiding (Def. 3.17) and the message hiding (Def. 3.14) properties. The key difference here is that we only achieve selective security w.r.t the revoke list L.

Function Hiding The security proof for function hiding will follow the same sequence of hybrids as in the Theorem 3.10 except the following differences :

- In Hyb_0, the challenger generates and uses LOT parameters to compute the challenge ciphertext. Concretely, the challenger computes $\mathsf{FE.ct}$ as $\mathsf{FE.Enc}(\mathsf{FE.mpk}, (\mathsf{crs}, \mathsf{digest}, f_0, m, K, 0, \perp, \perp))$ and returns $\mathsf{RPE.ct} = (\mathsf{crs}, \mathsf{FE.ct})$ to the adversary.

- In Hyb_1 to Hyb_3 , the challenger computes $\mathsf{LOT.ct}'_{\mathsf{lb}} = \mathsf{LOT.Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{lb}, 0, \tilde{m};\ F(K, (\mathsf{lb}, x)))$ instead of $\mathsf{ABE.ct}'_{\mathsf{lb}}$ while answering the key queries.

- Similarly, in Hyb_4 to Hyb_6, for each key query (lb, x), instead of $\mathsf{ABE.ct}'_{\mathsf{lb}}$, $\mathsf{LOT.ct}'_{\mathsf{lb}}$ is computed as follows:

  - If $\mathsf{flag} = 1$, $\mathsf{LOT.ct}'_{\mathsf{lb}} = \mathsf{LOT.Enc}(\mathsf{crs}, \mathsf{digest}, \mathsf{lb}, 0, \tilde{m}; r)$, where $r \leftarrow \{0,1\}^t$.

  - Else, if $\mathsf{flag} = 0$, $\mathsf{LOT.ct}'_{\mathsf{lb}} = \mathsf{LOT.Enc}(\mathsf{crs}, \mathsf{digest}, \mathsf{lb}, 0, \tilde{m}; F(K, (\mathsf{lb}, x)))$.

- In Hyb_7, when $\mathsf{flag} = 1$, $\mathsf{LOT.ct}'_{\mathsf{lb}}$ is simulated using LOTSim, i.e., $\mathsf{LOT.ct}'_{\mathsf{lb}} \leftarrow \mathsf{LOTSim}(D, \mathsf{lb}, 0)$. We observe that when $\mathsf{flag} = 1$, $\mathsf{lb} \in L$ (by admissibility), so $D[\mathsf{lb}] = 0$ and we have $m_{D[\mathsf{lb}]} = m_0 = 0$. Hence, the indistinguishabilty between Hyb_6 and Hyb_7 follows from the sender privacy of LOT (Def. 3.10).

We note that after Hyb_8, we rewind the changes made in the preceding hybrids accordingly. The reason why we only achieve selective security w.r.t the revoke list L is that while answering the key queries, computation of $\mathsf{LOT.ct}'_{\mathsf{lb}}$ uses digest. This digest is computed using the database D, which in turn is derived using the revoke list L.

Message Hiding The proof for message hiding is given via the same sequence of hybrids as in the Theorem 3.22. The differences in the hybrids are similar to the case of function hiding as highlighted in the above paragraph, where we use LOT parameters to compute the challenge ciphertext and $\mathsf{LOT.ct}'_{\mathsf{lb}}$ instead of $\mathsf{ABE.ct}'_{\mathsf{lb}}$ while answering the key queries.

3.6 REVOCABLE MIXED FUNCTIONAL ENCRYPTION

3.6.1 Definition

A revocable mixed functional encryption (RMFE) scheme with input domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in[\mathbb{N}]}$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in[\mathbb{N}]}$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda\in[\mathbb{N}]}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x) \to \mathsf{sk}_{\mathsf{lb},x}$. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}_\lambda$ and an input $x \in \mathcal{X}_\lambda$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},x}$.

$\mathsf{PK\text{-}Enc}(\mathsf{mpk}, L) \to \mathsf{ct}$. The public key encryption algorithm takes as input the master public key mpk and a revocation list $L \subseteq \mathcal{L}_\lambda$ and outputs a ciphertext ct.

$\mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L) \to \mathsf{ct}$. The secret key encryption algorithm takes as input the master secret key msk, a function $f \in \mathcal{F}_\lambda$ and a revocation list $L \subseteq \mathcal{L}_\lambda$, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm takes the secret key $\mathsf{sk}_{\mathsf{lb},x}$, a revocation list $L \subseteq \mathcal{L}_\lambda$ and a ciphertext ct and outputs a bit.

Definition 3.19 (Correctness)**.** A RMFE scheme is said to be correct if there exists

negligible functions $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) = 1 : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x); \\ \mathsf{ct} \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}, L) \end{array} \right] \geq 1 - \mathsf{negl}_1(\lambda).$$

$$\mathsf{lb} \notin L \Rightarrow \Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},x}, L, \mathsf{ct}) = f(x) : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{sk}_{\mathsf{lb},x} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, x); \\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L) \end{array} \right] \geq 1 - \mathsf{negl}_2(\lambda).$$

Security. Here we define the security requirements of RMFE scheme.

Definition 3.20 (q-query Mode Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RMFE scheme satisfies q-query mode hiding security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\beta^{\mathsf{Pr}ime} = \beta : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ f, L \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_0 \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f, L); \\ \mathsf{ct}_1 \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}, L); \\ \beta^{\mathsf{Pr}ime} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the $\mathsf{SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle and is admissible only if for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, $f(x) = 1$.

Definition 3.21 (q-query Selective Function Hiding)**.** Let $q(\cdot)$ be any fixed polynomial. A RMFE scheme satisfies q-query selective function hiding security if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\beta^{\mathsf{Pr}ime} = \beta : \begin{array}{l} L \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda); \\ (f_0, f_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f_\beta, L); \\ \beta^{\mathsf{Pr}ime} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_\beta) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{A}$ can make at most $q(\lambda)$ queries to the $\mathsf{SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle and for all the key queries (lb, x) to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

Remark 5. We note that when the function space $\mathcal{F}_\lambda$ is polynomially small and q is a constant, a variant of Definition 3.21 where the adversary outputs the challenge functions (f_0, f_1) and the $\mathsf{SK\text{-}Enc}$ query functions $\{\bar{f_i}\}_{i\in[q]}$ at the beginning of the game, before the $\mathsf{Setup}(1^\lambda)$ algorithm is run, is equivalent to Definition 3.21 where the adversary adaptively outputs the challenge functions (f_0, f_1) and can make $\mathsf{SK\text{-}Enc}$ queries adaptively, with polynomial loss. Similar comment also applies to Definition 3.20. We will use these simplifications in the security proofs.

3.6.2 Construction

In this section we give a construction of 1-query secure RMFE scheme, with input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$ and a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$. We assume that the size of $|\mathcal{F}_\lambda|$ is bounded by some polynomial in λ, which will suffice for our purpose.

Our scheme uses the following building blocks:

1. A 2-bounded semi-adaptive simulation based function-message private (Definition 3.6) SKFE scheme $\mathsf{SKFE} = (\mathsf{SKFE.Setup}, \mathsf{SKFE.KeyGen}, \mathsf{SKFE.Enc}, \mathsf{SKFE.Dec})$ that supports the function class $\mathcal{F}$. This can be instantiated from one-way functions (Lemma 3.2).

2. A key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{kpABE.Setup}, \mathsf{kpABE.Enc}, \mathsf{kpABE.KeyGen}, \mathsf{kpABE.Dec})$ for the circuit class $C_{\ell(\lambda),d(\lambda)}$ with message space $\{0,1\}^\lambda$ satisfying $\mathsf{Sel\text{-}IND}$ security (Definition 2.5) and efficiency properties described in Theorem 3.3 . We set $\ell(\lambda) = \ell_{\mathsf{lb}} + \log(\lambda) + 1$ and $d(\lambda) = \omega(\log \lambda)$, where ℓ_{lb} is the label length.[16] This can be instantiated from the LWE assumption (Theorem 3.3).

3. A lockable obfuscation scheme $\mathsf{LO} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$ with lock space $\{0,1\}^\lambda$ that supports circuits of the form CC defined in Fig. 3.3. As we will analyze later, the circuit is of fixed polynomial size in λ and $|f|$, where $|f|$ is the description size of the function $f \in \mathcal{F}$. This can be instantiated from the LWE assumption (Theorem 3.7).

Below we describe our construction of a 1-query secure RMFE scheme $\mathsf{RMFE} =$

[16] Concretely, we can choose $d(\lambda) = \Theta(\log \lambda \log\log \lambda)$ for example.

$(\mathsf{RMFE.Setup}, \mathsf{RMFE.KeyGen}, \mathsf{RMFE.PK\text{-}Enc}, \mathsf{RMFE.SK\text{-}Enc}, \mathsf{RMFE.Dec})$.

$\mathsf{RMFE.Setup}(1^\lambda) \to (\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$. The setup algorithm does the following:

- Generate $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$.
- Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$.
- Output $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.

$\mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}, x) \to \mathsf{RMFE.sk}_{\mathsf{lb},x}$. The key generation algorithm does the following:

- Parse $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
- For all $j \in [\lambda], b \in \{0,1\}$, sample $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda$.
  Denote $K = \{K_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ and $R = \{R_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.
- Compute
$$\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (x, K, R)).$$
- For all $j \in [\lambda], b \in \{0,1\}$, compute
$$\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b}).$$
- Output $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$.

$\mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L) \to \mathsf{RMFE.ct}$. The public key encryption algorithm does the following:

- Computes a simulated code $\mathsf{RMFE.ct} \leftarrow \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$[17].
- It outputs $\mathsf{RMFE.ct}$ as the ciphertext.

[17] Here, CC represents the maximum possible size of $\mathsf{CC}[\cdot,\cdot]$ circuit defined in Figure 3.3.

$\mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L) \to \mathsf{RMFE.ct}$. The secret key encryption algorithm does the following:

- Parse $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$, and sample a tag $\mathbf{z} \leftarrow \{0,1\}^\lambda$ and a lock value $\alpha \leftarrow \{0,1\}^\lambda$.
- For all $j \in [\lambda]$, compute $\mathsf{kpABE.sk}_{L,j,z_j} \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C_{L,j,z_j})$, where the function C_{L,j,z_j} has L, j and z_j hardwired and is defined as follows :
  On input $(\mathsf{lb}, i, b) \in \mathcal{L}_\lambda \times [\lambda] \times \{0,1\}$,

$$C_{L,j,z_j}(\mathsf{lb}, i, b) = \begin{cases} 1 & \text{if } (\mathsf{lb} \notin L) \wedge (i = j) \wedge (b = z_j) \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

- Compute $\mathsf{SKFE.sk} \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f,\mathbf{z},\alpha})$, where the function $P_{f,\mathbf{z},\alpha}$ has $f, \mathbf{z}, \alpha$ hardwired and is defined as follows :
  On input $x \in \mathcal{X}_\lambda$, $K = \{K_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$, $R = \{R_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$,

$$P_{f,\mathbf{z},\alpha}(x, K, R) = \begin{cases} \bigoplus_j K_{j,z_j} \oplus \alpha & \text{if } f(x) = 0 \\ \bigoplus_j R_{j,z_j} & \text{if } f(x) = 1. \end{cases} \tag{3.3}$$

- Construct function $\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}]$, with $\mathsf{SKFE.sk}$ and $\{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}$ hardwired and is defined as in Figure 3.3.
- Output $\mathsf{RMFE.ct} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}], \alpha)$.

$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) \to \{0,1\}$. The decryption algorithm does the following:

- Parse $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$ and $\mathsf{RMFE.ct} = \widetilde{\mathsf{CC}}$, where $\widetilde{\mathsf{CC}}$ is regarded as an obfuscated circuit of LO.
- For all $j \in [\lambda], b \in \{0,1\}$, compute

$$\mathsf{kpABE.off}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Dec}^{\mathsf{off}}(\mathsf{kpABE.mpk}, C_{L,j,b}, (\mathsf{lb}, j, b)).$$

- Compute

$$y = \mathsf{LO.Eval}(\widetilde{\mathsf{CC}}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})).$$

Function $\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}]$

Hardwired values: A SKFE secret key $\mathsf{SKFE.sk}$ and kpABE keys $\{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}$.

Inputs: A SKFE ciphertext $\mathsf{SKFE.ct}$ and kpABE ciphertexts $\{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$.

Output : A binary string $\alpha^* \in \{0,1\}^\lambda$.

1. For all $j \in [\lambda]$, compute $m_j = \mathsf{kpABE.Dec}^{\mathsf{on}}(\mathsf{kpABE.sk}_{L,j,z_j}, \mathsf{kpABE.ct}_{\mathsf{lb},j,z_j}, \mathsf{kpABE.off}_{\mathsf{lb},j,z_j})$.
   Let $M_0 = \bigoplus_j m_j$
2. Compute $M_1 = \mathsf{SKFE.Dec}(\mathsf{SKFE.sk}, \mathsf{SKFE.ct})$.
3. Output $M_1 \oplus M_0$.

Figure 3.3: Compute and Compare function CC

- Output 1 if $y = \perp$, else output 0.

Remark 6. We note that by performing the part of the ABE decryption that uses $C_{L,j,b}$, outside of CC , we do not need to provide $C_{L,j,b}$ (or L) as input to CC. Instead, we provide $\{\mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ whose size is independent of the size of $C_{L,j,b}$ (and thus that of L). This helps us in getting succinct ciphertext.

Correctness. We prove the correctness via the following theorem.

Theorem 3.23. *Suppose* kpABE, LO *and* SKFE *are correct and* LO *is secure, then the above construction of* RMFE *satisfies correctness as defined in Def. 3.19.*

Proof. We consider the following two cases:

1. **Public Encryption Correctness.**
   For $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$, we have that $\mathsf{RMFE.ct} = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$. Firstly, we note that from LO security, RMFE.ct is indistinguishable from $\mathsf{RMFE.ct}'$ computed as $\mathsf{LO.Obf}(C, \alpha)$, for any circuit C of the same size as that used by the simulator and has output of length λ. Now, since $\alpha \leftarrow \{0,1\}^\lambda$ has high entropy, for all but negligible inputs w, $C(w) \neq \alpha$. So, from the correctness of LO, it follows that with all but negligible probability

$$\mathsf{LO.Eval}(\mathsf{RMFE.ct}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})) = \perp.$$

Hence the $\mathsf{RMFE.Dec}$ algorithm outputs 1 with all but negligible probability.

2. **Secret Encryption Correctness.**
For $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$, we have $\mathsf{RMFE.ct} = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}], \alpha)$. Now consider the following steps of $\mathsf{LO.Eval}(\mathsf{RMFE.ct}, (\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}, \mathsf{kpABE.off}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}))$

   - Since $\mathsf{lb} \notin L$, by correctness of kpABE, for all $j \in [\lambda]$, $M_0 = \oplus_j m_j = \oplus_j K_{j,z_j}$ with all but negligible probability.
   - By correctness of SKFE, if $f(x) = 0$, we have $M_1 = \bigoplus_j K_{j,z_j} \oplus \alpha$, else $M_1 = \bigoplus_j R_{j,z_j}$.
   - We have $M_0 \oplus M_1 = \alpha$ if $f(x) = 0$.

   So, by the correctness of LO, $\mathsf{LO.Eval}$ outputs 1 if $f(x) = 0$, $\perp$ otherwise with all but negligible probability. Hence the $\mathsf{RMFE.Dec}$ algorithm, by construction, outputs 0 if $f(x) = 0$ and 1 if $f(x) = 1$ with all but negligible probability.

This proves the correctness of the above construction. ■

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that the sizes of the parameters are independent of $|L|$. We first observe that $C_{L,j,b}$ as defined in Eq (3.2) can be implemented with depth $d = \omega(\log \lambda)$, since the membership check $\mathsf{lb} \stackrel{?}{\in} L$ can be done with depth $\log(|\mathsf{lb}| \cdot |L|) = \log(\mathrm{poly}(\lambda))$ and the equality check can be done with depth $\log(|j|) = \log\log\lambda$. We then bound the size of parameters.

1. Public key size $|\mathsf{RMFE.mpk}|$: By the efficiency property of kpABE (Theorem 3.3), we have $|\mathsf{RMFE.mpk}| = |\mathsf{kpABE.mpk}| = \mathrm{poly}(\lambda, d, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{lb}|)$.

2. Secret key size $|\mathsf{RMFE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RMFE.sk}_{\mathsf{lb},x}| = |\mathsf{SKFE.ct}| + |\mathsf{kpABE.mpk}| + 2 \cdot \lambda(|(\mathsf{lb}, j, b)| + |\mathsf{kpABE.ct}|)$. The first term can be bounded by $\mathrm{poly}(\lambda, |f|, |x|)$, the second is $\mathrm{poly}(\lambda, |\mathsf{lb}|)$, and the last terms is $\mathrm{poly}(\lambda, |\mathsf{lb}|)$. Therefore, the total size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

3. Ciphertext size $|\mathsf{RMFE.ct}|$: We first bound the size of the circuit CC defined in Fig. 3.3. The dominant operations in the circuit is the decryption of SKFE and the online decryption of kpABE. We can see that the former is implemented by a circuit of size $\mathrm{poly}(\lambda, |x|, |f|)$ by the efficiency of SKFE. The latter can be implemented by a circuit of size $\mathrm{poly}(\lambda, \ell, d) = \mathrm{poly}(\lambda, \mathsf{lb})$ by the online efficiency

of kpABE (Theorem 3.3). Therefore, the total size CC is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$. By the efficiency of LO, the size of the ciphertext is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as well.

4. Depth of the circuit implementing RMFE.Dec: We also evaluate the depth of the circuit implementing RMFE.Dec and show that it is independent from $|L|$, since it will be used later in Sec. 3.7. We first observe that the dominant operations in RMFE.Dec are the offline decryption of kpABE and the evaluation of CC. The depth of the former can be bounded by $\mathrm{poly}(\lambda, \ell, \mathsf{depth}(C_{L,j,b})) \leq \mathrm{poly}(\lambda, |\mathsf{lb}|)$ by Theorem 3.3. The depth of the latter can be bounded by its size and thus is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as we have seen in the previous item. The total depth is thus bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$, which is independent from $|L|$.

3.6.3 Security

In this section we show that our construction of RMFE scheme satisfies all the security requirements.

We will use the following notations in the security proof:

For $X \in \{K, R\}$,

- $X := \{X_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$, $X_b := \{X_{j,b}\}_{j\in[\lambda]}$, $X_j := \{X_{j,b}\}_{b\in\{0,1\}}$.
- For any vector $\mathbf{z} \in \{0,1\}^\lambda$, $X_{\mathbf{z}} := \{X_{j,z_j}\}_{j\in[\lambda]}$.

Mode hiding

Theorem 3.24. *Assume that* SKFE *and* LO *are secure as per definitions 3.6 and 3.8, respectively. Furthermore, assume* $|\mathcal{F}_\lambda| \leq \mathrm{poly}(\lambda)$. *Then the* RMFE *construction satisfies* 1*-query mode hiding security as per Definition 3.20.*

Proof. Recall that for mode hiding, we need

$$\mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f^*, L^*) \approx_c \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L^*),$$

where for all key queries (lb, x), $f^*(x) = 1$.

The proof proceeds via the following sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hyb_0 : This is the real world with $\beta = 0$, where the challenge ciphertext for (f^*, L^*) is computed using the RMFE.SK-Enc algorithm. We write the complete game here to set up the notations and easy reference in later hybrids.

1. The adversary outputs the challenge function f^* and the SK-Enc query function $\bar{f}$[18].

2. The challenger generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$. It sends RMFE.mpk to $\mathcal{A}$.

3. **Key Queries:** For each key query (lb, x), the challenger computes SKFE.ct and $\mathsf{kpABE.ct}_{\mathsf{lb},j,b}$ as in the construction and returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$ to $\mathcal{A}$.

4. **Challenge Query:** When the adversary outputs L^* for the challenge query, the challenger does the following:

   - Samples a tag $\mathbf{z}^* \leftarrow \{0,1\}^\lambda$ and a lock value $\alpha^* \leftarrow \{0,1\}^\lambda$.
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.sk}^*$ as in the construction.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 3.3 and returns $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to the adversary $\mathcal{A}$.

5. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, the challenger does the following:

   - Samples a tag $\bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$ and a lock value $\bar{\alpha} \leftarrow \{0,1\}^\lambda$.
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.}\bar{\mathsf{sk}}$ as in the construction.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to the adversary $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'.

[18] To keep the proofs and notations simple, we let f^* and $\bar{f}$ to be given selectively. This is sufficient to achieve security as in Def 3.20, as mentioned in Remark 5.

Hyb_1 : This hybrid is same as the previous hybrid except the following changes:

1. The challenger samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ in the beginning of the game after the adversary outputs $f^*, \bar{f}$.

2. The challenger then computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{\mathrm{poly}(|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|)})$ in this order using SKFE simulators.

3. The key generation, challenge and $\mathsf{SK\text{-}Enc}$ queries are answered as follows:

   - For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$. Note that $P_{f^*,\mathbf{z}^*,\alpha^*}(x, K, R) = \bigoplus_j R_{j,z_j^*}$, due to admissibility requirement.

   - To answer the challenge and the $\mathsf{SK\text{-}Enc}$ queries, $\mathsf{SKFE.sk}^*$ and $\mathsf{SKFE.}\bar{\mathsf{sk}}$ generated by SKFE simulators in Step 2 are used for generating $\mathsf{RMFE.ct}^*$ and $\mathsf{RMFE.}\overline{\mathsf{ct}}$, respectively.

Hyb_2 : This hybrid is same as the previous hybrid except that the challenger uses $\mathsf{LO.Sim}$ to generate the challenge ciphertext.

$$\mathsf{RMFE.ct}^* = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})^{19}.$$

Hyb_3 : This hybrid is same as the previous hybrid except that the challenger uses $\mathsf{SKFE.Enc}$ and $\mathsf{SKFE.KeyGen}$ to generate SKFE ciphertexts and keys respectively. Formally,

$$\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}})$$

For each key query (lb, x),

$$\mathsf{SKFE.ct} = \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (K, R, x)),$$

where vectors K and R are freshly sampled for each key as defined in the

[19] Here, $|\mathsf{CC}|$ represents the maximum size of the circuit $\mathsf{CC}[\cdot,\cdot]$ defined in Figure 3.3.

construction. This is the real world with $\beta = 1$, where the challenge ciphertext is computed using the $\mathsf{RMFE.PK\text{-}Enc}$ algorithm.

Indistinguishability of hybrids

Claim 3.25. Assume that SKFE is secure (Def. 3.6), then $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ who can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of SKFE scheme with the same advantage ϵ. The reduction is as follows.

1. Upon being invoked by the SKFE challenger, $\mathcal{B}$ invokes $\mathcal{A}$ which outputs $f^*, \bar{f}$.

2. $\mathcal{B}$ samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$.

3. $\mathcal{B}$ defines the functions $P_{f^*,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$ as defined in Eq. (3.3) and sends it to the SKFE challenger in this order as key queries. The challenger generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and samples $\hat{\beta} \leftarrow \{0,1\}$.
   If $\hat{\beta} = 0$, it computes $\mathsf{SKFE.sk}^* = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f^*,\mathbf{z}^*,\alpha^*})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}})$
   else it computes $\mathsf{SKFE.sk}^* = \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$, $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$ and sends $\{\mathsf{SKFE.sk}^*, \mathsf{SKFE.}\bar{\mathsf{sk}}\}$ to $\mathcal{B}$.

4. $\mathcal{B}$ generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and sends $\mathsf{RMFE.mpk}$ to $\mathcal{A}$.

5. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:

   - Samples $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda, \forall j \in [\lambda], b \in \{0,1\}$.
   - Computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.msk}, (\mathsf{lb}, j, b), K_{j,b})$ for $j \in [\lambda], b \in \{0,1\}$.
   - It sends (x, K, R) as challenge message to the SKFE challenger. The challenger returns $\mathsf{SKFE.ct}_{\hat{\beta}}$, where $\mathsf{SKFE.ct}_0 = \mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, (x, K, R))$ and $\mathsf{SKFE.ct}_1 = \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
   - Returns

$\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}_{\hat{\beta}}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.

6. **Challenge Query:** When $\mathcal{A}$ outputs L^* for the challenge query, $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 3.3 and returns $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to $\mathcal{A}$.
7. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to $\mathcal{A}$.
8. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the SKFE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_0, else Hyb_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_0) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1)|$ $= \epsilon$ (by assumption). ■

Claim 3.26. Assume that LO is secure (Def. 3.8), then $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. We show that if there exists a PPT adversary $\mathcal{A}$ who can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of LO scheme with the same advantage ϵ. The reduction is as follows

1. Upon being invoked by the LO challenger, $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs $f^*, \bar{f}$.
2. $\mathcal{B}$ samples $\bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$. It also defines the function $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$.
3. $\mathcal{B}$ generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$ and sends RMFE.mpk to $\mathcal{A}$.
4. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:

- Samples $K_{j,b}, R_{j,b} \leftarrow \{0,1\}^\lambda, \forall j \in [\lambda], b \in \{0,1\}$.
- Computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b})$ $\forall j \in [\lambda], b \in \{0,1\}$ and $\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, \bigoplus_j R_{j,z_j^*}, P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
- Returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.

5. **Challenge Query:** When the adversary outputs L^* for the challenge ciphertext , $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$.
   - Constructs a function $\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}]$ as defined in Figure 3.3, where $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f^*,\mathbf{z}^*,\alpha^*}|})$[20].
   - It sends $\mathsf{CC}[\mathsf{SKFE.sk}^*, \mathsf{kpABE.sk}_{L^*,j,z_j^*}]$ to the LO challenger. The challenger samples a lock value $\alpha^* \leftarrow \{0,1\}^\lambda$ and $\hat{\beta} \leftarrow \{0,1\}$, computes and return $\mathsf{Obf}_{\hat{\beta}}$, where $\mathsf{Obf}_0 = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ and $\mathsf{Obf}_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|\mathsf{CC}|})$.
   - It sets $\mathsf{RMFE.ct}^* = \mathsf{Obf}_{\hat{\beta}}$ and sends it to the adversary $\mathcal{A}$.
6. SK-Enc **Query:** When the adversary outputs $\bar{L}$ for the SK-Enc query, $\mathcal{B}$ does the following:
   - Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$.
   - Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{L,j,z_j}\}_{j\in[\lambda]}]$, where $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$.
   - Computes $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}}\}_{j\in[\lambda]}], \bar{\alpha})$ and returns it to the adversary $\mathcal{A}$.
7. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the LO challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_1, else Hyb_2 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_2)|$ $= \epsilon$ (by assumption). ■

[20]The size of $P_{f^*,\mathbf{z}^*,\alpha^*}$ is independent of any specific lock value and hence can be computed without the knowledge of α^*.

Claim 3.27. Assume that SKFE is secure (Def. 3.6), then $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

The proof of this claim is similar to that of Claim 3.25, hence omitted. ■

Function Hiding

Theorem 3.28. *Assume* SKFE *is secure (Def. 3.6),* kpABE *satisfies* Sel-IND *security (Def. 2.5). Furthermore, assume* $|\mathcal{F}_\lambda| \leq \text{poly}(\lambda)$. *Then, the* RMFE *construction satisfies* 1*-query function hiding as defined in Definition 3.21.*

Proof. We prove the theorem via the following sequence of hybrids.

Hyb_0 : This is the real world with $\beta = 0$. We summarize the steps of the game here to set up the notations used in the later hybrids.

1. The adversary outputs the challenge query f_0, f_1, L^* and the SK-Enc query function $\bar{f}$ in the beginning of the game[21]. The challenger then does the following:

   - Generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$, $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$, sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{RMFE.msk} = (\mathsf{SKFE.msk}, \mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
   - It also samples a tag $\mathbf{z}^* \in \{0,1\}^\lambda$ and a lock value α^*.
   - Computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*})$ and $\{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}$ as defined in the construction.
   - Returns $\mathsf{RMFE.mpk}$ and the challenge ciphertext $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$ to $\mathcal{A}$.

2. **Key Queries:** For each key query (lb, x), the challenger returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}})$, where $\mathsf{SKFE.ct}, \{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ are computed as in the construction.

3. SK-Enc **Query :** When the adversary outputs $\bar{L}$ for the SK-Enc query, the challenger does the following:

[21] To keep the proofs and notations simple, we let f_0, f_1 and $\bar{f}$ to be given selectively. This is sufficient to achieve security as in Def 3.20, as mentioned in Remark 5.

- Samples a tag $\bar{\mathbf{z}} \leftarrow \{0,1\}^\lambda$ and a lock value $\bar{\alpha} \leftarrow \{0,1\}^\lambda$.
- Computes kpABE secret keys $\{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}$ and a SKFE secret key $\mathsf{SKFE.}\bar{\mathsf{sk}}$ as in the construction.
- Constructs $\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}]$ and returns $\mathsf{RMFE.}\bar{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$ to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a bit β'.

Hyb_1 : This hybrid is same as the previous hybrid except the following:

1. The challenger samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ and defines the functions $P_{f_0,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$ in the beginning of the game.
2. The SKFE ciphertexts and keys are computed using SKFE simulators as:
   - For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x,K,R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x,K,R))$.
   - The SKFE secret keys $\mathsf{SKFE.sk}^*$ and $\mathsf{SKFE.}\bar{\mathsf{sk}}$ in $\mathsf{RMFE.ct}^*$ and $\mathsf{RMFE.}\overline{\mathsf{ct}}$ are computed as $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$ and $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$, respectively.

Hyb_2 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, for all $j \in [\lambda]$, $\mathsf{kpABE.ct}_{\mathsf{lb},j,1-\bar{z}_j}$ in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1-\bar{z}_j), 0^\lambda)$.

Hyb_3 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, K-values are chosen differently as follows: let $i \in [\lambda]$ be the first position where $z_i^* \neq \bar{z}_i$. If no such i exists, then the challenger aborts the game. Else, it samples $\{K_{j,b}\}_{j\in[\lambda]\setminus\{i\},b\in\{0,1\}}$, $K_{i,\bar{z}_i}$, $\{R_{j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ uniformly randomly as in the original game. It then sets $K_{i,1-\bar{z}_i} = \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus \bigoplus_{j\in[\lambda]} R_{j,z_j^*}$.

Hyb_4 : This hybrid is same as the previous hybrid, except that the SKFE ciphertexts

and keys in RMFE secret keys and ciphertexts are now generated with function f_1 in place of f_0 as follows:

- For each key query (lb, x), the SKFE ciphertext in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*}(x,K,R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x,K,R))$.
- The SKFE secret key $\mathsf{SKFE.sk}^*$ in $\mathsf{RMFE.ct}^*$ is computed as $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_1,\mathbf{z}^*,\alpha^*}|})$.

Now, from here we undo the changes made in the previous hybrids.

Hyb_5 : In this hybrid, for each key query (lb, x) with $\mathsf{lb} \in L^*$, $K_{i,1-\bar{z}_i}$ (where i is the first position where $\mathbf{z}^*$ and $\bar{\mathbf{z}}$ differ) is also sampled randomly.

Hyb_6 : In this hybrid, for each such key query (lb, x) with $\mathsf{lb} \in L^*$, $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1-\bar{z}_j), 0^\lambda)$ is changed back to $\mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, 1-\bar{z}_j), K_{j,1-\bar{z}_j})$ in $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.

Hyb_7 : In this hybrid SKFE ciphertexts in RMFE secret keys and SKFE secret keys in RMFE ciphertexts are computed as in the real world. That is,

- For each key query (lb, x), $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.Enc}(\mathsf{SKFE.msk}, K, R), \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), K_{j,b})\}_{j\in[\lambda],b\in\{0,1\}})$.
- $\mathsf{RMFE.ct}^* = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,z_j^*}\}_{j\in[\lambda]}], \alpha^*)$, $\mathsf{RMFE.}\overline{\mathsf{ct}} = \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{z}_j}\}_{j\in[\lambda]}], \bar{\alpha})$, where $\mathsf{SKFE.sk}^* = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*})$ and $\mathsf{SKFE.}\bar{\mathsf{sk}} = \mathsf{SKFE.KeyGen}(\mathsf{SKFE.msk}, P_{\bar{f},\mathbf{z}^*,\alpha^*})$

Note that this hybrid corresponds to the real world for $\beta = 1$.

Indistinguishability of the hybrids. Now we show that the consecutive hybrids are computationally indistinguishable for any PPT adversary $\mathcal{A}$.

Claim 3.29. Assume that SKFE is secure (Def. 3.6). Then, $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. The proof follows similar steps as the proof for Claim 3.25, hence omitted. ■

Claim 3.30. Assume that kpABE is selectively secure. Then, $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. To prove the claim we consider the following sub hybrids between Hyb_1 and Hyb_2. Let $L^* = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L^*|}\}$ with some fixed ordering between the labels in L^*. Let $L_{[1:k]} \subseteq L^*$ denote the set of first k labels in L^*, i.e. $L^*_{[1:k]} = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_k\}$. Then for all $1 \leq i \leq |L^*|$ and $0 \leq \tau \leq \lambda$, define $\mathsf{Hyb}_{1.(i,\tau)}$: which is same as Hyb_1 except that for any key query (lb, x), $\{\mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ is computed differently as follows:

$$\mathsf{kpABE.ct}_{\mathsf{lb},j,b} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}, j, b), W),$$

where

$$W = \begin{cases} K_{i,b} & \text{if } (b = \bar{\mathbf{z}}_j) \vee (\mathsf{lb} \notin L_{[1:i]}) \vee (\mathsf{lb} = \mathsf{lb}_i \wedge j > \tau), \\ 0^\lambda & \text{otherwise, i.e. } (b = 1 - \bar{\mathbf{z}}_j) \wedge (\mathsf{lb} \in L_{[1:i-1]}) \vee (\mathsf{lb} = \mathsf{lb}_i \wedge j \leq \tau). \end{cases}$$

Then, we observe that $\mathsf{Hyb}_{1.(1,0)} = \mathsf{Hyb}_1$, $\mathsf{Hyb}_{1.(|L^*|,\lambda)} = \mathsf{Hyb}_2$ and $\mathsf{Hyb}_{1.(i-1,\lambda)} = \mathsf{Hyb}_{1,(i,0)}$.

Hence, all we need to show is that for all $i \in [|L^*|]$, $\tau \in [\lambda]$,

$$\mathsf{Hyb}_{1.(i,\tau-1)} \approx_c \mathsf{Hyb}_{1.(i,\tau)}.$$

This follows from the $\mathsf{Sel\text{-}IND}$ security of kpABE. In particular, if there is no key query issued for lb_i, then the hybrids are identical. On the other hand, if there is a key query (lb, x), such that $\mathsf{lb} = \mathsf{lb}_i$, then we show that if $\mathcal{A}$ can distinguish between the two hybrids with non-negligible advantage ϵ then we can design a PPT algorithm $\mathcal{B}$ with the same advantage ϵ against $\mathsf{Sel\text{-}IND}$ security of kpABE. $\mathcal{B}$ is defined as follows:

1. Upon being invoked by the kpABE challenger, $\mathcal{B}$ invokes $\mathcal{A}$ which outputs f_0, f_1, L^*, $\bar{f}$. Let $L^* = \{\mathsf{lb}_1, \ldots, \mathsf{lb}_{|L^*|}\}$.

2. $\mathcal{B}$ sends $(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau)$ as challenge attribute to the kpABE challenger. The kpABE challenger samples $\hat{\beta} \leftarrow \{0,1\}$ and $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda)$ and sends $\mathsf{kpABE.mpk}$ to $\mathcal{B}$.

3. $\mathcal{B}$ generates $\mathsf{SKFE.msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and sets $\mathsf{RMFE.mpk} = \mathsf{kpABE.mpk}$. It also samples $\alpha^*, \bar{\alpha}, \mathbf{z}^*, \bar{\mathbf{z}}$ and defines $P_{f_0,\mathbf{z}^*,\alpha^*}$ and $P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}$. It then does the following:

   - For each $j \in [\lambda]$, defines the circuit $C_{L^*,j,\mathbf{z}^*_j}$ and sends a key query for $C_{L^*,j,\mathbf{z}^*_j}$ to the kpABE challenger. The kpABE challenger returns $\mathsf{kpABE.sk}_{L^*,j,\mathbf{z}^*_j}$.
   - Computes $\mathsf{SKFE.sk}^* \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$.
   - Computes $\mathsf{RMFE.ct}^* \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.sk}^*, \{\mathsf{kpABE.sk}_{L^*,j,\mathbf{z}^*_j}\}_{j\in[\lambda]}])$.
   - Returns $\mathsf{RMFE.mpk}$ and $\mathsf{RMFE.ct}^*$ to $\mathcal{A}$.

4. **Key Queries:** For each key query (lb, x), $\mathcal{B}$ does the following:

   - Computes $\mathsf{SKFE.ct} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x, K, R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x, K, R))$.
   - To compute kpABE ciphertext, $\mathsf{kpABE.sk}_{\mathsf{lb},j,b}$ for $j \in [\lambda], b \in \{0,1\}$,
     - if $(\mathsf{lb} = \mathsf{lb}_i) \wedge j = \tau \wedge b = 1 - \bar{\mathbf{z}}_j$[22], then $\mathcal{B}$ sends challenge query with messages $\mu_0 = K_{j,1-\bar{\mathbf{z}}_j}$ and $\mu_1 = 0^\lambda$. The kpABE challenger returns $\mathsf{kpABE.ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, (\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau), \mu_{\hat{\beta}})$, which $\mathcal{B}$ sets as $\mathsf{kpABE.ct}_{\mathsf{lb}_i,\tau,1-\bar{\mathbf{z}}_\tau}$.
     - else, $\mathcal{B}$ computes $\mathsf{kpABE.ct}_{\mathsf{lb},j,b}$ on its own using $\mathsf{kpABE.mpk}$ as defined for $\mathsf{Hyb}_{1.(i,\tau-1)}$ (same for $\mathsf{Hyb}_{1.(i,\tau)}$).
   - Returns $\mathsf{RMFE.sk}_{\mathsf{lb},x} = (\mathsf{SKFE.ct}, \mathsf{kpABE.mpk}, \{(\mathsf{lb}, j, b), \mathsf{kpABE.ct}_{\mathsf{lb},j,b}\}_{j,b})$ to $\mathcal{A}$.

5. $\mathsf{SK\text{-}Enc}$ **Query:** When $\mathcal{A}$ outputs $\bar{L}$ as part of $\mathsf{SK\text{-}Enc}$ query, $\mathcal{B}$ does the following:

   - Computes $\mathsf{SKFE.}\bar{\mathsf{sk}} \leftarrow \mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}|})$.
   - For each $j \in [\lambda]$, defines circuit $C_{\bar{L},j,\bar{\mathbf{z}}_j}$ and sends a kpABE key query for $C_{\bar{L},j,\bar{\mathbf{z}}_j}$. The kpABE challenger returns $\mathsf{kpABE.sk}_{\bar{L},j,\bar{\mathbf{z}}_j}$.
   - Computes $\mathsf{RMFE.}\overline{\mathsf{ct}} \leftarrow \mathsf{LO.Obf}(\mathsf{CC}[\mathsf{SKFE.}\bar{\mathsf{sk}}, \{\mathsf{kpABE.sk}_{\bar{L},j,\bar{\mathbf{z}}_j}\}_{j\in[\lambda]}])$.

[22] If there are more than one key queries with $\mathsf{lb} = \mathsf{lb}_i$, then we use multi-challenge version of $\mathsf{Sel\text{-}IND}$ security of kpABE which can easily be shown equivalent to the one defined in Definition 2.5.

- Returns $\mathsf{RMFE}.\overline{\mathsf{ct}}$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards β' as its guess bit to the kpABE challenger.

We observe that if $\hat{\beta} = 0$, then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1.(i,\tau-1)}$, else $\mathsf{Hyb}_{1.(i,\tau)}$ with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{1.(i,\tau-1)}) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_{1.(i,\tau)})| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$: Firstly, we observe that $\mathcal{B}$ issues key queries for only the following set of circuits: $\{C_{L^*,j,\mathbf{z}_j^*}\}_{j\in[\lambda]}$, $\{C_{\bar{L},j,\bar{\mathbf{z}}_j}\}_{j\in[\lambda]}$ and the challenge attribute is $(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau)$, where $\mathsf{lb}_i \in L^*$. Next, we note that

- $C_{L^*,j,\mathbf{z}_j^*}(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau) = 0$ for all $j \in [\lambda]$, because $\mathsf{lb}_i \in L^*$.
- $C_{\bar{L},j,\bar{\mathbf{z}}_j}(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau) = 0$ for all $j \neq \tau$, and $C_{\bar{L},\tau,\bar{\mathbf{z}}_j}(\mathsf{lb}_i, \tau, 1 - \bar{\mathbf{z}}_\tau) = 0$, because $\bar{\mathbf{z}}_\tau \neq 1 - \bar{\mathbf{z}}_\tau$.

This establishes the admissibility of $\mathcal{B}$. ∎

Claim 3.31. Hyb_2 and Hyb_3 are statistically indistinguishable.

Proof. Firstly, we observe that $\mathsf{Pr}[\mathbf{z}^* = \bar{\mathbf{z}}] = 1/2^\lambda$. Hence, with probability $1 - 1/2^\lambda$, the challenger does not abort in Hyb_3. Next we show that if the game is not aborted, the two hybrids are statistically indistinguishable in the view of the adversary. In this case, the only difference between the two hybrids is the following: for each key query (lb, x) with $\mathsf{lb} \in L^*$, the value of $K_{i,1-\bar{z}_i}$ (i.e. K_{i,z_i^*}), where i is the first index such that $z_i^* \neq \bar{z}_i$, are computed differently. In Hyb_2, K_{i,z_i^*} is sampled uniformly, while in Hyb_3, it is computed as $K_{i,z_i^*} = \bigoplus_{j\in[\lambda]\setminus\{i\}}(K_{j,z_j^*} \oplus R_{j,z_j^*}) \oplus R_{i,z_i^*} \oplus \alpha^*$. However, note that if $f_0(x) = 1$, then K_{i,z_i^*} is not used in anywhere because of the change we introduced in Hyb_2 and hence does not affect the adversary's view. On the other hand, if $f_0(x) = 0$, R_{i,z_i^*} is not used anywhere, and hence K_{i,z_i^*} is uniformly random in the adversary's view because of the randomness of R_{i,z_i^*}. ∎

Claim 3.32. Hyb_3 and Hyb_4 are identical in the view of the adversary.

Proof. Observe that the two hybrids differ only in the computation (simulation) of SKFE ciphertexts and secret keys. In particular,

- $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_0,\mathbf{z}^*,\alpha^*}|})$ is changed to $\mathsf{SKFE.Sim}^{\mathsf{SK}}(\mathsf{SKFE.msk}, 1^{|P_{f_1,\mathbf{z}^*,\alpha^*}|})$ in the computation of $\mathsf{RMFE.ct}^*$. This is just a conceptual change and both the computations are exactly the same, since $|P_{f_0,\mathbf{z}^*,\alpha^*}| = |P_{f_1,\mathbf{z}^*,\alpha^*}|$.[23]

- For each key query (lb, x), $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_0,\mathbf{z}^*,\alpha^*}(x,K,R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x,K,R))$ is changed to $\mathsf{SKFE.Sim}^{\mathsf{CT}}(\mathsf{SKFE.msk}, P_{f_1,\mathbf{z}^*,\alpha^*}(x,K,R), P_{\bar{f},\bar{\mathbf{z}},\bar{\alpha}}(x,K,R))$. The change is again only conceptual and does not affect the actual computation as we argue below:

  - For $f_0(x) = f_1(x)$: there is no change.

  - For $f_0(x) \neq f_1(x)$: let $f_0(x) = 1$ and $f_1(x) = 0$. Then, $\mathsf{lb} \in L^*$ and thus

$$
\begin{aligned}
P_{f_0,\mathbf{z}^*,\alpha^*}(x,K,R) &= \bigoplus_{j\in[\lambda]} R_{j,z_j^*} \\
P_{f_1,\mathbf{z}^*,\alpha^*}(x,K,R) &= \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus K_{i,z_i^*} \\
&= \bigoplus_{j\in[\lambda]\setminus\{i\}} K_{j,z_j^*} \oplus \alpha^* \oplus \Big(\bigoplus_{j\in[\lambda]\setminus\{i\}} (K_{j,z_j^*} \oplus R_{j,z_j^*}) \oplus R_{i,z_i^*}\Big) \oplus \alpha^* \\
&= \bigoplus_{j\in[\lambda]} R_{j,z_j^*}.
\end{aligned}
$$

  - The same argument works for $f_0(x) = 0$ and $f_1(x) = 1$.

■

Indistinguishability between the rest of the hybrids can be argued in the same way as their counterparts in the previous set of hybrids. In particular, proofs for indistinguishability between Hyb_4 and Hyb_5 is same as the proof of claim 3.31, Hyb_5 and Hyb_6 is same as the proof for claim 3.30 and Hyb_6 and Hyb_7 is same as the proof for claim 3.29. ■

[23] we can use padding to make the sizes equal.

3.7 SECRET KEY RPE FROM EVASIVE AND TENSOR LWE

In this section we construct a secret-key RPE scheme from evasive and tensor LWE, followed by the efficiency and security analysis of our scheme.

3.7.1 Construction

We give a construction of the secret-key RPE scheme $\mathsf{RPE} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Broadcast}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ for an attribute space $\mathcal{X} = \{\mathcal{X}_\lambda\}_\lambda$, a function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$, a label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_\lambda$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$. We assume that the size of $|\mathcal{F}_\lambda|$ is bounded by some polynomial in λ, which will suffice for our purpose. Our scheme uses the following building blocks:

1. An RMFE scheme $\mathsf{RMFE} = (\mathsf{RMFE.Setup}, \mathsf{RMFE.KeyGen}, \mathsf{RMFE.PK\text{-}Enc}, \mathsf{RMFE.SK\text{-}Enc}, \mathsf{RMFE.Dec})$ with attribute space $\mathcal{X}$, label space $\mathcal{L}$ and function family $\mathcal{F}$. We instantiate it by our construction in Sec. 3.6.2 based on LWE.

2. A CP-ABE scheme $\mathsf{cpABE} = (\mathsf{cpABE.Setup}, \mathsf{cpABE.Enc}, \mathsf{cpABE.KeyGen}, \mathsf{cpABE.Dec})$ with message space $\mathcal{M}$ satisfying VerSel-IND security (Def. 2.8) that supports the circuit class $\mathcal{C}_{\ell(\lambda),d(\lambda)}$ and the efficiency property listed in Theorem 3.4. We set $\ell(\lambda) = |\mathsf{RMFE.sk}_{\mathsf{lb},x}|$ and $d(\lambda)$ to be the upper bound on the depth of the circuit $C_{L,\mathsf{RMFE.ct}}$ in Eq. (3.4). Looking ahead, we show that the depth is bounded by some polynomial $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$. We choose d so that it is larger than the value. We can instantiate the scheme by the one proposed by [169] based on tensor and evasive LWE.

We describe our construction below.

$\mathsf{RPE.Setup}(1^\lambda) \to (\mathsf{RPE.mpk}, \mathsf{RPE.msk})$. The setup algorithm takes as input the security parameter λ and does the following:

- Generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk}) \leftarrow \mathsf{RMFE.Setup}(1^\lambda)$ and $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$.
- Outputs $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.

$\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x) \to \mathsf{RPE.sk}_{\mathsf{lb},x}$. The key generation algorithm takes as input the master secret key $\mathsf{RPE.msk}$, a label $\mathsf{lb} \in \mathcal{L}$ and an attribute $x \in \mathcal{X}$

and does the following:

- Parse $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.
- Compute $\mathsf{RMFE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}, x)$.
- Set $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$.
- Compute $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
- Output $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$.

$\mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L) \rightarrow \mathsf{RPE.ct}$. The broadcast algorithm takes as input the master public key $\mathsf{RPE.mpk}$, a message m, and a revocation list $L \subseteq \mathcal{L}$ and does the following:

- Parse $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$.
- Compute $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$.
- Construct circuit $C_{L,\mathsf{RMFE.ct}}$, with L and $\mathsf{RMFE.ct}$ hardwired, defined as follows:
  On input (lb, x) and $\mathsf{RMFE.sk}_{\mathsf{lb},x}$,

$$C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}) = (\mathsf{lb} \notin L) \wedge (\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1). \tag{3.4}$$

- Compute $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m)$.
- Output $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.

$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L) \rightarrow \mathsf{RPE.ct}$. The encryption algorithm takes as input the master secret key, a function f, a message m, and a revocation list $L \subseteq \mathcal{L}$ and does the following:

- Parse $\mathsf{RPE.msk} = (\mathsf{RMFE.msk}, \mathsf{cpABE.msk})$.
- Compute $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$.

- Construct circuit $C_{L,\mathsf{RMFE.ct}}$ from L and $\mathsf{RMFE.ct}$ as defined in Eq. (3.4).
- Compute $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m)$.
- Output $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.

$\mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},x}, \mathsf{RPE.ct}, L) \to m$. The decryption algorithm takes as input the secret key $\mathsf{RPE.sk}_{\mathsf{lb},x}$, a ciphertext $\mathsf{RPE.ct}$, and a revocation list L and does the following:

- Parse $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}), \mathsf{cpABE.sk}_{\mathsf{lb},x})$ and $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$.
- Construct circuit $C_{L,\mathsf{RMFE.ct}}$ from L and $\mathsf{RMFE.ct}$ as defined in Eq. (3.4).
- Compute and output $\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}})$.

Correctness. First, we show that $C_{L,\mathsf{RMFE.ct}} \in C_{\ell,d}$. In particular, it suffices to bound the depth of the circuit d by some fixed polynomial $\mathrm{poly}(\lambda)$. We first observe that checking whether $\mathsf{lb} \in L$ or not can be done with depth $|\log(|\mathsf{lb}| \cdot |L|) = \log(\mathrm{poly}(\lambda)) \leq \lambda$. We also have that the depth of $\mathsf{RMFE.Dec}$ is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ as we saw in Sec. 3.6.2. Therefore, the total depth is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

Theorem 3.33. *Suppose* RMFE *and* cpABE *are correct, then the above construction of secret-key* RPE *satisfies correctness (Def. 3.12).*

Proof. For any (lb, x), function $f \in \mathcal{F}_\lambda$ and a revocation list $L \subseteq \mathcal{L}_\lambda$ such that $f(x) = 1, \forall x \in \mathcal{X}_\lambda$ and $\mathsf{lb} \notin L$, consider the following two cases:

1. **Broadcast Correctness:** For any ciphertext $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L)$, we have $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$, where $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$. So, by the public encryption correctness of RMFE scheme, with all but negligible probability

$$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1.$$

Since $\mathsf{lb} \notin L$, we have $C_{L,\mathsf{RMFE.ct}}(\mathsf{att}) = 1$, where $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$. Hence by the correctness of cpABE scheme we have that with all but negligible

probability

$$\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}}) = m.$$

2. **Encryption Correctness:** For any ciphertext $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L)$, we have $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct})$, where $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$. If $(\mathsf{lb} \notin L) \wedge f(x) = 1$, then by the correctness of $\mathsf{RMFE.SK\text{-}Enc}$ algorithm, we have with all but negligible probability
$$\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb},x}, \mathsf{RMFE.ct}, L) = 1.$$
Furthermore, $\mathsf{lb} \notin L$ implies $C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x}) = 1$. Hence by the correctness of cpABE scheme, we have that with all but negligible probability
$$\mathsf{cpABE.Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathsf{lb},x}, \mathsf{att}, \mathsf{cpABE.ct}, C_{L,\mathsf{RMFE.ct}}) = m.$$

Hence the above construction of secret-key RPE satisfies correctness. ■

Efficiency. Here we argue that our construction achieves optimal parameters. Namely, we show that all the parameters are independent from $|L|$.

1. Public key size $|\mathsf{RPE.mpk}|$: We have $|\mathsf{RPE.mpk}| = |\mathsf{cpABE.mpk}| + |\mathsf{RMFE.mpk}|$. The former is bounded by $\mathrm{poly}(\lambda, \ell, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ using Theorem 3.4, where we additionally used $\ell = |\mathsf{RMFE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and $d = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

2. Secret key size $|\mathsf{RPE.sk}_{\mathsf{lb},x}|$: We have $|\mathsf{RPE.sk}_{\mathsf{lb},x}| = |\mathsf{att}| + |\mathsf{cpABE.sk}_{\mathsf{lb},x}|$. We have $|\mathsf{att}| = |\mathsf{lb}| + |x| + |\mathsf{RMFE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and $|\mathsf{cpABE.sk}_{\mathsf{lb},x}| = \mathrm{poly}(\ell, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ by Theorem 3.4. Therefore, the overall size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

3. Ciphertext size $|\mathsf{RPE.ct}|$: We have $|\mathsf{RPE.ct}| = |\mathsf{RMFE.ct}| + |\mathsf{cpABE.ct}|$. The former is bounded by $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ and the latter is bounded by $\mathrm{poly}(\lambda, d) = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$ by Theorem 3.4. Therefore, the overall size is $\mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|)$.

3.7.2 Security

In this section we show that our construction of secret-key RPE is secure.

Message Hiding Security

Theorem 3.34. *Assume that* cpABE *is very selectively secure (Def. 2.8) ,* RMFE *is correct and* $|\mathcal{F}| \leq \text{poly}(\lambda)$. *Then* RPE *scheme satisfies* 1*-query very selective message hiding security (Def. 3.15).*

Proof. Recall that in the message hiding security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_1, L),$$

where for all the key queries (lb, x) to the $\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \cdot, \cdot)$ oracle, either $f(x) = 0$ or $\mathsf{lb} \in L$. We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the selective message hiding security game, then there exists a PPT adversary $\mathcal{B}$ against the security of cpABE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ first runs $\mathcal{A}$. $\mathcal{A}$ outputs the key queries $\{(\mathsf{lb}_1, x_1), (\mathsf{lb}_2, x_2), \ldots, (\mathsf{lb}_Q, x_Q)\}$, f and L[24].

2. $\mathcal{B}$ generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk}) \leftarrow \mathsf{RMFE.Setup}(1^\lambda)$ and computes $\mathsf{RMFE.ct} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$.

3. $\mathcal{B}$ computes $\mathsf{RMFE.sk}_{\mathsf{lb}_i, x_i} \leftarrow \mathsf{RMFE.KeyGen}(\mathsf{RMFE.msk}, \mathsf{lb}_i, x_i)$ for all $i \in [Q]$. It also constructs the circuit $C_{L,\mathsf{RMFE.ct}}$ as defined in the construction and sends $(C_{L,\mathsf{RMFE.ct}}, \{(\mathsf{lb}_i, x_i), \mathsf{RMFE.sk}_{\mathsf{lb}_i, x_i}\}_{i \in [Q]})$ as the challenge function and key attributes to the cpABE challenger.
   The cpABE challenger generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$ and keys $\mathsf{cpABE.sk}_{\mathsf{lb}_i, x_i} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, (\mathsf{lb}_i, x_i), \mathsf{RMFE.sk}_{\mathsf{lb}_i, x_i})$ and returns $(\mathsf{cpABE.mpk}, \{\mathsf{cpABE.sk}_{\mathsf{lb}_i, x_i}\}_{i \in [Q]}$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends $(\mathsf{RPE.mpk}, \{\mathsf{cpABE.sk}_{\mathsf{lb}_i, x_i}\}_{i \in [Q]})$ to $\mathcal{A}$.

4. **Challenge Query:** When $\mathcal{A}$ sends the challenge messages (m_0, m_1), $\mathcal{B}$ forwards it to the cpABE challenger. The cpABE challenger samples a bit $\hat{\beta} \leftarrow \{0, 1\}$ and returns $\mathsf{cpABE.ct}_{\hat{\beta}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}}, m_{\hat{\beta}})$ to $\mathcal{B}$. $\mathcal{B}$ sends $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}, \mathsf{cpABE.ct}_{\hat{\beta}})$ to $\mathcal{A}$.

[24]To keep the proofs simple, we let f to be given selectively. This is sufficient to achieve security as in Def 3.14, as mentioned in Remark 4.

5. **Encryption Query:** When $\mathcal{A}$ makes the encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:

   - Computes $\mathsf{RMFE.}\overline{\mathsf{ct}} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, \bar{f}, \bar{L})$ and constructs the circuit $C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}$ as defined in the construction.
   - Computes $\mathsf{cpABE.}\overline{\mathsf{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}, \bar{m})$.
   - Returns $\mathsf{RPE.}\overline{\mathsf{ct}} = (\mathsf{RMFE.}\overline{\mathsf{ct}}, \mathsf{cpABE.}\overline{\mathsf{ct}})$ to $\mathcal{A}$.

6. In the end, $\mathcal{A}$ outputs a bit β'.

Observe that if the cpABE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where m_0 was encrypted, else it simulated the real world where m_1 was encrypted with $\mathcal{A}$.

Hence, advantage of $\mathcal{B}$ is $|\mathsf{Pr}[\beta' = 1|\hat{\beta} = 0] - \mathsf{Pr}[\beta' = 1|\hat{\beta} = 1]| = |\mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_0, L)] - \mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m_1, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. We observe that for the challenge circuit $C_{L,\mathsf{RMFE.ct}}$ and for all key queries $(\mathsf{lb}_i, x_i), i \in [Q]$, queried by $\mathcal{B}$, we have $C_{L,\mathsf{RMFE.ct}}((\mathsf{lb}_i, x_i), \mathsf{RMFE.sk}_{\mathsf{lb}_i,x_i}) = 0$ as either (i) $\mathsf{lb}_i \in L$, or (ii) $f(x_i) = 0$, which implies $\mathsf{RMFE.Dec}(\mathsf{RMFE.sk}_{\mathsf{lb}_i,x_i}, \mathsf{RMFE.ct}) = 0$ (due to RMFE correctness), by the admissibility condition on $\mathcal{A}$. So, if $\mathcal{A}$ is admissible then so is $\mathcal{B}$. ■

Function Hiding Security

Theorem 3.35. *Assume* RMFE *satisfies* 1*-query selective function hiding security (Def. 3.21), then the* RPE *scheme satisfies* 1*-query selective function hiding security (Def. 3.17).*

Proof. Recall that in the function hiding security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_0, m, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_1, m, L),$$

where for all the key queries (lb, x) to the $\mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \cdot, \cdot)$ oracle, either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$.

We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the 1-query selective function hiding security game, then there exists a PPT adversary $\mathcal{B}$ against the 1-query selective function-hiding security of RMFE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ runs $\mathcal{A}$ and gets the revocation list L.

2. $\mathcal{B}$ sends L as the challenge revocation list to the RMFE challenger. The challenger generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$ and returns $\mathsf{RMFE.mpk}$ to $\mathcal{B}$.

3. $\mathcal{B}$ generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends it to $\mathcal{A}$.

4. **Key Queries:** On each key query (lb, x), $\mathcal{B}$ does the following:

   - It sends a key query (lb, x) to the RMFE challenger and gets back $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.
   - Sets $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$ and computes $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
   - It returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$ to $\mathcal{A}$.

5. **Challenge Query:** When $\mathcal{A}$ sends the challenge functions (f_0, f_1) and message m, $\mathcal{B}$ does the following:

   - Sends (f_0, f_1) as the challenge functions to the RMFE challenger. The challenger samples $\hat{\beta} \leftarrow \{0, 1\}$, computes $\mathsf{RMFE.ct}_{\hat{\beta}} \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f_{\hat{\beta}}, L)$ and returns $\mathsf{RMFE.ct}_{\hat{\beta}}$ to $\mathcal{B}$.
   - Constructs the circuit $C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}$ as defined in the construction and computes $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}, m)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}_{\hat{\beta}}, \mathsf{cpABE.ct})$ to $\mathcal{A}$.

6. **Encryption Query:** When $\mathcal{A}$ makes an encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:

   - Sends a SK-Enc query $(\bar{f}, \bar{L})$ to the RMFE challenger and gets back $\mathsf{RMFE.}\overline{\mathsf{ct}}$.

- Constructs the circuit $C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}$ as defined in the construction.
- Computes $\mathsf{cpABE.}\overline{\mathsf{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}, \bar{m})$.
- Returns $\mathsf{RPE.}\overline{\mathsf{ct}} = (\mathsf{RMFE.}\overline{\mathsf{ct}}, \mathsf{cpABE.}\overline{\mathsf{ct}})$ to $\mathcal{A}$.

7. In the end, the adversary outputs a bit β'.

Observe that if the RMFE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where f_0 was encrypted, else it simulated the real world where f_1 was encrypted with $\mathcal{A}$. Hence, advantage of $\mathcal{B}$ is $|\mathsf{Pr}[\beta' = 1|\hat{\beta} = 0] - \mathsf{Pr}[\beta' = 1|\hat{\beta} = 1]| = |\mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_0, m, L)] - \mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_1, m, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. First, we note that since $\mathcal{A}$ is allowed to make only one query, $(\bar{f}, \bar{m}, \bar{L})$ to the $\mathsf{RPE.Enc}(\mathsf{msk}, \cdot, \cdot, \cdot)$ oracle, $\mathcal{B}$ also makes only one query, $(\bar{f}, \bar{L})$, to the $\mathsf{RMFE.SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle. Next, we observe that since $\mathcal{A}$ is restricted to make key queries (lb, x) such that either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$, thus $\mathcal{B}$ also issues key queries (lb, x) to RMFE challenger such that either $f_0(x) = f_1(x)$ or $\mathsf{lb} \in L$. Hence, if $\mathcal{A}$ is admissible, then so is $\mathcal{B}$. ■

Broadcast Security

Theorem 3.36. *Assume* RMFE *satisfies* 1*-query mode hiding security (Def. 3.20), then the* RPE *scheme satisfies* 1*-query selective broadcast security (Def. 3.18).*

Proof. Recall that in the broadcast security game, we want

$$\mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L) \approx_c \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L),$$

where $f(x) = 1, \forall x \in \mathcal{X}$.

We show that if there exists an adversary $\mathcal{A}$ who has non-negligible advantage ϵ in the broadcast security game, then there exists a PPT adversary $\mathcal{B}$ against the mode-hiding

security of RMFE scheme with the same advantage ϵ. The reduction is as follows.

1. $\mathcal{B}$ runs $\mathcal{A}$ and gets the revocation list L.

2. $\mathcal{B}$ sends L as the challenge revocation list to the RMFE challenger. The challenger generates $(\mathsf{RMFE.mpk}, \mathsf{RMFE.msk})$ and returns $\mathsf{RMFE.mpk}$ to $\mathcal{B}$.

3. $\mathcal{B}$ generates $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk}) \leftarrow \mathsf{cpABE.Setup}(1^\lambda)$. It sets $\mathsf{RPE.mpk} = (\mathsf{RMFE.mpk}, \mathsf{cpABE.mpk})$ and sends it to $\mathcal{A}$.

4. **Key Queries:** On each key query (lb, x), $\mathcal{B}$ does the following:

   - It sends a key query (lb, x) to the RMFE challenger and gets back $\mathsf{RMFE.sk}_{\mathsf{lb},x}$.
   - Sets $\mathsf{att} = ((\mathsf{lb}, x), \mathsf{RMFE.sk}_{\mathsf{lb},x})$ and computes $\mathsf{cpABE.sk}_{\mathsf{lb},x} \leftarrow \mathsf{cpABE.KeyGen}(\mathsf{cpABE.msk}, \mathsf{att})$.
   - It returns $\mathsf{RPE.sk}_{\mathsf{lb},x} = (\mathsf{att}, \mathsf{cpABE.sk}_{\mathsf{lb},x})$ to $\mathcal{A}$.

5. **Challenge Query:** When $\mathcal{A}$ sends the challenge function f and message m, $\mathcal{B}$ does the following:

   - Sends f as the challenge function to the RMFE challenger. The challenger samples $\hat{\beta} \leftarrow \{0,1\}$, and returns $\mathsf{RMFE.ct}_{\hat{\beta}}$ to $\mathcal{B}$, where $\mathsf{RMFE.ct}_0 \leftarrow \mathsf{RMFE.SK\text{-}Enc}(\mathsf{RMFE.msk}, f, L)$ and $\mathsf{RMFE.ct}_1 \leftarrow \mathsf{RMFE.PK\text{-}Enc}(\mathsf{RMFE.mpk}, L)$.
   - Constructs the circuit $C_{L,\mathsf{RMFE.ct}_{\hat{\beta}}}$ as defined in the construction and computes $\mathsf{cpABE.ct} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{L,\mathsf{RMFE,ct}_{\hat{\beta}}}, m)$.
   - Returns $\mathsf{RPE.ct} = (\mathsf{RMFE.ct}_{\hat{\beta}}, \mathsf{cpABE.ct})$ to $\mathcal{A}$.

6. **Encryption Query:** When $\mathcal{A}$ makes an encryption query $(\bar{f}, \bar{m}, \bar{L})$, $\mathcal{B}$ does the following:

   - Sends a $\mathsf{SK\text{-}Enc}$ query $(\bar{f}, \bar{L})$ to the RMFE challenger and gets back $\mathsf{RMFE.}\overline{\mathsf{ct}}$.
   - Constructs the circuit $C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}$ as defined in the construction.
   - Computes $\mathsf{cpABE.}\overline{\mathsf{ct}} \leftarrow \mathsf{cpABE.Enc}(\mathsf{cpABE.mpk}, C_{\bar{L},\mathsf{RMFE.}\overline{\mathsf{ct}}}, \bar{m})$.
   - Returns $\mathsf{RPE.}\overline{\mathsf{ct}} = (\mathsf{RMFE.}\overline{\mathsf{ct}}, \mathsf{cpABE.}\overline{\mathsf{ct}})$ to $\mathcal{A}$.

7. In the end, the adversary outputs a bit β'.

Observe that if the RMFE challenger chose $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the real world where the ciphertext was computed using Enc algorithm else it simulated the real world where the ciphertext was computed using Broadcast algorithm, with $\mathcal{A}$.
Hence, the advantage of $\mathcal{B}$ is $|\mathsf{Pr}[\beta' = 1|\hat{\beta} = 0] - \mathsf{Pr}[\beta' = 1|\hat{\beta} = 1]| = |\mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f, m, L)] - \mathsf{Pr}[\beta' = 1|\mathsf{RPE.ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.msk}, m, L)]| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. First, we note that since $\mathcal{A}$ is allowed to make only one query, $(\bar{f}, \bar{m}, \bar{L})$ to the $\mathsf{RPE.Enc}(\mathsf{msk}, \cdot, \cdot, \cdot)$ oracle, $\mathcal{B}$ also makes only one query, $(\bar{f}, \bar{L})$, to the $\mathsf{RMFE.SK\text{-}Enc}(\mathsf{msk}, \cdot, \cdot)$ oracle. Next, we observe that since $\mathcal{A}$ is restricted to output f, such that $f(x) = 1$ for all $x \in \mathcal{X}$, this implies $\mathcal{B}$ also issues such f as the challenge query to the RMFE challenger in the above mode hiding security game. Thus, if $\mathcal{A}$ is admissible, then so is $\mathcal{B}$. ■

3.8 EMBEDDED IDENTITY TRACE AND REVOKE

In this section, we define different variants of an embedded identity trace and revoke system (EITR). Our definitions extend the different notions that Goyal et al. [113] introduced, in the context of embedded identity traitor tracing, to incorporate the revocation list. Concretely, we define three variants of an EITR scheme: 1) Indexed EITR, 2) Bounded EITR, and 3) Unbounded EITR. Our goal is Unbounded EITR and other variants are introduced as intermediate goals. We show a construction of indexed EITR from RPE in Sec. 3.9, bounded EITR from indexed EITR in Sec. 3.10, and unbounded EITR from bounded EITR in Sec. 3.11. These implications hold for both secret key and public key settings.

An EITR scheme consists of five polynomial time algorithms–Setup, KeyGen, Enc, Dec and Trace. The syntax of the EITR variants mentioned above differs only in the inputs to the Setup, KeyGen and Trace algorithms.

Here we give a unified definition and later specify the distinctness of the three variants.

Consider a general identity space $\mathcal{GID}$, a label space $\mathcal{L}$ and a message space $\mathcal{M}$. An embedded identity trace and revoke scheme $\mathsf{EITR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ has the following syntax:

$\mathsf{Setup}(1^\lambda, \mathsf{params}_1) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and parameters params_1. It outputs a master public key mpk and a master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{gid}) \to \mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$. The key generation algorithm takes as input the master secret key msk, a label $\mathsf{lb} \in \mathcal{L}$, and a general identity $\mathsf{gid} \in \mathcal{GID}$. It outputs a secret key $\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk, a message $m \in \mathcal{M}$, a revocation list $L \subseteq \mathcal{L}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}, \mathsf{ct}, L) \to y$. The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}$, a ciphertext ct, and a revocation list L and outputs $y \in \mathcal{M} \cup \{\perp\}$.

$\mathsf{Trace}^D(\mathsf{tk}, \mathsf{params}_2, m_0, m_1, L) \to T$. The tracing algorithm takes as input a tracing key tk, parameter params_2, two messages m_0, m_1, a revocation list L and has an oracle access to a decoder D. It outputs a set of traitors T.

The above syntax captures both public key and secret key trace EITR schemes. For any *public tracing* EITR scheme, we have $\mathsf{tk} = \mathsf{mpk}$ and in the *secret tracing* EITR scheme, $\mathsf{tk} = \mathsf{msk}$. We now describe the properties satisfied by an EITR scheme.

Definition 3.22 (Correctness)**.** An EITR scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, any label $\mathsf{lb} \in \mathcal{L}$, and any revocation

list $L \subseteq \mathcal{L}$ such that $\mathsf{lb} \notin L$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{gid}}, \mathsf{ct}, L) = m : \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{params}_1); \\ \mathsf{sk}_{\mathsf{lb},\mathsf{gid}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{gid}); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m, L) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 3.23 (ind-cpa Security)**.** An EITR scheme is said to be ind-cpa secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_\beta) = \beta : \begin{array}{l} \mathsf{params}_1 \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{params}_1); \\ (m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}); \\ \beta \leftarrow \{0,1\}; \mathsf{ct}_\beta \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b, L) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where the adversary has the access to the $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$ oracle which has msk hardwired and $\mathcal{A}$ is admissible only if for all the key generation queries $(\mathsf{lb}, \mathsf{gid})$ to the KeyGen oracle, $\mathsf{lb} \in L$.

Definition 3.24 (Very Selective ind-cpa Security)**.** The very selective ind-cpa security of an EITR scheme is defined in the same way as Def. 3.23, except that the adversary outputs the revocation list L and all the key queries $\{\mathsf{lb}_i, \mathsf{gid}_i\}_{i\in[Q]}$, where Q is the number of key queries made, along with params_1 before the Setup algorithm is run.

In the following, we specify the space $\mathcal{GID}$, inputs $\mathsf{params}_1, \mathsf{gid}$ and params_2 to the Setup, KeyGen and Trace algorithm, respectively, and then define the secure tracing guarantee of each of the EITR notions separately. We let $\mathcal{ID} = \{0,1\}^\kappa$ denote the identity space.

Remark 7. We assume that there exists an efficiently computable mapping $\mathsf{map} : \mathcal{GID} \to \mathcal{L}$, that *uniquely* maps an $\mathsf{gid} \in \mathcal{GID}$ to a label $\mathsf{lb} \in \mathcal{L}$. This can be easily ensured, for e.g. by making label lb a part of id. We further note that in real world applications one may also want to ensure that any label lb is associated with at most one id. This can be achieved by using a collision resistant hash function.

Experiment $\mathsf{Exp\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$

- $1^\kappa, 1^{n_{\mathsf{ind}}} \leftarrow \mathcal{A}(1^\lambda)$
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa, n_{\mathsf{ind}})$
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot)}(\mathsf{mpk})$
- $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{1/\epsilon(\lambda)}, m_0, m_1, L)$

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id}, i)$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$. The adversary is admissible only if it makes at most one key query for each index.

Figure 3.4: Security Experiment: $\mathsf{Exp\text{-}Ind\text{-}TR}$

3.8.1 Indexed Trace and Revoke with Embedded Identity

In an indexed EITR scheme, the key is generated w.r.t. an identity and an index. We have $\mathcal{GID} = \mathcal{ID} \times [n_{\mathsf{ind}}]$, where $[n_{\mathsf{ind}}]$ is the index space for $n_{\mathsf{ind}} \in \mathbb{N}$, $\mathsf{params}_1 = (1^\kappa, n_{\mathsf{ind}})$, $\mathsf{gid} = (\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$, and $\mathsf{params}_2 = y$ for some $y > 1$.

We now define the secure tracing requirement.

Definition 3.25 (Secure Tracing)**.** Let $\mathsf{Ind\text{-}TR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be an indexed EITR scheme. For any non-negligible function $\epsilon(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define an experiment $\mathsf{Exp\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$ as in Figure 3.4.

Let S be the set of key queries $(\mathsf{lb}, \mathsf{id}, i)$ queried by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L}, i \in [n_{\mathsf{ind}}] \text{ s.t } (\mathsf{lb}, \mathsf{id}, i) \in S\}$.

Consider the following probabilistic events and their corresponding probabilities:

- $\mathsf{Good\text{-}Decoder} : \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b, L)] \geq 1/2 + \epsilon(\lambda)$
  $\mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) = \mathsf{Pr}[\mathsf{Good\text{-}Decoder}]$.
- $\mathsf{Cor\text{-}Tr} : |T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists i \in [n_{\mathsf{ind}}], \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}, i) \in S\}$.
  $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \mathsf{Pr}[\mathsf{Cor\text{-}Tr}]$.
- $\mathsf{Fal\text{-}Tr} : (T \not\subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \mathsf{Pr}[\mathsf{Fal\text{-}Tr}]$.

The Ind-TR scheme is said to satisfy secure tracing property if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\text{Pr-Fal-Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \text{Pr-Cor-Tr}_{\mathcal{A},\epsilon}(\lambda) \geq \text{Pr-Good-Decoder}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 3.26 (Very Selective Secure Tracing)**.** The very selective secure tracing of an indexed EITR scheme is defined in the same way as Def. 3.25, except that the adversary outputs the challenge revocation list L and all the key queries $(\mathsf{lb}, \mathsf{id}, i)$ along with $(1^\kappa, 1^{n_{\mathsf{ind}}})$ in the beginning of the $\text{Exp-Ind-TR}_{\mathcal{A},\epsilon}(\lambda)$.

Remark 8. In the definition above, Setup takes n_{ind} as an input in the binary form rather than in the unary form. This indicates that Setup runs in polynomial time, even if $n_{\mathsf{ind}} = 2^{\text{poly}(\lambda)}$. On the other hand, the adversary outputs n_{ind} in the unary form in the security game (Fig. 3.4). This indicates that we consider the security game only for the case where $n_{\mathsf{ind}} = \text{poly}(\lambda)$. Looking ahead, the former property is necessary when we convert bounded EITR into unbounded EITR in Sec. 3.11.

3.8.2 Bounded Trace and Revoke with Embedded Identity

In a bounded EITR scheme, we have $\mathcal{GID} = \mathcal{ID}$, $\mathsf{params}_1 = (1^\kappa, 1^{n_{\mathsf{bd}}})$, where $n_{\mathsf{bd}} \in \mathbb{N}$ is the bound on the number of key queries that an adversary can make in the *correct trace experiment* game, $\mathsf{gid} = \mathsf{id}$ for $\mathsf{id} \in \mathcal{ID}$, and $\mathsf{params}_2 = y$ for some $y > 1$.

We now define the secure tracing requirement.

Definition 3.27 (Secure Tracing)**.** Let BD-TR = (Setup, KeyGen, Enc, Dec, Trace) be a bounded EITR scheme. For any non-negligible function $\epsilon(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define an experiment $\text{Exp-BD-TR}_{\mathcal{A},\epsilon}(\lambda)$ as in Figure 3.5.

Let S be the set of key queries made by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L} \text{ s.t } (\mathsf{lb}, \mathsf{id}) \in S\}$.

Consider the following probabilistic events and their corresponding probabilities :

- Good-Decoder : $\text{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \text{Enc}(\mathsf{mpk}, m_b, L)] \geq 1/2 + \epsilon(\lambda)$
  $\text{Pr-Good-Decoder}_{\mathcal{A},\epsilon}(\lambda) = \text{Pr}[\text{Good-Decoder} \wedge |S_{\mathcal{ID}}| \leq n_{\mathsf{ind}}]$.

Experiment $\mathsf{Exp\text{-}BD\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$

- $1^{\kappa}, 1^{n_{\mathsf{bd}}} \leftarrow \mathcal{A}(1^{\lambda})$.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\kappa}, n_{\mathsf{bd}})$.
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk})$.
- $T \leftarrow \mathsf{Trace}^{D}(\mathsf{tk}, 1^{1/\epsilon}, m_0, m_1, L)$.

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id})$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$.

Figure 3.5: Security Experiment: Exp-BD-TR

- Cor-Tr : $|T| > 0, (T \subseteq S_{\mathcal{I}_{\mathcal{D}}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}) \in S\}$. $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \mathsf{Pr}[\mathsf{Cor\text{-}Tr}]$.
- Fal-Tr : $(T \not\subseteq S_{\mathcal{I}_{\mathcal{D}}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) = \mathsf{Pr}[\mathsf{Fal\text{-}Tr}]$.

The scheme is said to satisfy secure tracing if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$, such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 3.28 (Very Selective Secure Tracing)**.** The very selective secure tracing of a bounded EITR scheme is defined in the same way as Def. 3.27, except that the adversary outputs the challenge revocation list L and all the key queries $(\mathsf{lb}, \mathsf{id})$ along with $(1^{\kappa}, 1^{n_{\mathsf{bd}}})$ in the beginning of the $\mathsf{Exp\text{-}Ind\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$.

Remark 9. We point out that the bound on the number of key queries by the adversary is only required for the correct trace guarantee. The false trace guarantee will hold even if the adversary exceeds n_{bd} key queries– this is essential for the transformation from bounded-EITR scheme to unbounded-EITR scheme.

Experiment $\mathsf{Exp\text{-}TR}_{\mathcal{A},\epsilon,p}(\lambda)$

- $1^\kappa \leftarrow \mathcal{A}(1^\lambda)$.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$.
- $(D, m_0, m_1, L) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk})$.
- $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{1/\epsilon(\lambda)}, p(\lambda), m_0, m_1, L)$.

Here $\mathcal{A}$ has the oracle access to $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)$, which has msk hardwired and on query $(\mathsf{lb}, \mathsf{id})$, it outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id}}$.

Figure 3.6: Security Experiment: Exp-TR

3.8.3 Unbounded Trace and Revoke with Embedded Identity

In an unbounded EITR scheme, we have $\mathcal{GID} = \mathcal{ID}$, $\mathsf{params}_1 = 1^\kappa$, $\mathsf{gid} = \mathsf{id}$ for $\mathsf{id} \in \mathcal{ID}$, and $\mathsf{params}_2 = (y, \mathrm{Q}_{\mathsf{bd}})$ where $y > 1, \mathrm{Q}_{\mathsf{bd}} \in \mathbb{N}$.

We now define the secure tracing requirement.

Definition 3.29 (Secure Tracing)**.** Let $\mathsf{TR} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ be an unbounded EITR scheme. For any non-negligible function $\epsilon(\cdot)$, polynomial $p(\cdot)$ and stateful PPT adversary $\mathcal{A}$, define the experiment $\mathsf{Exp\text{-}TR}_{\mathcal{A},\epsilon,p}(\lambda)$ as in Figure 3.6.

Let S be the set of key queries made by $\mathcal{A}$ and $S_{\mathcal{ID}} = \{\mathsf{id} : \exists \mathsf{lb} \in \mathcal{L} \text{ s.t } (\mathsf{lb}, \mathsf{id}) \in S\}$.

Consider the following probabilistic events and their corresponding probabilities :

- Good-Decoder : $\Pr[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b, L) \geq 1/2 + \epsilon(\lambda)]$
  $\mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Good\text{-}Decoder} \wedge |S_{\mathcal{ID}}| \leq p(\lambda)]$.
- Cor-Tr : $|T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$, where $T_{\mathsf{lb}} = \{\mathsf{lb} \in \mathcal{L} : \exists \mathsf{id} \in T, (\mathsf{lb}, \mathsf{id}) \in S\}$.
  $\mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Cor\text{-}Tr}]$.
- Fal-Tr : $(T \not\subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$
  $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) = \Pr[\mathsf{Fal\text{-}Tr}]$.

The scheme is said to satisfy secure traitor tracing property if for every PPT adversary $\mathcal{A}$ and non-negligible function $\epsilon(\cdot)$,there exists negligible functions $\mathsf{negl}_1(\cdot)$ and $\mathsf{negl}_2(\cdot)$,

such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) \leq \mathsf{negl}_1(\lambda), \quad \mathsf{Pr\text{-}Cor\text{-}Tr}_{\mathcal{A},\epsilon,p}(\lambda) \geq \mathsf{Pr\text{-}Good\text{-}Decoder}_{\mathcal{A},\epsilon,p}(\lambda) - \mathsf{negl}_2(\lambda).$$

Definition 3.30 (Very Selective Secure Tracing)**.** The very selective secure tracing of an unbounded EITR scheme is defined in the same way as Def. 3.29, except that the adversary outputs the challenge revocation list L and all the key queries $(\mathsf{lb}, \mathsf{id})$ along with 1^κ in the beginning of the $\mathsf{Exp\text{-}TR}_{\mathcal{A},\epsilon}(\lambda)$.

Remark 10. [Remark 10.4, [113]]. Note that here the trace algorithm takes an additional parameter Q_{bd}. In the correct trace definition, we require that as long as the tracing algorithm uses a bound greater than the number of keys queried, the tracing algorithm must identify at least one traitor. However, the false trace guarantee should hold for all polynomially bounded Q_{bd} values. In particular, even if the number of keys queried is more than the bound used in tracing, the trace algorithm must not output an identity that was not queried. We can show that this definition implies the 'standard' tracing definition where the trace algorithm does not take this bound as input. One simply needs to run this bounded-version of trace with increasing powers of two until the trace algorithm outputs at least one traitor.

3.9 INDEXED TRACE AND REVOKE WITH EMBEDDED IDENTITY

In this section we construct an indexed (secret/public tracing)-EITR scheme from a (secret/public key)-RPE scheme. We present the construction and proofs for the secret trace setting primarily and also outline the differences in the public trace setting simultaneously.

3.9.1 Construction

Consider the identity space $\mathcal{ID} = \{0,1\}^\kappa$ and index bound n_{ind}. For our purpose, it suffices to assume $n_{\mathsf{ind}} \leq 2^{2\lambda}$. Let $\mathsf{RPE} = (\mathsf{RPE.Setup}, \mathsf{RPE.KeyGen}, \mathsf{RPE.Broadcast}, \mathsf{RPE.Enc}, \mathsf{RPE.Dec})$ be a

Inputs: an identity id and an index i.
Hardwired Values : Indices $j \in [n_{\mathsf{ind}}]$, $\ell \in [\kappa]$, a bit $b \in \{0,1\}$.
Output : $0/1$.

$$f[j,\ell,b](\mathsf{id},i) = \begin{cases} 1 & \text{if } (i > j) \vee (i = j \wedge \ell = \perp) \vee (i = j \wedge \mathsf{id}_\ell = 1 - b) \\ 0 & \text{otherwise} \end{cases}$$

Figure 3.7: Comparison Function $f[j,\ell,b]$

(secret/public key) RPE scheme[25] with attribute space $\mathcal{ID} \times [n_{\mathsf{ind}}]$, and supports the function class $\mathcal{F} = \{f[j,\ell,b]\}_{\substack{j\in[n_{\mathsf{ind}}],\ell\in[\kappa],\\ b\in\{0,1\}}}$, where $f[j,\ell,b] : \mathcal{ID} \times [n_{\mathsf{ind}}] \to \{0,1\}$ is as defined in figure 3.9.1.

We construct an indexed (secret/public tracing)-EITR scheme with identity space $\mathcal{ID} = \{0,1\}^\kappa$ and index bound n_{ind} as follows :

$\mathsf{Setup}(1^\lambda, 1^\kappa, n_{\mathsf{ind}}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

- Samples $(\mathsf{RPE.mpk}, \mathsf{RPE.msk}) \leftarrow \mathsf{RPE.Setup}(1^\lambda)$.
- Outputs $\mathsf{mpk} = \mathsf{RPE.mpk}$ and $\mathsf{msk} = \mathsf{RPE.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, (\mathsf{lb}, \mathsf{id}, i)) \to \mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$. The key generation algorithm does the following:

- Parse $\mathsf{msk} = \mathsf{RPE.msk}$.
- Sets $x = (\mathsf{id}, i)$ and runs $\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i} \leftarrow \mathsf{RPE.KeyGen}(\mathsf{RPE.msk}, \mathsf{lb}, x)$.
- Outputs $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i} = \mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \to \mathsf{ct}$. The encryption algorithm does the following:

- Parse $\mathsf{mpk} = \mathsf{RPE.mpk}$.
- Compute $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m, L)$. (In public trace setting, the algorithm computes $\mathsf{RPE.ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0]$,

[25] Public key RPE does not have the Broadcast algorithm.

$m, L)$).

- Outputs $\mathsf{ct} = \mathsf{RPE.ct}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}, \mathsf{ct}, L) \to m'$. The decryption algorithm does the following:

- Parse $\mathsf{sk}_{\mathsf{lb},\mathsf{id},i}$ as $\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}$ and ct as $\mathsf{RPE.ct}$.
- Computes and outputs $m' \leftarrow \mathsf{RPE.Dec}(\mathsf{RPE.sk}_{\mathsf{lb},\mathsf{id},i}, \mathsf{RPE.ct}, L)$.

$\mathsf{Trace}^{D}(\mathsf{tk}, y, m_0, m_1, L) \to T$. Parse tk as $\mathsf{msk} = \mathsf{RPE.msk}$ (in the public trace setting tk is $\mathsf{mpk} = \mathsf{RPE.mpk}$). The tracing algorithm is a two phased process. Briefly the tracing is implemented as follows:

1. First, we run the $\mathsf{Index.Trace}$ algorithm as defined in Figure 3.8, on the index space $[n_{\mathsf{ind}}]$. If the key associated with index $i \in [n_{\mathsf{ind}}]$ is used in constructing the decoder box, then the bit b in the output of the $\mathsf{Index.Trace}$ algorithm is 1. We maintain a set T_{index} to store all such indices on which $b = 1$.

2. Next, the $\mathsf{ID.Trace}$ algorithm, defined in Figure 3.9.1, takes the set T_{index} as input and uses the decoder D to compute the identity associated with each index.

Specifically, the algorithm runs as follows:

- Set $T_{\mathsf{index}} := \phi$. For $i = 1$ to n_{ind},
  - Compute $(b, p, q) \leftarrow \mathsf{Index.Trace}(\mathsf{tk}, y, m_0, m_1, L, i)$.
  - If $b = 1$, set $T_{\mathsf{index}} := T_{\mathsf{index}} \cup (i, p, q)$.
- Set $T = \phi$. For $(i, p, q) \in T_{\mathsf{index}}$,
  - Compute $\mathsf{id} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, y, m_0, m_1, L, (i, p, q))$.
  - Set $T := T \cup \{\mathsf{id}\}$.
- Output T.

Correctness. We prove that the above construction of indexed (secret/public tracing)-

Algorithm $\mathsf{Index.Trace}(\mathsf{tk}, y, m_0, m_1, L, i)$

Inputs: Tracing key tk, a parameter y, messages m_0, m_1, revocation list L and an index i.

Output : (b, p, q), where $b \in \{0,1\}$ and $p, q \in [0,1] \cup \{\perp\}$.

Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathsf{ind}}/\epsilon$ and $\mathsf{temp}_1 = \mathsf{temp}_2 = 0$. For $j = 1$ to N, it computes the following:

1. It samples $b_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,1} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \perp, 0], m_{b_j}, L)$ and sends $\mathsf{ct}_{j,1}$ to D.
   If D outputs b_j, set $\mathsf{temp}_1 = \mathsf{temp}_1 + 1$, else set $\mathsf{temp}_1 = \mathsf{temp}_1 - 1$.

2. It samples $c_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_{j,2} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i+1, \perp, 0], m_{c_j}, L)$ and sends $\mathsf{ct}_{j,2}$ to D.
   If D outputs c_j, set $\mathsf{temp}_2 = \mathsf{temp}_2 + 1$, else set $\mathsf{temp}_2 = \mathsf{temp}_2 - 1$.

If $\frac{\mathsf{temp}_1 - \mathsf{temp}_2}{N} > \frac{\epsilon}{4n_{\mathsf{ind}}}$, output $\left(1, \frac{\mathsf{temp}_1}{N}, \frac{\mathsf{temp}_2}{N}\right)$ else output $(0, \perp, \perp)$.

Figure 3.8: Index Tracing

Algorithm $\mathsf{ID.Trace}(\mathsf{tk}, y, m_0, m_1, L, (i, p, q))$

Inputs: Tracing key tk, a parameter y, messages m_0, m_1, revocation list L, index i, and probabilities p, q.

Output : $\mathsf{id} \in \{0,1\}^\kappa$

Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathsf{ind}}/\epsilon$ and $\mathsf{temp}_\ell = 0$ for $\ell \in [\kappa]$. For $\ell = 1$ to κ, it does as follows:

1. For $j = 1$ to N
   It samples $b_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}_j \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \ell, 0], m_{b_j}, L)$ and sends ct_j to D.
   If D outputs b_j, set $\mathsf{temp}_\ell = \mathsf{temp}_\ell + 1$, else set $\mathsf{temp}_\ell = \mathsf{temp}_\ell - 1$.

Let id be an empty string. For $\ell = 1$ to κ, do the following:

1. If $\frac{p+q}{2} > \frac{\mathsf{temp}_\ell}{N}$, set $\mathsf{id}_\ell = 0$, else set $\mathsf{id}_\ell = 1$.

Output $\mathsf{id} = \mathsf{id}_1 \| \cdots \| \mathsf{id}_\kappa$.

Figure 3.9: Identity Tracing

EITR scheme satisfies correctness (Def. 3.22) via the following theorem.

Theorem 3.37. *If* RPE *is a correct (secret/public key)-*RPE *scheme, then the above construction of indexed (secret/public tracing)-*EITR *scheme is correct.*

Proof. For the secret trace setting, the correctness follows directly from the broadcast correctness of the underlying (secret-key) RPE scheme.

For the public trace setting, firstly we observe that for any $(\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$, $f[1, \perp, 0](\mathsf{id}, i) = 1$. Hence, correctness follows directly from the encryption correctness of the underlying (public-key) RPE scheme.

■

Efficiency. We can instantiate the above construction using public key and secret key RPE. The size of each parameter is directly inherited from that of the underlying RPE. We have $|x| = |\mathsf{id}| + \log n_{\mathsf{ind}} \leq |\mathsf{id}| + \lambda$ and $|f| := |f[j, \ell, b]| = \log n_{\mathsf{ind}} + \log \kappa + 1 = O(\lambda)$. We note that here, $|f[j, \ell, b]|$ refers to the description size of the function $|f[j, \ell, b]|$, not the size of the circuit implementing it. In particular, the latter can depend on $|\mathsf{id}|$ while the former is independent from $|\mathsf{id}|$.

Secret Tracing Setting. If we instantiate the scheme with our secret key RPE in Sec. 3.7, we have $|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

Public Tracing Setting. If we instantiate the scheme with our public key RPE in Sec. 3.5, we have $|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |f|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{lb}|), |\mathsf{sk}| = \mathrm{poly}(\lambda, |f|, |x|, |\mathsf{lb}|) = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

3.9.2 Security

In this section we show that our construction of the indexed (secret/public tracing)-EITR scheme is secure.

IND-CPA security.

Theorem 3.38. *If (secret/public key)-*RPE *scheme satisfies (*0*-query- very selective/adaptive) message hiding property (Def. 3.14/Def. 3.13) then our construction of indexed (secret/public tracing)-*EITR *scheme is (very selective/adaptive)* ind-cpa *secure (Def. 3.23/Def. 3.24).*

Proof. Recall that for ind-cpa security, we need

$$\mathsf{Enc}(\mathsf{mpk}, m_0, L) \approx_c \mathsf{Enc}(\mathsf{mpk}, m_1, L).$$

For indexed *secret tracing* EITR scheme, this is equivalent to

$$\mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_0, L) \approx_c \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_1, L). \quad (3.5)$$

To prove this, we define the following hybrids:

Hyb_0 : This is the real world with the challenge bit $b = 0$. In particular, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_0, L)$.

Hyb_1 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1, \perp, 0], m_0, L)$. Indistinguishability from Hyb_0 follows from the broadcast security of secret-key RPE.

Hyb_2 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1, \perp, 0], m_1, L)$. Indistinguishability from Hyb_1 follows from the message hiding property of secret-key RPE.

Hyb_3 : In this hybrid, the challenger returns the ciphertext $\mathsf{ct} = \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_1, L)$. This is the real world with $b = 1$. Indistinguishability from Hyb_2 follows from the broadcast security of secret-key RPE.

For indexed *public trace* EITR scheme, ind-cpa security is equivalent to

$$\mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_0, L) \approx_c \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f[1,\perp,0], m_1, L). \tag{3.6}$$

The indistinguishability here follows directly from the message hiding property of the underlying RPE scheme. ■

Secure Tracing Analysis. We now show that our construction of indexed (secret/public tracing)-EITR scheme achieves the correct and false trace guarantees. The following analysis is mostly taken from ([113], Section 5.2.2) with appropriate modifications to incorporate the revocation list.

False Trace Guarantee. The false trace guarantee of a trace and revoke scheme ensures that the tracing algorithm does not falsely accuse any user. We prove that there does not exist a PPT adversary who can output a decoder D such that the tracing algorithm, when executed using this decoder D, outputs an identity for which key was not queried by the adversary or the corresponding label is in the revocation list.

Theorem 3.39. *For every stateful* PPT *adversary* $\mathcal{A}$ *and non-negligible function* $\epsilon(\cdot)$*, there exists negligible functions* $\mathsf{negl}(\cdot)$*, such that for all* $\lambda \in \mathbb{N}$

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}(\lambda)$$

where the probability $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}$ *is as defined in Def. 3.25.*

Proof. Let $S \subseteq \mathcal{L} \times \mathcal{ID} \times [n_{\mathsf{ind}}]$ be the set of label-identity-index pairs on which $\mathcal{A}$ issues key queries. That is, $S = \{(\mathsf{lb}, \mathsf{id}, i) : \mathcal{A}$ issues a key query on $(\mathsf{lb}, \mathsf{id}, i)\}$. Let us first recall the assumption that $\mathcal{A}$ can issue at most one key query for any index $i \in [n_{\mathsf{ind}}]$. Next, we set up some notations. For $\mathsf{lb} \in \mathcal{L}$, $i \in [n_{\mathsf{ind}}]$, $\mathsf{id} \in \mathcal{ID}$ and a revocation list L,

let

$$\begin{aligned} S_{\mathsf{index}} &= \{i : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ for some } (\mathsf{lb}, \mathsf{id}) \in \mathcal{L} \times \mathcal{ID}\}, \\ L_{\mathsf{index}} &= \{i : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ and } \mathsf{lb} \in L\}, \\ B &= \{(\mathsf{id}, i) : (\mathsf{lb}, \mathsf{id}, i) \in S \text{ and } \mathsf{lb} \notin L\}, \\ B_{\mathsf{index}} &= \{i : (\mathsf{id}, i) \in B\}. \end{aligned}$$

For any decoder box D, messages m_0, m_1, revocation list L, for any $i \in [n_{\mathsf{ind}}+1]$, $\ell \in [\kappa]$, $\mathsf{lb} \in \mathcal{L}$ we define

$$p^D_{i,\perp} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \perp, 0], m_b, L)]$$

$$p^D_{i,\ell} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[i, \ell, 0], m_b, L)]$$

where the probability is taken over the random coins to decoder D and the randomness used during the encryption. We show that $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon(\lambda)} \leq \mathsf{negl}(\lambda)$. For $i \in [n_{\mathsf{ind}}]$, $\ell \in [\kappa]$, we also define the following events:

$$\mathsf{Diff\text{-}Adv}^D_i : p^D_{i,\perp} - p^D_{i+1,\perp} > \epsilon/8n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{lwr}} : p^D_{i,\perp} - p^D_{i,\ell} > \epsilon/16n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{upr}} : p^D_{i,\ell} - p^D_{i+1,\perp} > \epsilon/16n_{\mathsf{ind}}$$

$$\mathsf{Diff\text{-}Adv}^D : \bigvee_{i \in [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})} \mathsf{Diff\text{-}Adv}^D_i \bigvee_{\substack{(\mathsf{id},i) \in B, \ell \in [\kappa], \\ \text{s.t } \mathsf{id}_\ell = 1}} \mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{lwr}} \bigvee_{\substack{(\mathsf{id},i) \in B, \ell \in [\kappa] \\ \text{s.t } \mathsf{id}_\ell = 0}} \mathsf{Diff\text{-}Adv}^D_{i,\ell,\mathsf{upr}}$$

We will drop the dependence on D for the ease of notation. We have

$$\begin{aligned} \mathsf{Pr}[\mathsf{Fal\text{-}Tr}] &= \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}]\mathsf{Pr}[\overline{\mathsf{Diff\text{-}Adv}}] + \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \mathsf{Diff\text{-}Adv}]\mathsf{Pr}[\mathsf{Diff\text{-}Adv}] \\ &\leq \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] + \mathsf{Pr}[\mathsf{Diff\text{-}Adv}] \end{aligned}$$

$$= \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] + \sum_{i \in [n_{\mathsf{ind}}]} \mathsf{Pr}[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_i]$$
$$+ \sum_{(i,\ell) \in [n_{\mathsf{ind}}] \times [\kappa]} \mathsf{Pr}\Big[\exists \mathsf{id} \in \{0,1\}^{\kappa} \text{ s.t } (\mathsf{id}, i) \in B$$
$$\wedge ([\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}} \wedge \mathsf{id}_\ell = 1] \vee [\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}} \wedge \mathsf{id}_\ell = 0])\Big]$$

Now, we argue that each of the terms on the RHS is bounded by a negligible function.

Lemma 3.40. *For every stateful* PPT *adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}_1(\cdot)$*, such that* $\forall \lambda \in \mathbb{N}$*, we have*

$$\mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq \mathsf{negl}_1(\lambda).$$

Proof. Note that the event Fal-Tr occurs if and only if the trace algorithm outputs an identity that was not queried by the adversary or outputs an identity which was queried but also revoked. Since, the tracing scheme is two phased, the false tracing can happen in any of the two stages. First, during the index-tracing procedure, it can happen that $\exists (i,p,q) \in T_{\mathsf{index}}$, such that, $i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}}$ and secondly, during the identity-tracing procedure, the ID.Trace algorithm outputs an incorrect identity corresponding to some $i \in S_{\mathsf{index}} \backslash L_{\mathsf{index}}$. So, we have

$$\mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}]$$
$$\leq \sum_{i \in [n_{\mathsf{ind}}]} \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \wedge i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge (\exists p,q : (i,p,q) \in T_{\mathsf{index}}) \mid \overline{\mathsf{Diff\text{-}Adv}}]$$
$$+ \sum_{(i,\ell) \in [n_{\mathsf{ind}}] \times [\kappa]} \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \wedge \exists \mathsf{id}, \hat{\mathsf{id}} : (\mathsf{id}, i) \in B \wedge (\exists p,q : (i,p,q) \in T_{\mathsf{index}})$$
$$\wedge \hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i,p,q)) \wedge \mathsf{id}_\ell \neq \hat{\mathsf{id}}_\ell \mid \overline{\mathsf{Diff\text{-}Adv}}].$$

Consider the first term on RHS. If $\overline{\mathsf{Diff\text{-}Adv}}$ happens then $\forall i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}}$ event $\overline{\mathsf{Diff\text{-}Adv}_i}$ happens.

We know that $\overline{\mathsf{Diff\text{-}Adv}_i}$ implies $p_{i,\perp} - p_{i+1,\perp} \leq \epsilon / 8 n_{\mathsf{ind}}$. Also the event $(\exists p,q : (i,p,q) \in T_{\mathsf{index}})$ implies $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} > \epsilon / 4 n_{\mathsf{ind}}$, where $\hat{p}_{j,\perp}$ is the estimated probability

for $p_{j,\perp}$.

By applying Chernoff bound, we have, for every $i \in [n_{\mathsf{ind}}]$

$$\mathsf{Pr}[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq e^{-\lambda/24} \leq 2^{-O(\lambda)}.$$

Now, in the second term, for a fixed (i, ℓ) the probability term corresponds to the event when the $\mathsf{ID.Trace}$ outputs an identity $\hat{\mathsf{id}}$ corresponding to index i, such that $\hat{\mathsf{id}}_\ell \neq \mathsf{id}_\ell$, where id is such that the adversary made a key query for (id, i). So, if the event $\overline{\mathsf{Diff\text{-}Adv}}$ happens then $\forall(\mathsf{id}, i) \in B, \ell \in [\kappa]$, event $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,X}}$ happens where $X = \mathsf{lwr}$ if $\mathsf{id}_\ell = 1$ else $X = \mathsf{upr}$.

Thus, when we have $\mathsf{id}_\ell = 1$, then $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}}}$ implies $p_{i,\perp} - p_{i,\ell} \leq \epsilon/16n_{\mathsf{ind}}$.

Also, the event $(\exists\ p, q\ :\ (i, p, q) \in T_{\mathsf{index}})$ implies that $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} \geq \epsilon/4n_{\mathsf{ind}}$ and the event $\hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i, p, q)) \wedge \hat{\mathsf{id}}_\ell = 0$ implies $(\hat{p}_{i,\perp} + \hat{p}_{i+1,\perp} > 2\hat{p}_{i,\ell})$. These together imply that $\hat{p}_{i,\perp} - \hat{p}_{i,\ell} > \epsilon/8n_{\mathsf{ind}}$. Similarly, when $\mathsf{id}_\ell = 0$ and $\hat{\mathsf{id}}_\ell = 1$, $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}}}$ implies $p_{i,\ell} - p_{i+1,\perp} \leq \epsilon/16n_{\mathsf{ind}}$ and following the similar reasoning as above, we get $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} \geq \epsilon/4n_{\mathsf{ind}}$ and $\hat{p}_{i,\perp} + \hat{p}_{i+1,\perp} \leq 2\hat{p}_{i,\ell}$ which implies $\hat{p}_{i,\ell} - \hat{p}_{i+1,\perp} \geq \epsilon/8n_{\mathsf{ind}}$.

Hence, using Chernoff bound, we get that

$$\mathsf{Pr}[\exists \mathsf{id}, \hat{\mathsf{id}} : (\mathsf{id}, i) \in S \wedge (\exists p, q : (i, p, q) \in T_{\mathsf{index}}) \wedge \hat{\mathsf{id}} \leftarrow \mathsf{ID.Trace}(\mathsf{tk}, 1^y, m_0, m_1, (i, p, q)) \wedge \hat{\mathsf{id}} \in T \wedge \mathsf{id}_\ell \neq \hat{\mathsf{id}}_\ell \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq 2^{-O(\lambda)}.$$

Combining the probability of both the RHS terms, we get

$$\mathsf{Pr}[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}] \leq n_{\mathsf{ind}} \cdot 2^{-O(\lambda)} + n_{\mathsf{ind}} \cdot \kappa \cdot 2^{-O(\lambda)} = \mathsf{negl}_1(\lambda).$$

■

Lemma 3.41. *If the underlying (secret/public key)-*RPE *satisfies (*1*-query-selective/adaptive) function hiding security property (Def. 3.17/Def. 3.16), then for every stateful* PPT *adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}_2(\cdot)$*, such*

that $\forall \lambda \in \mathbb{N}$ *and* $i \in [n_{\mathsf{ind}}]$*, we have*

$$\mathsf{Pr}[i \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_i] \leq \mathsf{negl}_2(\lambda).$$

Proof. We give a proof by contradiction. Suppose there exists a PPT adversary $\mathcal{A}$ that outputs a good decoder D along with messages m_0, m_1 and a revocation list L such that there exists $i^* \in [n_{\mathsf{ind}}]$ for which $\mathsf{Pr}[i^* \notin S_{\mathsf{index}} \backslash L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta$ where δ is some non-negligible function in the security parameter λ. We use this adversary $\mathcal{A}$ to build a reduction $\mathcal{B}$ that can break the function hiding security of the underlying RPE scheme. The reduction is as follows:

1. In the beginning, $\mathcal{A}$ outputs $1^{n_{\mathsf{ind}}}, 1^{\kappa}, L$ ($\mathcal{A}$ will output L adaptively in the public trae setting).

2. The RPE challenger samples $(\mathsf{RPE.mpk}, \mathsf{RPE.msk}) \leftarrow \mathsf{RPE.Setup}(1^{\lambda})$ and sends $\mathsf{RPE.mpk}$ to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{mpk} = \mathsf{RPE.mpk}$ and forwards it to $\mathcal{A}$.

3. When $\mathcal{A}$ issues a secret key query $(\mathsf{lb}, \mathsf{id}, j)$ (recall that $\mathcal{A}$ can make at most one query for each index j), $\mathcal{B}$ sends a key query to the RPE challenger on $(\mathsf{lb}, x = (\mathsf{id}, j))$. The challenger returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id},j}$ which $\mathcal{B}$ forwards to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a decoder D and messages m_0, m_1 (and a revocation list L in the case of public tracing EITR scheme) to $\mathcal{B}$. $\mathcal{B}$ then does the following:

   a) Samples $i \leftarrow [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})$ and sets $f_0 = f[i, \perp, 0]$ and $f_1 = f[i+1, \perp, 0]$.

   b) Samples $b \leftarrow \{0,1\}$ and sends (f_0, f_1, m_b) to the RPE challenger. The challenger samples $\alpha \leftarrow \{0,1\}$, computes $\mathsf{ct}_1 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.msk}, f_\alpha, m_b, L)$ and sends ct_1 to $\mathcal{B}$. (In public trace setting, $\mathsf{ct}_1 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\alpha, m_b, L)$.)

   c) $\mathcal{B}$ samples $\beta \leftarrow \{0,1\}$ and sends a encryption query on (f_β, m_b, L) and sets the ciphertext returned by the RPE challenger as ct_2. (In public trace setting, $\mathcal{B}$ computes ct_2 itself as $\mathsf{ct}_2 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\beta, m_b, L)$.)

   d) $\mathcal{B}$ samples $\beta \leftarrow \{0,1\}$, computes $\mathsf{ct}_2 \leftarrow \mathsf{RPE.Enc}(\mathsf{RPE.mpk}, f_\beta, m_b, L)$.

   e) If $D(\mathsf{ct}_1) = D(\mathsf{ct}_2)$, sets $\alpha' = \beta$, else $\alpha' = 1 - \beta$.

   f) $\mathcal{B}$ returns α' to the RPE challenger.

$\mathcal{B}$ wins the game if $\alpha = \alpha'$. Note that since $i \leftarrow [n_{\mathsf{ind}}] \setminus (S_{\mathsf{index}} \setminus L_{\mathsf{index}})$, for every key query $(\mathsf{lb}, \mathsf{id}, j)$ that $\mathcal{B}$ issues to the RPE challenger, either $f_0(\mathsf{id}, j) = f_1(\mathsf{id}, j)$ (when $i \notin S_{\mathsf{index}}$) or $\mathsf{lb} \in L$ (when $i \in S_{\mathsf{index}} \cap L_{\mathsf{index}}$). This establishes the admissibility of $\mathcal{B}$. Now, let us analyse the probability of $\mathcal{B}$ winning against the RPE challenger.

Let $q_{j,b} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[j, \perp, 0], m_b, L)]$.
Let E^D be the event that $\mathcal{B}$ wins and E^D_b be the event that $\mathcal{B}$ wins when it samples b as the message bit in step 4b. So, we have

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}^D_b] &= \frac{1}{4}\Big(\mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 0] + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 1] \\
&\qquad + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 1, \beta = 0] + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 1, \beta = 1]\Big) \\
&= \frac{1}{4}\left(q_{i,b}^2 + (1 - q_{i,b})^2 + 2(q_{i,b}(1 - q_{i+1,b}) + (1 - q_{i,b})q_{i+1,b}) + q_{i+1,b}^2 + (1 - q_{i+1,b})^2\right) \\
&= \frac{1}{2} + \frac{(q_{i,b} - q_{i+1,b})^2}{2}.
\end{aligned}$$

We have assumed $\exists i^* \in [n_{\mathsf{ind}}]$ such that $\mathsf{Pr}[i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta$. $\mathsf{Pr}[i = i^*] = 1/n_{\mathsf{ind}}$. So we get $\mathsf{Pr}[i = i^* \wedge i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}] \geq \delta/n_{\mathsf{ind}}$.

Let F be the event: $\exists i^* \in [n_{\mathsf{ind}}]$ such that $i = i^* \wedge i^* \notin S_{\mathsf{index}} \setminus L_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_{i^*}$. Then when F occurs, we have $p_{i,\perp} - p_{i+1,\perp} > \epsilon/8n_{\mathsf{ind}} \Rightarrow \exists b' \in \{0,1\}$ s.t $q_{i,b'} - q_{i+1,b'} > \epsilon/8n_{\mathsf{ind}}$. Also, $\mathsf{Pr}[\mathsf{E}^D_b] \geq 1/2$ for $b \in \{0,1\}$, irrespective of occurrence of F. Now,

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}^D_{b'}] =& \mathsf{Pr}[\mathsf{E}^D_{b'} \mid \mathsf{F}]\mathsf{Pr}[\mathsf{F}] + \mathsf{Pr}[\mathsf{E}^D_{b'} \mid \bar{\mathsf{F}}]\mathsf{Pr}[\bar{\mathsf{F}}] \\
\geq& \left(1/2 + \frac{\epsilon^2}{128{n_{\mathsf{ind}}}^2}\right) \times (\delta/n_{\mathsf{ind}}) + 1/2 \times (1 - \delta/n_{\mathsf{ind}}) \\
=& 1/2 + \frac{\epsilon^2\delta}{128{n_{\mathsf{ind}}}^3}.
\end{aligned}$$

Again,

$$\mathsf{Pr}[\mathsf{E}^D] = \frac{\mathsf{Pr}[\mathsf{E}^D_{b'}]}{2} + \frac{\mathsf{Pr}[\mathsf{E}^D_{\bar{b}'}]}{2} \geq \frac{1}{2}\left(\frac{1}{2} + \frac{\epsilon^2\delta}{128{n_{\mathsf{ind}}}^3}\right) + \frac{1}{4} = \frac{1}{2} + \eta, \text{ where } \eta = \frac{\epsilon^2\delta}{128{n_{\mathsf{ind}}}^3}.$$

Thus, $\mathcal{B}$ wins against the function-hiding security of the underlying RPE scheme with advantage $\geq \eta$, which is non-negligible for non-negligible ϵ and δ, a contradiction. Hence the lemma follows.

■

Lemma 3.42. *If the underlying (secret/public key)-*RPE *scheme satisfies (*1*-query-selective/adaptive) function hiding security property (Def. 3.17/Def. 3.16), then for every stateful* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}_3(\cdot)$*, such that* $\forall \lambda \in \mathbb{N}$ *and* $i \in [n_{\mathsf{ind}}], \ell \in [\kappa]$*, we have*

$$\mathsf{Pr}\left[\exists \mathsf{id} \in \{0,1\}^\kappa \; s.t \; (\mathsf{id}, i) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{lwr}} \wedge \mathsf{id}_\ell = 1] \vee [\mathsf{Diff\text{-}Adv}_{i,\ell,\mathsf{upr}} \wedge \mathsf{id}_\ell = 0])\right] \leq \mathsf{negl}_3(\lambda).$$

Proof. The proof is similar to the proof of Lemma 3.41. We show that if there exists a PPT adversary $\mathcal{A}$ who outputs a good decoder D along with messages m_0, m_1 and a revocation list L for which there exists $i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $\mathsf{Pr}[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*, \mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*} = 1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*} = 0])] \geq \delta$.

We use this adversary $\mathcal{A}$ to build a reduction $\mathcal{B}$ that can break the function hiding security of the underlying RPE scheme. $\mathcal{B}$ is defined in the same way as in the proof of Lemma 3.41, except Step 4a, which is now described as follows:

$\mathcal{B}$ samples $i \leftarrow B_{\mathsf{index}}$, $\ell \in [\kappa]$ and a bit $g \leftarrow \{0,1\}$ and sets $f_0 = f[i, \perp, 0]$ and $f_1 = f[i, \ell, 0]$, if $g = 0$, else sets $f_0 = f[i, \ell, 0]$ and $f_1 = f[i+1, \perp, 0]$.

Analysis of $\mathcal{B}$*'s advantage:*

Let $q_{j,k,b} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Enc}(\mathsf{tk}, f[j,k,0], m_b, L)]$.

Let E^D be the event that $\mathcal{B}$ wins and E^D_b be the event that $\mathcal{B}$ wins when $\mathcal{B}$ samples b as the message bit in step 4b. So, we have

$$\begin{aligned}\mathsf{Pr}[\mathsf{E}^D_b] = \frac{1}{8}\Big(&\mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 0, g = 0] + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 1, g = 0] \\ &+\mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 1, \beta = 0, g = 0] + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 1, \beta = 1, g = 0] \\ &+\mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 0, g = 1] + \mathsf{Pr}[\mathsf{E}^D_b \mid \alpha = 0, \beta = 1, g = 1]\end{aligned}$$

$$
\begin{aligned}
&+\mathsf{Pr}[\mathsf{E}_b^D \mid \alpha=1,\beta=0,g=1]+\mathsf{Pr}[\mathsf{E}_b^D \mid \alpha=1,\beta=1,g=1]\Big)\\
&=\frac{1}{8}\Big(q_{i,\perp,b}^2+(1-q_{i,\perp,b})^2+2(q_{i,\perp,b}(1-q_{i,\ell,b})+(1-q_{i,\perp,b})q_{i,\ell,b})+q_{i,\ell,b}^2+\\
&\quad(1-q_{i,\ell,b})^2+q_{i,\ell,b}^2+(1-q_{i,\ell,b})^2+2(q_{i,\ell,b}(1-q_{i+1,\perp,b})+\\
&\quad(1-q_{i,\ell,b})q_{i+1,\perp,b})+q_{i+1,\perp,b}^2+(1-q_{i+1,\perp,b})^2\Big)\\
&=\frac{1}{2}+\frac{(q_{i,\perp,b}-q_{i,\ell,b})^2}{4}+\frac{(q_{i,\ell,b}-q_{i+1,\perp,b})^2}{4}.
\end{aligned}
$$

We have assumed that $\exists\, i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $\mathsf{Pr}[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*,\mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*}=1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*}=0])] \geq \delta$.

Using $\mathsf{Pr}[i = i^* \wedge \ell = \ell^*] = 1/\kappa n_{\mathsf{ind}}$, we get $\mathsf{Pr}[(i=i^* \wedge \ell=\ell^*) \wedge \exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } ((i^*,\mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*}=1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*}=0]))] \geq \frac{\delta}{\kappa n_{\mathsf{ind}}}$

Let F be the event: $\exists\, i^* \in [n_{\mathsf{ind}}], \ell^* \in [\kappa]$ such that $(\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t } (i^*,\mathsf{id}) \in B \wedge ([\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{lwr}} \wedge \mathsf{id}_{\ell^*}=1] \vee [\mathsf{Diff\text{-}Adv}_{i^*,\ell^*,\mathsf{upr}} \wedge \mathsf{id}_{\ell^*}=0]))$. Then when F occurs, we have $(p_{i,\perp}-p_{i,\ell} > \epsilon/8n_{\mathsf{ind}}) \vee (p_{i,\ell}-p_{i+1,\perp} > \epsilon/8n_{\mathsf{ind}})$. This implies that there exists $b' \in \{0,1\}$ s.t $(q_{i,\perp,b'}-q_{i,\ell,b'} > \epsilon/8n_{\mathsf{ind}}) \vee ((q_{i,\ell,b'}-q_{i+1,\perp,b'} > \epsilon/8n_{\mathsf{ind}}))$. Also, $\mathsf{Pr}[\mathsf{E}_b^D] \geq 1/2$ for $b \in \{0,1\}$, irrespective of occurrence of F. Now,

$$
\begin{aligned}
\mathsf{Pr}[\mathsf{E}_{b'}^D] =& \mathsf{Pr}[\mathsf{E}_{b'}^D \mid \mathsf{F}]\mathsf{Pr}[\mathsf{F}]+\mathsf{Pr}[\mathsf{E}_{b'}^D \mid \bar{\mathsf{F}}]\mathsf{Pr}[\bar{\mathsf{F}}]\\
\geq& \big(1/2+\frac{\epsilon^2}{256{n_{\mathsf{ind}}}^2}\big)\times(\delta/\kappa n_{\mathsf{ind}})+1/2\times(1-\delta/\kappa n_{\mathsf{ind}})\\
=& 1/2+\frac{\epsilon^2\delta}{256\kappa{n_{\mathsf{ind}}}^3}.
\end{aligned}
$$

Again,

$$
\mathsf{Pr}[\mathsf{E}^D]=\frac{\mathsf{Pr}[\mathsf{E}_{b'}^D]}{2}+\frac{\mathsf{Pr}[\mathsf{E}_{\bar{b}'}^D]}{2} \geq \frac{1}{2}\left(\frac{1}{2}+\frac{\epsilon^2\delta}{256\kappa{n_{\mathsf{ind}}}^3}\right)+\frac{1}{4}=\frac{1}{2}+\eta, \text{ where } \eta=\frac{\epsilon^2\delta}{512{n_{\mathsf{ind}}}^3\kappa}.
$$

■

Combining the result of Lemmas 3.40, 3.41 and 3.42, we get

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda) + n_{\mathsf{ind}} \cdot \mathsf{negl}_2(\lambda) + n_{\mathsf{ind}} \cdot \kappa \cdot \mathsf{negl}_3(\lambda) = \mathsf{negl}(\lambda).$$

■

Correct trace guarantee. We prove that whenever an adversary outputs a good decoder, the tracing algorithm will output, with all but negligible probability, at least one valid user identity which was queried by the adversary.

Theorem 3.43. *If the underlying (secret/public key)-*RPE *scheme in the indexed (secret/public tracing)-*EITR *scheme construction satisfies (*1*-query-selective broadcast and very selective message/adaptive message) hiding property, then for every stateful* PPT *adversary* $\mathcal{A}$ *for the (very selective/adaptive)-tracing game (Def. 3.26/Def. 3.25) and non-negligible function* ϵ*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds*

$$\mathsf{Pr}[\mathsf{Cor\text{-}Tr}] \geq \mathsf{Pr}[\mathsf{Good\text{-}Decoder}] - \mathsf{negl}(\lambda)$$

Proof. Let us define $p^D_{\mathsf{Broadcast}} = \mathsf{Pr}[D(\mathsf{ct}) = b : b \leftarrow \{0,1\}, \mathsf{ct} \leftarrow \mathsf{RPE.Broadcast}(\mathsf{RPE.mpk}, m_b, L)]$. Then in the secret trace setting, if the event $\mathsf{Good\text{-}Decoder}_{\mathcal{A},\epsilon}$ occurs, then this implies $p^D_{\mathsf{Broadcast}} \geq 1/2 + \epsilon$. Further, from the broadcast security of (secret-key) RPE, we get $p^D_{1,\perp} \geq p^D_{\mathsf{Broadcast}} - \mathsf{negl}_1(\lambda)$, which implies

$$p^D_{1,\perp} \geq 1/2 + \epsilon - \mathsf{negl}_1(\lambda). \tag{3.7}$$

In public trace setting, the event $\mathsf{Good\text{-}Decoder}_{\mathcal{A},\epsilon}$ directly implies $p^D_{1,\perp} \geq 1/2 + \epsilon$ by definition. Also, by message hiding property of the underlying (secret/public key) RPE scheme, we have

$$p^D_{n_{\mathsf{ind}}+1,\perp} \leq 1/2 + \mathsf{negl}_2(\lambda) \tag{3.8}$$

for some negligible function $\mathsf{negl}_2(\cdot)$ with overwhelming probability. This is so, because

$f[n_{\mathsf{ind}}, \perp, 0](\mathsf{id}, i) = 0$ for all $(\mathsf{id}, i) \in \mathcal{ID} \times [n_{\mathsf{ind}}]$.

Combining equations (3.7) and (3.8), we get $p^D_{1,\perp} - p^D_{n_{\mathsf{ind}}+1,\perp} > \epsilon/2$.

Let $S^{\mathsf{ind}} = \{i \in [n_{\mathsf{ind}}] \mid p^D_{i,\perp} - p^D_{i+1,\perp} > \epsilon/2n_{\mathsf{ind}}\}$. Then if the event Good-Decoder occurs, $S^{\mathsf{ind}} \neq \phi$.

By Chernoff bound, we have

$$\forall i \in S^{\mathsf{index}}, \quad \Pr\left[\hat{p}^D_{i,\perp} - \hat{p}^D_{i+1,\perp} \leq \epsilon/4n_{\mathsf{ind}}\right] \leq 2^{-O(\lambda)} = \mathsf{negl}_3(\lambda)$$

for some negligible function $\mathsf{negl}_3(\cdot)$. Here, $\hat{p}$ denotes the estimate for p computed by tracing algorithm.

So, with all but negligible probability,

$$T_{\mathsf{index}} \neq \phi \text{ and } \forall (i, p, q) \in T_{\mathsf{index}}, p - q > \epsilon/4n_{\mathsf{ind}}$$

where T_{index} and p, q are as defined in the Trace algorithm 3.9.1.

Note that the ID.Trace algorithm (Figure 3.9.1) takes as input $(i, p, q) \in T_{\mathsf{index}}$ and outputs a corresponding id, where $\mathsf{id}_\ell = 1$ if $\hat{p}^D_{i,\ell} > (p + q)/2$ else $\mathsf{id}_\ell = 0$ for $\ell \in [\kappa]$. Then, for every $(i, p, q) \in T_{\mathsf{index}}$, the tracing algorithm outputs an identity. Hence, if $T_{\mathsf{index}} \neq \phi \Rightarrow T \neq \phi$, where T is defined as in the tracing algorithm. So,

$$\begin{aligned}
\Pr[T \neq \phi] \geq& \Pr[T \neq \phi \wedge \text{Good-Decoder}] \\
\geq& (1 - \mathsf{negl}_2(\lambda))\Pr[\text{Good-Decoder}] \\
\geq& \Pr[\text{Good-Decoder}] - \mathsf{negl}(\lambda)
\end{aligned}$$

for some negligible function $\mathsf{negl}(\cdot)$. Combining this with the false trace guarantee, we have

$$\Pr[\text{Cor-Tr}] \geq \Pr[\text{Good-Decoder}] - \mathsf{negl}(\lambda).$$

■

3.10 BOUNDED TRACE AND REVOKE WITH EMBEDDED IDENTITY

In this section we show how to construct a bounded (secret/public tracing)-EITR scheme from an indexed (secret/public tracing)-EITR scheme. The construction and security analysis in this section is an adaptation of ([113], Section 9) with appropriate modifications to incorporate the revocation list L. We present the construction and proofs for the secret trace setting primarily and also outline the differences in the public trace setting simultaneously.

3.10.1 Construction

Let Ind-TR = (Ind.Setup, Ind.KeyGen, Ind.Enc, Ind.Dec, Ind.Trace) be an indexed (secret/public tracing)-EITR system. Let $\sigma = (\sigma.\mathsf{KeyGen}, \sigma.\mathsf{Sign}, \sigma.\mathsf{Verify})$ be a signature scheme that satisfies unforgeability with signature space $\{0,1\}^{\ell_s}$. We let n_{bd} denote the bound on the number of key queries that the adversary can make. For our purpose, we can assume $n_{\mathsf{bd}} \leq 2^{\lambda}$. We construct a bounded (secret/public tracing)-EITR scheme as follows:

$\mathsf{Setup}(1^{\lambda}, 1^{\kappa}, n_{\mathsf{bd}}) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

1. Set $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$.
2. For $j = 1$ to λ, sample $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^{\lambda}, 1^{\kappa'}, n_{\mathsf{index}})$, where κ' is $\kappa + \ell_s$.
3. Sample $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \sigma.\mathsf{KeyGen}(1^{\lambda})$.
4. Output $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$ and $\mathsf{msk} = (\mathsf{sig.vk}, \mathsf{sig.sk}, \{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{lb},\mathsf{id}}$. The key generation algorithm does the following:

1. Parse msk as $(\mathsf{sig.vk}, \mathsf{sig.sk}, \{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$.
2. Compute $\sigma = \sigma.\mathsf{Sign}(\mathsf{sig.sk}, \mathsf{id})$ and let $\mathsf{id}' = (\mathsf{id}, \sigma)$.
3. For $j = 1$ to λ, do the following:
   a) Sample $i_j \leftarrow [2{n_{\mathsf{bd}}}^2]$.

b) $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_j, \mathsf{lb}, \mathsf{id}', i_j)$.

4. Return $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}\}_{j\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \rightarrow \mathsf{ct}$. The encryption algorithm does the following:

1. Parse mpk as $(\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$.
2. For $j = 1$ to $\lambda - 1$, randomly sample $r_j \leftarrow \mathcal{M}$ and set $r_\lambda = m \oplus r_1 \oplus \ldots \oplus r_{\lambda-1}$.
3. For $j = 1$ to λ, compute $\mathsf{Ind.ct}_j \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_j, r_j, L)$.
4. Return $\mathsf{ct} = \{\mathsf{Ind.ct}_j\}_{j\in[\lambda]}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb},\mathsf{id}}, \mathsf{ct}, L) \rightarrow m'$. The decryption algorithm does the following:

1. Parse $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}\}_{j\in[\lambda]}$ and $\mathsf{ct} = \{\mathsf{Ind.ct}_j\}_{j\in[\lambda]}$.
2. For $j = 1$ to λ, compute $r'_j = \mathsf{Ind.Dec}(\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},j}, \mathsf{Ind.ct}_j, L)$.
3. If any of the decryption fails then output $\perp$, else output $m' = r'_1 \oplus \cdots \oplus r'_\lambda$.

$\mathsf{Trace}^D(\mathsf{tk}, y, m_0, m_1, L) \rightarrow T^{\mathsf{final}}$. The trace algorithm uses two algorithms Bnd-isGoodDecoder and Bnd-Subtrace defined in Figures 3.10 and 3.11, respectively as subroutines and is defined as follows:

1. Parse tk as $\mathsf{msk} = (\mathsf{sig.vk}, \mathsf{sig.sk}, \{\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j\}_{j\in[\lambda]})$ and let $\mathsf{Ind.tk}_j = \mathsf{Ind.msk}_j$ for $j \in [\lambda]$. (In the public trace setting, $\mathsf{tk} = \mathsf{mpk}$ and $\mathsf{Ind.tk}_j = \mathsf{Ind.mpk}_j$).
2. Set $j = 1$.
3. Set $\mathsf{flag} = 0$. For $\mathsf{itr} = 1$ to $\lambda \cdot y$, do the following
   a) Choose a random message $r \leftarrow \mathcal{M}$.
   b) Run Bnd-isGoodDecoder as $\mathsf{flag} \leftarrow \mathsf{Bnd\text{-}isGoodDecoder}^D(\{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$.
   c) If $\mathsf{flag} = 1$, break. Else, continue.
4. If $\mathsf{flag} = 1$, run Bnd-Subtrace as

Algorithm $\mathsf{Bnd\text{-}isGoodDecoder}^{D}(\mathsf{key}, 1^y, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and position-index $i \in [\lambda]$.

Output : 0/1.

1. Set $\mathsf{count} = 0$. Let $\epsilon = 1/y$.
2. For $j = 1$ to $\lambda \cdot y$:
   - Split r in $\lambda - 1$ shares. That is, randomly sample $\lambda - 1$ messages, $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_\lambda$ such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.
   - Sample $b \leftarrow \{0,1\}$. For $k \in [\lambda] \setminus \{i\}$, compute $\mathsf{Ind.ct}_k \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_k, r_k, L)$. Compute $\mathsf{Ind.ct}_i \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_i, r \oplus m_b, L)$.
   - Query D on $\mathsf{ct} = (\mathsf{Ind.ct}_1, \ldots, \mathsf{Ind.ct}_\lambda)$. Let b' be D's response.
   - If $b' = b$, set $\mathsf{count} = \mathsf{count} + 1$.
3. If $\mathsf{count}/(\lambda \cdot y) \geq 1/2 + \epsilon/3$, then output 1, else output 0.

Figure 3.10: Algorithm Bnd-isGoodDecoder

$T \leftarrow \mathsf{Bnd\text{-}Subtrace}^{D}(\{\mathsf{Ind.mpk}_j, \mathsf{Ind.tk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$. Else, set $T = \phi$.

5. If $T = \phi$ and $j < \lambda$, set $j = j + 1$ and go to step 3. Otherwise do the following:
   - Set $T^{\mathsf{temp}} = \phi$. For each $\mathsf{id}' = (\mathsf{id}, \sigma) \in T$, if $\sigma.\mathsf{Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma) = 1$, add id to T^{temp}. Concretely,
     $$T^{\mathsf{temp}} = \{\mathsf{id} : \exists\sigma \; s.t. \; (\mathsf{id}, \sigma) \in T \text{ and } \sigma.\mathsf{Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma) = 1\}.$$
   - Recall the function $\mathsf{map} : \mathcal{ID} \rightarrow \mathcal{L}$, that maps a given identity id to its corresponding label lb (Remark 7). For each $\mathsf{id} \in T^{\mathsf{temp}}$, if $\mathsf{map}(\mathsf{id}) \in L$, then set $T^{\mathsf{temp}} = T^{\mathsf{temp}} \setminus \{\mathsf{id}\}$.
   - Set $T^{\mathsf{final}} = T^{\mathsf{temp}}$.
   - If $T^{\mathsf{final}} = \phi$ and $j < \lambda$, set $j = j + 1$ and go to step 3. Otherwise exit and return T^{final}.

Algorithm $\mathsf{Bnd\text{-}Subtrace}^{D}(\mathsf{key}, 1^{y}, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j, \mathsf{Ind.tk}_j\}_{j\in[\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and index-position $i \in [\lambda]$.

Output : $T \subseteq \{0,1\}^{\kappa}$

1. Define oracle $\tilde{D}[\{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}, r, L, i]$ as in Figure 3.12.

2. Output $T \leftarrow \mathsf{Ind.Trace}^{\tilde{D}}(\mathsf{Ind.tk}_i, 4y, m_0 \oplus r, m_1 \oplus r, L)$.

Figure 3.11: Algorithm Bnd-Subtrace

Algorithm $\tilde{D}^{D}[\mathsf{key}, r, L, i]$

Hardwired values: keys $\mathsf{key} = \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}$, message r, revocation list L and index-position $i \in [\lambda]$.

Inputs: $\mathsf{Ind.ct}$.

Output : 0/1

Upon input $\mathsf{Ind.ct}$, the $\tilde{D}$ oracle does the following:

- Shares r in $\lambda - 1$ components as follows: it chooses $\lambda - 1$ random messages r_k for $k \in [\lambda] \setminus \{i\}$, such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.

- For $k \in [\lambda] \setminus \{i\}$, computes $\mathsf{Ind.ct}_k = \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_k, r_k, L)$.

- Sets $\mathsf{ct_{bd}} = (\mathsf{Ind.ct}_1, \ldots, \mathsf{Ind.ct}_{i-1}, \mathsf{Ind.ct}, \mathsf{Ind.ct}_{i+1}, \ldots, \mathsf{Ind.ct}_{\lambda})$.

- Queries oracle D as $b' \leftarrow D(\mathsf{ct_{bd}})$.

- Outputs b'.

Figure 3.12: Oracle $\tilde{D}$

Correctness. We prove that the above construction of bounded (secret/public tracing)-EITR scheme satisfies correctness (Def. 3.22) via the following theorem.

Theorem 3.44. *Assume* Ind-TR *is a correct indexed (secret/public tracing)-*EITR *scheme then the above construction of bounded (secret/public tracing)-*EITR *scheme is correct.*

Proof. We have, as per the construction, that any message m is split in λ components $r_1, \ldots, r_\lambda$ such that $r_1 \oplus \cdots \oplus r_\lambda = m$. Then, from the correctness of Ind.TR, we get $r'_k = r_k$ for all $k \in [\lambda]$, where $r'_k \leftarrow \mathsf{Ind.Dec}(\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},k}, \mathsf{Ind.ct}_k, L)$, as long as $\mathsf{lb} \notin L$. Hence, the decryption algorithm correctly outputs m as $r_1 \oplus \cdots \oplus r_\lambda$. ■

Efficiency. We can instantiate the above construction by the indexed public/secret tracing-EITR scheme in Sec. 3.9.1. The above construction is basically λ times repetition of the underlying indexed EITR scheme. Additional overhead is induced due to the usage of the signature scheme, where identity becomes longer by $\ell_s = \mathrm{poly}(\lambda)$ bit and the master public key is longer by $|\mathsf{sig.vk}| = \mathrm{poly}(\lambda)$ bit.[26] These changes do not alter the dependency on $|\mathsf{id}|$ and $|\mathsf{lb}|$ of the parameter size. Therefore, the parameter size is as follows.

Secret Tracing Setting. In the secret tracing setting, we have $|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

Public Tracing Setting. In the public tracing setting, we have $|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathsf{lb}|), |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

3.10.2 Security

In this section, we prove that our construction of bounded (secret/public tracing)-EITR scheme is secure.

[26] Parameter sizes of most signature schemes depend on the length of the message space, which is $|\mathsf{id}|$ in our case. However, this dependency can be removed by hashing the message before signing using the collision resistant hash functions.

IND-CPA security.

Theorem 3.45. *If* Ind-TR *is a (very selective/adaptive)-*ind-cpa *secure indexed (secret/public tracing)-*EITR *scheme, then the above construction of bounded (secret/public tracing)-*EITR *scheme is (very selective/adaptive)-*ind-cpa *secure.*

Proof. We show that if there exists a PPT adversary that breaks the ind-cpa security of the bounded EITR scheme, then we can use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks ind-cpa security of the underlying Ind-TR scheme. The reduction is as follows:

1. $\mathcal{B}$ gets 1^κ, $1^{n_{\mathsf{bd}}}$, L and key queries $\{(\mathsf{lb}_i, \mathsf{id}_i)\}_{i\in[Q]}$, where Q is the number of key queries issued, from the adversary $\mathcal{A}$. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and outputs L adaptively along with the challenge messages in Step 6).

2. $\mathcal{B}$ generates $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \sigma.\mathsf{KeyGen}(1^\lambda)$.

3. For the i-th key query $(\mathsf{lb}_i, \mathsf{id}_i)$, $i \in [Q]$, $\mathcal{B}$ samples $i_1 \leftarrow [n_{\mathsf{index}}]$. It sets $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$ and sends $1^{\kappa+\ell_s}$, $1^{n_{\mathsf{index}}}$, L and $\{(\mathsf{lb}_i, \mathsf{id}'_i, i_1)\}_{i\in[Q]}$, where id' is computed as in the construction, to the Ind-TR challenger. The Ind-TR challenger returns $\mathsf{Ind.mpk}$ and $\{\mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i}\}_{i\in[Q]}$ where $\mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}, \mathsf{lb}_i, \mathsf{id}'_i, i_1)$.

4. $\mathcal{B}$ sets $\mathsf{Ind.mpk}_1 = \mathsf{Ind.mpk}$, generates $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^\lambda, 1^{\kappa+\ell_s}, n_{\mathsf{index}})$ for $j \in \{2, \cdots, \lambda\}$. It sends $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]})$ to $\mathcal{A}$.

5. For each key query $(\mathsf{lb}_i, \mathsf{id}_i)$, $\mathcal{B}$ sets $\mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i,i_1} = \mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i}$ and computes $\mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i,i_j}$ for $j \in \{2, \cdots, \lambda\}$ itself and returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{Ind.sk}_{\mathsf{lb}_i,\mathsf{id}_i,i_j}\}_{j\in\lambda}$ to $\mathcal{A}$.

6. When $\mathcal{A}$ sends the challenge messages (m_0, m_1) for the challenge query, $\mathcal{B}$ samples $r_j \leftarrow \mathcal{M}$ for $j \in \{2, \cdots, \lambda\}$, sets $m'_b = \bigoplus_{j>1} r_j \oplus m_b$, sends (m'_0, m'_1) for the challenge query to the Ind-TR challenger and gets back $\mathsf{Ind.ct}$. $\mathcal{B}$ sets $\mathsf{ct}_1 = \mathsf{Ind.ct}$, computes $\mathsf{ct}_j \leftarrow \mathsf{Ind.Enc}(\mathsf{Ind.mpk}_j, r_j, L)$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{ct} = \{\mathsf{ct}_j\}_{j\in\lambda}$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ returns b' to the challenger.

Observe that $\mathcal{B}$ issues key queries to the Ind-TR challenger only when there is a key query from $\mathcal{A}$. From the admissibility of $\mathcal{A}$, for any key query of the form $(\mathsf{lb}_i, \mathsf{id}_i, i_1)$ that $\mathcal{B}$ issues $\mathsf{lb} \in L$. So, $\mathcal{A}$ wins the ind-cpa security game of the BD-TR scheme

with advantage ϵ, then $\mathcal{B}$ also wins the ind-cpa security game of the underlying Ind-TR scheme with the same advantage. ■

Secure Tracing Analysis. Now we show that our construction satisfies false trace and correct trace guarantees.

False Trace Guarantee. First, we show that the probability of false trace in our scheme is negligible in the security parameter via the following theorem.

Theorem 3.46. *Assume that σ is unforgeable, then our construction of bounded (secret/public tracing)-*EITR *scheme satisfies the (very selective/adaptive) false trace guarantee (Def. 3.27/Def. 3.28), even if the adversary makes unbounded polynomial number of key queries.*

Proof. Let us first setup some notations. For $\mathsf{lb} \in \mathcal{L}$, $\mathsf{id} \in \mathcal{ID}$ and a revocation list L, let

$$\begin{aligned} S_{\mathcal{ID}} &= \{\mathsf{id} : (\mathsf{lb}, \mathsf{id}) \in S\}, \\ L_{\mathcal{ID}} &= \{\mathsf{id} : (\mathsf{lb}, \mathsf{id}) \in S \wedge \mathsf{lb} \in L\}, , \\ T^{\mathsf{final}}_{\mathsf{lb}} &= \{\mathsf{map}(\mathsf{id}) : \mathsf{id} \in T^{\mathsf{final}}\} \end{aligned}$$

where map is as defined in Remark 7 and T^{final} is as in the construcion.
False trace happens when $T^{\mathsf{final}} \not\subseteq S_{\mathcal{ID}}$ or $T^{\mathsf{final}}_{\mathsf{lb}} \cap L \neq \phi$. Observe that $T^{\mathsf{final}}_{\mathsf{lb}} \cap L = \phi$ by definition. So, all we need to argue is that $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}}$.

Recall that the tracing algorithm uses Bnd-Subtrace algorithm to find a set $T = \{(\mathsf{id}_k, \sigma_k)\}_k$. Identity id_k is added to the set T^{temp} and then to set T^{final} only if $\sigma.\mathsf{Verify}(\mathsf{sig.vk}, \mathsf{id}_k, \sigma_k) = 1$. Next we show that if there is an adversary $\mathcal{A}$ who outputs a decoder D along with messages m_0, m_1 and revocation list L such that the tracing algorithm outputs T^{final} where $T^{\mathsf{final}} \not\subseteq S_{\mathcal{ID}}$, then there exists a reduction $\mathcal{B}$ against

unforgeability of signature scheme σ. The reduction is defined as follows:

Upon receiving $\mathsf{sig.vk}$ from the σ challenger, $\mathcal{B}$ does the following:

1. Run $\mathcal{A}$ on input 1^λ to obtain $1^\kappa, 1^{n_{\mathsf{bd}}}, L$ and key queries $\{(\mathsf{lb}_i, \mathsf{id}_i)\}_{i\in[Q]}$, where Q is the number of key queries issued. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and output L adaptively along with the challenge messages).

2. Samples $(\mathsf{Ind.mpk}_j, \mathsf{Ind.msk}_j) \leftarrow \mathsf{Ind.Setup}(1^\lambda, 1^{\kappa'}, n_{\mathsf{index}})$ for $j \in [\lambda]$ and sends $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{mpk}_j\}_{j\in[\lambda]})$ to $\mathcal{A}$.

3. For each key query $(\mathsf{lb}_i, \mathsf{id}_i)$, $i \in [Q]$, $\mathcal{B}$ sends id_i to the σ challenger for signature. The σ challenger returns $\sigma_i = \sigma(\mathsf{sig.sk}, \mathsf{id}_i)$. $\mathcal{B}$ generates the secret key $\mathsf{sk}_{\mathsf{lb}_i,\mathsf{id}_i}$ using $\mathsf{id}'_i = (\mathsf{id}_i, \sigma_i)$ as in the construction and sends $\mathsf{sk}_{\mathsf{lb}_i,\mathsf{id}_i}$ to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a decoder D, messages m_0, m_1.

5. $\mathcal{B}$ runs the trace algorithm with the help of the decoder D and gets a set T^{final}.

6. If there exists an $\mathsf{id}^* \in T^{\mathsf{final}}$ such that $\mathsf{id}^* \notin S_{\mathcal{ID}}$, then $\mathcal{B}$ returns a forgery for id^* to σ challenger as follows: $\mathsf{id}^* \in T^{\mathsf{final}}$ implies that there exists σ^*, such that $(\mathsf{id}^*, \sigma^*) \in T$, and $\sigma.\mathsf{Verify}(\mathsf{sig.vk}, \mathsf{id}^*, \sigma^*) = 1$. $\mathcal{B}$ returns $(\mathsf{id}^*, \sigma^*)$ as a forgery to the σ challenger.

   Note that since $\mathsf{id}^* \notin S_{\mathcal{ID}}$, $\mathcal{B}$ must not have queried a signature on id^* to the σ challenger and hence $(\mathsf{id}^*, \sigma^*)$ is a valid forgery. If the false tracing happens with non-negligible probability ϵ, then $\mathcal{B}$ also wins the unforgeability game with probability ϵ.

■

Correct Trace Guarantee. Recall that in the experiment for correct tracing, the adversary first sends $(1^\kappa, 1^{n_{\mathsf{bd}}})$, a revocation list L (in the public trace setting, it outputs L adaptively), and at most n_{bd} key queries. Let $S = \{(\mathsf{lb}, \mathsf{id})\}$ be the set of label-identity pairs for which the adversary issues key queries. Next, the challenger sends the public key to the adversary. At the end, the adversary outputs a decoder box D along with two messages m_0 and m_1. If D is a good decoder then for correct tracing we want that the tracing algorithm outputs non empty set of traitors $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. We prove the correct trace guarantee of our scheme via the following theorem.

Theorem 3.47. *Assume that the underlying (secret/public tracing)-*Ind-TR *scheme satisfies (very selective/adaptive) correct trace guarantee (Def. 3.26 /Def. 3.25) and let* n_{bd} *be the bound on the number of key queries for an admissible adversary, then our bounded (secret/public tracing)-*EITR *scheme also satisfies the (very selective/adaptive) correct trace guarantee (Def. 3.28 /Def. 3.27).*

Proof. Similar to [113] we begin with defining some events of interest. We modify the definition of some events to take into account the constraint that $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. We drop the subscripts $\mathcal{A}$, ϵ and security parameter λ in the following to keep the notations simple.

- Event Admissible-Adversary: It is defined as the event that the adversary $\mathcal{A}$ makes at most n_{bd} key queries.

- Event Tr (Tracing without correctness): Similar to Corr-Tr, except that we don't need $T^{\mathsf{final}} \subseteq S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$, i.e., Tr is the event that $T^{\mathsf{final}} \neq \phi$ occurs. Denote Pr-Tr := Pr[Tr].

- Event Dist-Indx (Position with distinct indices for each key): Defined as the event that there exists $i \in [\lambda]$ such that the i-th index of each key is distinct.

- Event Dist-Indx$_i$: It is defined as the event that $i \in [\lambda]$ is the first position such that the i-th index of each key is distinct. By definition, Dist-Indx$_i$ are disjoint events for all $i \in [\lambda]$ and $\cup_{i\in[\lambda]}$Dist-Indx$_i$ = Dist-Indx.

- Event Tr$_i$ (Tracing without correctness in i-th iteration): Let T_i denote the set of (identity, signature) pairs traced in the i-th iteration. The event Tr$_i$ happens if T_i is non-empty.

- Event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ (Tracing with same signature as that received in key): The event that T_i is not empty and for all $(\mathsf{id}, \sigma) \in T_i$, $(\mathsf{lb}, \mathsf{id})$ was queried for a key and $\mathsf{lb} \notin L$, i.e. $\mathsf{id} \in S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$ and that the key generation oracle output $\mathsf{Ind.sk}_{\mathsf{lb},\mathsf{id},i} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_i, \mathsf{lb}, (\mathsf{id}, \sigma), j)$ for some j.

- Event $\mathsf{Found\text{-}Good\text{-}r}_i$: This event occurs if flag is set to 1 in the i-th iteration of the tracing algorithm.

- Event $\mathsf{Good\text{-}\tilde{D}}_i$ (Good decoder $\tilde{D}$ during the $\mathsf{Bnd\text{-}Subtrace}$ routine execution in i-th iteration): It is defined as the event that in the i-th iteration, the execution reaches step 3 (that is, it found a 'good' r in the i-th iteration), and the decoder $\tilde{D}$ constructed is an $\epsilon/4$ good decoder for distinguishing messages $m_0 \oplus r$ and $m_1 \oplus r$.

  Note that if no good r is found in step 3, $\mathsf{Good\text{-}\tilde{D}}_i$ is said to not have happened.

With the above events, the correctness of tracing is argued via the following series of inequalities:

$$\mathsf{Pr\text{-}Corr\text{-}Tr}(\lambda) \geq \mathsf{Pr\text{-}Tr} - \mathsf{negl} \tag{3.9}$$

$$\geq \mathsf{Pr}[\mathsf{Tr} \wedge \mathsf{Dist\text{-}Indx}] - \mathsf{negl} \tag{3.10}$$

$$= \sum_{i \in [\lambda]} \mathsf{Pr}[\mathsf{Tr} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.11}$$

$$\geq \sum_{i \in [\lambda]} \mathsf{Pr}[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.12}$$

$$\geq \sum_{i \in [\lambda]} \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.13}$$

$$\geq \sum_{i \in [\lambda]} \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.14}$$

$$\geq \sum_{i \in [\lambda]} \mathsf{Pr}[\mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary}$$

$$\wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.15}$$

$$\geq \sum_{i\in[\lambda]} \mathsf{Pr}[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \tag{3.16}$$

$$= \mathsf{Pr}[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}] - \mathsf{negl} \tag{3.17}$$

$$\geq \mathsf{Pr}[\mathsf{Good\text{-}Decoder} \wedge \mathsf{Admissible\text{-}Adversary}] - \mathsf{negl} \tag{3.18}$$

Explanation for each of the inequalities is the same as in [113] and is omitted. Here, we argue the transitions from equations (3.11) to (3.12) and (3.12) to (3.13) only, since our definition of event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ is slightly modified.

- Transition from (3.11) to (3.12) follows from the observation that the event $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ implies the event Tr because by definition, if $\mathsf{Corr\text{-}Tr\text{-}Sig}_i$ happens then T_i is non empty and for each (id, σ) pair in T_i, $\sigma.\mathsf{Verify}(\mathsf{sig.vk}, \mathsf{id}, \sigma)$ verifies and $\mathsf{id} \in S_{\mathcal{ID}} \setminus L_{\mathcal{ID}}$. Hence, the set of traitors T^{final} obtained from T_i will also be non empty.

- Transition from (3.12) to (3.13) is argued via the following claim:

Claim 3.48. Assume that the underlying $\mathsf{Ind\text{-}TR}$ scheme satisfies correct trace guarantee. Then for all $i \in [\lambda]$

$$\mathsf{Pr}[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i] \geq \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl}.$$

Proof. We prove the claim by contradiction. We assume that there exists an adversary $\mathcal{A}$ who outputs a good decoder D along with messages m_0, m_1 and a revocation list L such that $\mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{Pr}[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i]$ is non-negligible. Then we use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against correct trace guarantee of $\mathsf{Ind\text{-}TR}$, defined as follows:

1. Run $\mathcal{A}$ on input 1^λ to obtain 1^κ, $1^{n_{\mathsf{bd}}}$, L and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k)\}_{k\in[Q]}$, where Q is the number of key queries issued. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and output L adaptively along with the challenge messages). It sets $n_{\mathsf{index}} = 2{n_{\mathsf{bd}}}^2$.

2. Samples $(\mathsf{sig.sk}, \mathsf{sig.vk}) \leftarrow \sigma.\mathsf{KeyGen}(1^\lambda)$, computes $\sigma_k \leftarrow \sigma.\mathsf{Sign}(\mathsf{sig.sk}, \mathsf{id}_k)$ and sets $\mathsf{id}'_k = (\mathsf{id}_k, \sigma_k)$ for $k \in [Q]$.

3. For each $k \in [Q]$ and $j \in [\lambda]$, it randomly samples $k_j \leftarrow n_{\mathsf{index}}$. It sends $1^{\kappa+\ell_s}, 1^{n_{\mathsf{index}}}, L$ and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k, k_i)\}_{k\in[Q]}$ to the Ind-TR challenger and gets back $\mathsf{Ind.mpk}$ and $\{\mathsf{Ind.sk}_{\mathsf{lb}_k,\mathsf{id}_k,k_i}\}_{k\in[Q]}$.

4. $\mathcal{B}$ sets $\mathsf{Ind.mpk}_i = \mathsf{Ind.mpk}$ and does the following:

   - For $j \in [\lambda] \setminus \{i\}$, $(\mathsf{Ind.mpk}_k, \mathsf{Ind.msk}_k) \leftarrow \mathsf{Ind.Setup}(1^{\kappa+\ell_s}, 1^{n_{\mathsf{index}}})$.
   - Set $\mathsf{mpk} = (\mathsf{sig.vk}, \{\mathsf{Ind.mpk}_k\}_{k\in[\lambda]})$ and sends mpk to $\mathcal{A}$.

5. For each key query $(\mathsf{lb}_k, \mathsf{id}_k)$ by $\mathcal{A}$, $\mathcal{B}$ does the following:

   - For $j \in [\lambda] \setminus \{i\}$, $\mathsf{Ind.sk}_{\mathsf{lb}_k,\mathsf{id}_k,k_j} \leftarrow \mathsf{Ind.KeyGen}(\mathsf{Ind.msk}_k, \mathsf{lb}_k, \mathsf{id}'_k, k_j)$.
   - Sends $\mathsf{sk}_{\mathsf{lb,id}} = (\mathsf{Ind.sk}_{\mathsf{lb}_k,\mathsf{id}_k,k_1}, \ldots, \mathsf{Ind.sk}_{\mathsf{lb}_k,\mathsf{id}_k,k_\lambda})$ to $\mathcal{A}$.

6. If i is not the first position for which the i-th position indices (k_i's) are distinct for all the key queries, then $\mathcal{B}$ outputs a random decoder and quits the game.

7. In the end, $\mathcal{A}$ outputs a decoder D along with messages m_0, m_1.

8. $\mathcal{B}$ runs $\mathsf{Bnd\text{-}isGoodDecoder}(\{\mathsf{Ind.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, L, r)$ for uniformly and independently sampled r until it finds a r for which $\mathsf{Bnd\text{-}isGoodDecoder}$ algorithm outputs 1. If $\mathsf{Bnd\text{-}isGoodDecoder}$ algorithm does not output 1 even after $\lambda \cdot y$ attempts, then $\mathcal{B}$ outputs a random decoder and quits.

9. $\mathcal{B}$ constructs decoder $\tilde{D}$ as defined in Figure 3.12 and sets $\tilde{m}_b = m_b \oplus r$ for $b \in \{0, 1\}$ and sends $(\tilde{D}, \tilde{m}_0, \tilde{m}_1, L)$ to the Ind-TR challenger.

Now let us analyze the probability that $\mathcal{B}$ outputs a $1/4y$ good decoder box. This happens if (i) i is the first position such that the i-th position indices are different for all the key queries (ii) $\mathsf{Bnd\text{-}isGoodDecoder}$ outputs 1 for some r and $\tilde{D}$ is $\epsilon/4 = 1/4y$ good decoder for distinguishing $m_0 \oplus r, m_1 \oplus r$. First event is same as $\mathsf{Dist\text{-}Indx}_i$ and second event is same as $\mathsf{Good\text{-}\tilde{D}}_i$. Hence, the probability that $\mathcal{B}$ outputs a $1/4y$ good decoder box is $\mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Dist\text{-}Indx}_i]$. Then, by correct trace guarantee of Ind-TR,

$$\begin{aligned}\mathsf{Pr}[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i] &\geq \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl} \\ &\geq \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{negl}.\end{aligned}$$

This contradicts our assumption that $\mathsf{Pr}[\mathsf{Good\text{-}}\tilde{\mathsf{D}}_i \wedge \mathsf{Admissible\text{-}Adversary} \wedge \mathsf{Dist\text{-}Indx}_i] - \mathsf{Pr}[\mathsf{Corr\text{-}Tr\text{-}Sig}_i \wedge \mathsf{Dist\text{-}Indx}_i]$ is non-negligible, hence the proof. ■

■

3.11 UNBOUNDED TRACE AND REVOKE WITH EMBEDDED IDENTITIES

In this section we show how to construct unbounded (secret/public tracing)-EITR scheme from a bounded (secret/public tracing)-EITR scheme. The transformation technique and the security analysis in this section is adapted from [113] with modifications to incorporate the revocation list. We present the construction and proofs for *secret-key* trace setting primarily and also outline the differences in the public-key trace setting simultaneously.

3.11.1 Construction

Let $\mathsf{BD\text{-}TR} = (\mathsf{BD.Setup}, \mathsf{BD.KeyGen}, \mathsf{BD.Enc}, \mathsf{BD.Dec}, \mathsf{BD.Trace})$ be a bounded (secret/public tracing)-EITR scheme for identity space $\mathcal{ID} = \{0,1\}^\kappa$. We construct an unbounded (secret/public tracing)-EITR scheme as follows.

$\mathsf{Setup}(1^\lambda, 1^\kappa) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following:

1. For $j = 1$ to λ, sample $(\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, n_{\mathsf{bd}} = 2^j)$.
2. Output $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ and $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{lb}, \mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{lb},\mathsf{id}}$. The KeyGen algorithm does the following:

1. Parse $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$.
2. For $j = 1$ to λ, it computes $\mathsf{BD.sk}_j \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}, \mathsf{id})$.
3. Returns $\mathsf{sk}_{\mathsf{lb},\mathsf{id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, m, L) \rightarrow \mathsf{ct}$. The encryption algorithm does the following:

1. Parse $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$.

2. Secret share m in λ shares as follows. For $j = 1$ to $\lambda - 1$, randomly sample $r_j \leftarrow \mathcal{M}$ and set $r_\lambda = m \oplus r_1 \oplus \ldots \oplus r_{\lambda-1}$.

3. For $j = 1$ to λ, compute $\mathsf{BD.ct}_j = \mathsf{BD.Enc}(\mathsf{BD.mpk}_j, r_j, L)$.

4. Output $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j\in[\lambda]}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{lb,id}}, \mathsf{ct}, L) \to m'$. The decryption algorithm does the following:

1. Parse $\mathsf{sk}_{\mathsf{lb,id}} = \{\mathsf{BD.sk}_j\}_{j\in[\lambda]}$ and $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j\in[\lambda]}$.

2. For $j = 1$ to λ, compute $r'_j = \mathsf{BD.Dec}(\mathsf{BD.sk}_j, \mathsf{BD.ct}_j, L)$.

3. If any of the decryption fails then output $m' = \perp$, else output $m' = \bigoplus_{j\in[\lambda]} r'_j$.

$\mathsf{Trace}^D(\mathsf{tk}, y, \mathrm{Q}_{\mathsf{bd}}, m_0, m_1, L) \to T$. The trace algorithm uses two algorithms isGoodDecoder and SubTrace defined in Figures 3.13 and 3.14, respectively as subroutines. The tracing algorithm is as follows.

1. Parse tk as $\mathsf{msk} = \{\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j\}_{j\in[\lambda]}$ and let $\mathsf{BD.tk}_j = \mathsf{BD.msk}_j$ for $j \in [\lambda]$. (For the public trace setting $\mathsf{tk} = \mathsf{mpk}$ and $\mathsf{BD.tk}_j = \mathsf{BD.mpk}_j$)

2. Set $j = \lceil \log \mathrm{Q}_{\mathsf{bd}} \rceil$.

3. Set $\mathsf{flag} = 0$. For $\mathsf{itr} = 1$ to $\lambda \cdot y$, do the following

   a) Choose a random message $r \leftarrow \mathcal{M}$.

   b) Run isGoodDecoder as
   $\mathsf{flag} \leftarrow \mathsf{isGoodDecoder}^D(\{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$

   c) If $\mathsf{flag} = 1$, break. Else, continue.

4. If $\mathsf{flag} = 1$, run $T \leftarrow \mathsf{SubTrace}^D(\{\mathsf{BD.mpk}_j, \mathsf{BD.tk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, j)$. Else, set $T = \phi$.

5. Output T.

Correctness. We show that the above construction of bounded (secret/public tracing)-EITR satisfies correctness (Def. 3.22) via the following theorem.

Theorem 3.49. *Assume* BD-TR *is a correct bounded (secret/public tracing)-*EITR

Algorithm $\mathsf{isGoodDecoder}^{D}(\mathsf{key}, 1^{y}, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and a position-index $i \in [\lambda]$.

Output : 0/1.

1. Set $\mathsf{count} = 0$. Let $\epsilon = 1/y$.
2. For $j = 1$ to $\lambda \cdot y$:
   - Sample $\lambda - 1$ messages, $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_\lambda$ randomly such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.
   - Sample $b \leftarrow \{0,1\}$, and compute ciphertexts $\mathsf{BD.ct}_k = \mathsf{BD.Enc}(\mathsf{BD.mpk}_k, r_k, L)$ for $k \in [\lambda] \setminus \{i\}$ and $\mathsf{BD.ct}_i = \mathsf{BD.Enc}(\mathsf{BD.mpk}_i, r \oplus m_b, L)$.
   - Query D on $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_\lambda)$. Let b' be the response of D.
   - If $b' = b$, set $\mathsf{count} = \mathsf{count} + 1$.
3. If $\mathsf{count}/(\lambda \cdot y) \geq 1/2 + \epsilon/3$, then output 1, else output 0.

Figure 3.13: Algorithm isGoodDecoder for Unbounded EITR

Algorithm $\mathsf{SubTrace}^{D}(\mathsf{key}, 1^{y}, m_0, m_1, r, L, i)$

Inputs: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j, \mathsf{BD.tk}_j\}_{j\in[\lambda]}$, parameter y, messages m_0, m_1, r, revocation list L and a position-index $i \in [\lambda]$.

Output : $T \subseteq \{0,1\}^{\kappa}$.

1. Define oracle $\tilde{D}[\{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}, r, L, i]$ as in Figure 3.15.
2. Output $T \leftarrow \mathsf{BD.Trace}^{\tilde{D}}(\mathsf{BD.tk}_i, 4y, m_0 \oplus r, m_1 \oplus r, L)$.

Figure 3.14: Algorithm: SubTrace for Unbounded EITR

Algorithm $\tilde{D}^{D}[\mathsf{key}, r, L, i]$

Hardwired values: keys $\mathsf{key} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$, message r, revocation list L and a position-index $i \in [\lambda]$.
Inputs: $\mathsf{BD.ct}$.
Output : 0/1
On input $\mathsf{BD.ct}$, the $\tilde{D}$ oracle does the following:

- It first shares r in $\lambda - 1$ components as follows: it chooses $\lambda - 1$ random messages r_k for $k \in [\lambda] \setminus \{i\}$, such that $\oplus_{k\in[\lambda]\setminus\{i\}} r_k = r$.
- It computes $\mathsf{BD.ct}_k = \mathsf{BD.Enc}(\mathsf{BD.mpk}_k, r_k, L)$ for $k \in [\lambda] \setminus \{i\}$.
- Sets $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_{i-1}, \mathsf{BD.ct}, \mathsf{BD.ct}_{i+1}, \ldots, \mathsf{BD.ct}_\lambda)$.
- Outputs $b' \leftarrow D(\mathsf{ct})$.

Figure 3.15: Oracle $\tilde{D}$ for Unbounded EITR

*scheme then the above construction of unbounded (secret/public tracing)-*EITR *scheme is correct.*

Proof. For all $k \in [\lambda]$, if $\mathsf{BD.ct}_k \leftarrow \mathsf{BDEnc}(\mathsf{BD.mpk}_k, r_k, L)$ and $\mathsf{BD.sk}_k \leftarrow \mathsf{BDKeyGen}(\mathsf{BD.msk}_k, (\mathsf{lb}, \mathsf{id}))$, we have $r_k \leftarrow \mathsf{BD.Dec}(\mathsf{BD.sk}_k, \mathsf{BD.ct}_k, L)$, as long as $\mathsf{id} \notin L$, from the correctness of the underlying BD-TR scheme. Hence, the decryption of $\mathsf{ct} = (\mathsf{BD.ct}_1, \ldots, \mathsf{BD.ct}_\lambda)$ correctly outputs m as $r_1 \oplus \cdots \oplus r_\lambda$. ∎

Efficiency. We can instantiate the above construction by the bounded public/secret tracing-EITR scheme in Sec. 3.10. Since the above construction is simple λ times repetition of the underlying bounded EITR scheme, the parameter size of the scheme is as follows.

Secret Tracing Setting. In the secret tracing setting, we have $|\mathsf{mpk}|, |\mathsf{ct}|, |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

Public Tracing Setting. In the public tracing setting, we have $|\mathsf{mpk}|, |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathsf{lb}|), |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathsf{id}|, |\mathsf{lb}|)$.

3.11.2 Security

In this section, we prove that our construction of unbounded (secret/public tracing)-EITR scheme is secure.

IND-CPA security.

Theorem 3.50. *If the underlying* BD-TR *scheme is (very selective/adaptive)* ind-cpa *secure bounded (secret/public tracing)-*EITR *scheme, then the above construction of unbounded (secret/public tracing)-*EITR *is (very selective/adaptive)* ind-cpa *secure.*

Proof. We show that if there exists a PPT adversary that breaks the ind-cpa security of unbounded EITR scheme, then we can use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks ind-cpa security of the underlying BD-TR scheme. ■

The reduction is defined as follows:

1. $\mathcal{B}$ first gets $1^\kappa, L$ and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k)\}_{k\in[Q]}$, where Q is the number of key queries issued, from the adversary $\mathcal{A}$. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and output L adaptively along with the challenge messages).

2. It sets $n_{\mathsf{bd}} = 2$ and sends $1^\kappa, 1^{n_{\mathsf{bd}}}, L$ and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k)\}_{k\in[Q]}$ to the BD-TR challenger. The BD-TR challenger returns $\mathsf{BD.mpk}$ and $\{\mathsf{BD.sk}_k\}_{k\in[Q]}$.

3. $\mathcal{B}$ sets $\mathsf{BD.mpk}_1 = \mathsf{BD.mpk}$, generates $((\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, 2^j))$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

4. For each key query $(\mathsf{lb}_k, \mathsf{id}_k)$, $\mathcal{B}$ sets $\mathsf{BD.sk}_{k,1} = \mathsf{BD.sk}_k$, computes $\mathsf{BD.sk}_{k,j} \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}_k, \mathsf{id}_k)$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{sk}_{\mathsf{lb}_k,\mathsf{id}_k} = \{\mathsf{BD.sk}_{k,j}\}_{j\in\lambda}$ to $\mathcal{A}$.

5. When $\mathcal{A}$ sends the challenge query (m_0, m_1), $\mathcal{B}$ samples $r_j \leftarrow \mathcal{M}$ for $j \in \{2, \cdots, \lambda\}$, sets $m'_b = \bigoplus_{j>1} r_j \oplus m_b$ for $b \in \{0, 1\}$ and sends (m'_0, m'_1) as challenge query to the BD challenger and and sets the returned ciphertext as $\mathsf{BD.ct}_1$. $\mathcal{B}$ then computes $\mathsf{BD.ct}_j \leftarrow \mathsf{BD.Enc}(\mathsf{BD.mpk}_j, r_j, L)$ for $j \in \{2, \cdots, \lambda\}$ and sends $\mathsf{ct} = \{\mathsf{BD.ct}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

6. $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ sends b' to the BD-TR challenger.

We observe that $\mathcal{B}$ issues a key query $(\mathsf{lb}, \mathsf{id})$ to the BD-TR challenger only when $\mathcal{A}$ makes a key query on $(\mathsf{lb}, \mathsf{id})$. So , by the admissibility of $\mathcal{A}$, we have $\mathsf{lb} \in L$ and thus $\mathcal{B}$ is admissible in the ind-cpa game of BD-TR. Also, if $\mathcal{A}$ has an advantage ϵ in distinguishing the encryptions of m_0, m_1, then clearly $\mathcal{B}$ has the same advantage in distinguishing the encryptions of m'_0, m'_1 and thus breaking the ind-cpa security of BD-TR.

Secure Tracing Analysis. Now we show that our construction satisfies false trace and correct trace guarantees.

False Trace Guarantee.

Theorem 3.51. *Assume that the underlying (secret/public tracing)-*BD-TR *scheme satisfies (selective/adaptive) false trace guarantee, then our construction of unbounded (secret/public tracing)-*EITR *satisfies (selective/adaptive) false trace guarantee as defined in Def. 3.30/Def. 3.29.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$, polynomial $p(\lambda)$ and non-negligible functions $\epsilon(\cdot), \delta(\cdot)$ such that $\mathsf{Pr}[\mathsf{Fal\text{-}Tr}]_{\mathcal{A},\epsilon,p} \geq \delta(\lambda)$, then we can build a PPT reduction $\mathcal{B}$ that can break the false trace guarantee of BD-TR. The reduction is as follows:

1. $\mathcal{B}$ first gets $1^\kappa, L$ and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k)\}_{k\in[Q]}$, where Q is the number of key queries issued, from the adversary $\mathcal{A}$. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and output L adaptively along with the challenge messages).

2. It sets $i = \lceil \log p(\lambda) \rceil$, sends $1^\kappa, 1^{2^i}, L$ and and key queries $\{(\mathsf{lb}_k, \mathsf{id}_k)\}_{k\in[Q]}$ to the BD-TR challenger. The BD-TR challenger returns $\mathsf{BD.mpk}$ and $\{\mathsf{BD.sk}_k\}_{k\in[Q]}$.

3. $\mathcal{B}$ sets $\mathsf{BD.mpk}_i = \mathsf{BD.mpk}$, generates $((\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, 2^j))$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

4. For each key query $(\mathsf{lb}_k, \mathsf{id}_k)$, $\mathcal{B}$ sets $\mathsf{BD.sk}_{k,i} = \mathsf{BD.sk}_k$, computes

$\mathsf{BD.sk}_{k,j} \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}_k, \mathsf{id}_k)$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{sk}_{\mathsf{lb}_k,\mathsf{id}_k} = \{\mathsf{BD.sk}_{k,j}\}_{j\in\lambda}$ to $\mathcal{A}$.

5. Finally, when $\mathcal{A}$ outputs a decoder box D and messages m_0, m_1 , $\mathcal{B}$ runs $\mathsf{isGoodDecoder}((\mathsf{BD.mpk}_j)_{j\in[\lambda]}, 1^{1/\epsilon}, m_0, m_1, r, L, i)$ as defined in Figure 3.13, for $\lambda \cdot y$ many choices of r, until it finds an r s.t $\mathsf{isGoodDecoder}$ outputs 1.

6. $\mathcal{B}$ constructs a decoding box $\tilde{D}$ as defined in Figure 3.14, sends $(\tilde{D}, m_0 \oplus r, m_1 \oplus r)$ to the BD-TR challenger.

We observe that $\tilde{D}$ uses the decoder D as a subroutine and it returns the response of D as its output. So, if $\mathcal{A}$ outputs (D, m_0, m_1, L) such that the false trace guarantee does not hold with non-negligible probability, then $\mathcal{B}$ breaks the false trace guarantee of the underlying BD-TR scheme. ■

Correct Trace Guarantee.

Theorem 3.52. *Assume that the underlying (secret/public tracing)-*BD-TR *scheme satisfies (very selective/adaptive) correct trace guarantee, then our construction of unbounded (secret/public tracing)-*EITR *satisfies (very selective/adaptive) correct trace guarantee as defined in Def. 3.30/Def. 3.29.*

Proof. Let $i = \lceil \log p(\lambda) \rceil$ and let $S, S_{\mathcal{ID}}, T_{\mathsf{lb}}$ be as defined in Def. 3.29. Consider the following events

Event $\mathsf{Cor\text{-}Tr}_i$: is defined as the event that the $\mathsf{SubTrace}$ algorithm, when run for position i outputs a correct traitor set T, i.e. $|T| > 0, (T \subseteq S_{\mathcal{ID}}) \wedge (T_{\mathsf{lb}} \cap L = \phi)$.

Event $\mathsf{Good\text{-}\tilde{D}}_i$: is defined as the event that the flag is set to 1 in step 3 and the decoder $\tilde{D}$ defined in Fig 3.15 is $\epsilon/4$ good decoder.

Event $\mathsf{Found\text{-}Good\text{-}r}_i$: is defined as the event that the $\mathsf{isGoodDecoder}$ algorithm, when run for position i outputs 1.

With these definitions, the correctness is argued via following series of inequalities:

$$\begin{aligned}
\mathsf{Pr\text{-}Corr\text{-}Tr}(\lambda) &= \mathsf{Pr}[\mathsf{Corr\text{-}Tr}_i] && (3.19)\\
&\geq \mathsf{Pr}[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i] && (3.20)\\
&\geq \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (3.21)\\
&\geq \mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (3.22)\\
&\geq \mathsf{Pr}[\mathsf{Found\text{-}Good\text{-}r}_i \wedge \mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (3.23)\\
&\geq \mathsf{Pr}[\mathsf{Good\text{-}Decoder} \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{negl}(\lambda) && (3.24)
\end{aligned}$$

Equation (3.19) and (3.20) follow directly from the definition of the events involved. Equation (3.21) follows from the following claim:

Claim 3.53. If BD-TR guarantees correct tracing then

$$\mathsf{Pr}\left[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|\right] - \mathsf{Pr}[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i] \leq \mathsf{negl}(\lambda).$$

Proof. We show that if there exists an adversary $\mathcal{A}$ who outputs a good decoder along with messages m_0, m_1 and a revocation list L such that $\mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{Pr}[\mathsf{Cor\text{-}Tr}_i]$ is non negligible then we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against correct trace guarantee of the underlying BD-TR. The reduction is defined as follows:

It sets $i = \lceil \log p(\lambda) \rceil$,

1. $\mathcal{B}$ first gets $1^\kappa, L$ and key queries $\{(\mathsf{Ib}_k, \mathsf{id}_k)\}_{k\in[Q]}$, where Q is the number of key queries issued, from the adversary $\mathcal{A}$. (In the public trace setting, $\mathcal{A}$ can make adaptive key queries and output L adaptively along with the challenge messages).

2. It sends $1^\kappa, 1^{2^i}, L$ and and key queries $\{(\mathsf{Ib}_k, \mathsf{id}_k)\}_{k\in[Q]}$ to the BD-TR challenger. The BD-TR challenger returns BD.mpk and $\{\mathsf{BD.sk}_k\}_{k\in[Q]}$.

3. $\mathcal{B}$ sets $\mathsf{BD.mpk}_i = \mathsf{BD.mpk}$, generates $((\mathsf{BD.mpk}_j, \mathsf{BD.msk}_j) \leftarrow \mathsf{BD.Setup}(1^\lambda, 1^\kappa, 2^j))$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{mpk} = \{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}$ to $\mathcal{A}$.

4. For each key query $(\mathsf{lb}_k, \mathsf{id}_k)$, $\mathcal{B}$ sets $\mathsf{BD.sk}_{k,i} = \mathsf{BD.sk}_k$, computes $\mathsf{BD.sk}_{k,j} \leftarrow \mathsf{BD.KeyGen}(\mathsf{BD.msk}_j, \mathsf{lb}_k, \mathsf{id}_k)$ for $j \in [\lambda] \setminus \{i\}$ and sends $\mathsf{sk}_{\mathsf{lb}_k,\mathsf{id}_k} = \{\mathsf{BD.sk}_{k,j}\}_{j\in\lambda}$ to $\mathcal{A}$.

5. In the end, $\mathcal{A}$ outputs a decoder D along with messages m_0, m_1.

6. $\mathcal{B}$ does the following:

   - If $Q > p(\lambda)$, then $\mathcal{B}$ outputs a random decoder and quits.
   - Else, $\mathcal{B}$ runs $\mathsf{isGoodDecoder}(\{\mathsf{BD.mpk}_j\}_{j\in[\lambda]}, 1^y, m_0, m_1, r, L, i)$ for uniformly and independently sampled r until it finds a r for which $\mathsf{isGoodDecoder}$ algorithm outputs 1. If $\mathsf{isGoodDecoder}$ algorithm does not output 1 even after $\lambda \cdot y$ attempts, then $\mathcal{B}$ outputs a random decoder and quits.
   - Else, $\mathcal{B}$ constructs decoder $\tilde{D}$ as defined in Figure 3.15 and sets $\tilde{m}_b = m_b \oplus r$ for $b \in \{0, 1\}$.

7. $\mathcal{B}$ sends $(\tilde{D}, \tilde{m}_0, \tilde{m}_1)$ to the $\mathsf{BD\text{-}TR}$ challenger.

Now we analyze the probability that $\mathcal{B}$ outputs a good decoder. Observe that $\mathcal{B}$ does not abort and outputs a genuine decoder when both $\mathsf{Found\text{-}Good\text{-}r}_i$ and $|S_{ID}| \leq p(\lambda)$ happens. Since the decoder returned by $\mathcal{B}$ is the same decoder as defined in the $\mathsf{SubTrace}$ algorithm, we get that the probability that $\mathcal{B}$ outputs a good decoder is $\mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|]$. Furthermore, the probability that the $\mathsf{BD.Trace}$ algorithm outputs correct set of traitors T (using the decoder returned by $\mathcal{B}$) is same as $\mathsf{Pr}[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i]$. Hence, if $\mathsf{Pr}[\mathsf{Good\text{-}\tilde{D}}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i \wedge p(\lambda) \geq |S_{ID}|] - \mathsf{Pr}[\mathsf{Corr\text{-}Tr}_i \wedge \mathsf{Found\text{-}Good\text{-}r}_i]$ is non negligible, then it breaks the security of correct trace guarantee of the underlying $\mathsf{BD\text{-}TR}$ scheme. ■

Equation (3.22) again follows directly from the definition. Argument for transition from equation (3.22) to (3.23) and (3.23) to (3.24) is same as that of transition from (3.16) to (3.17) and (3.17) to (3.18), respectively. This completes the proof. ■

3.12 EXTENSION TO SUPER-POLYNOMIAL SIZE REVOCATION LIST

While our main focus in the paper is on the case where the revocation list L is of polynomial size, we can consider an extension where it is of super-polynomial size. In particular, we consider the setting where L has efficient representation by a circuit C_L of polynomial size. Namely, we have $C_L(\mathsf{lb}) = 0$ for revoked label lb and $C_L(\mathsf{lb}) = 1$ for non-revoked label lb. We assume that the depth of C_L is bounded by some polynomial $\bar{d}$. Namely, $C_L \in \mathcal{C}_{|\mathsf{lb}|,\bar{d}}$. This assumption is necessary because we will use kpABE (resp., cpABE) with the same restriction to generate a secret key (resp., a ciphertext) associated with C_L.

EITR scheme for this setting can be obtained both in the secret and the public tracing settings very similarly to the case where the revocation list is of polynomial size. Namely, we first construct (secret key/public key) RPE with super-polynomial revocation list and then apply the chain of conversions (Sec. 3.9, 3.10, and 3.11). The conversions work almost without change with the natural adaptation of replacing L with C_L and the membership check $\mathsf{lb} \stackrel{?}{\in} L$ with $C(\mathsf{lb}) \stackrel{?}{\in} 0$. The constructions of (pubic key/secret key) RPE are also almost the same as those in (Sec. 3.5/Sec. 3.6 and 3.7) with the natural adaptation of generating kpABE secret key for C_L[27] in Sec. 3.5 and 3.6 and replacing the condition $\mathsf{lb} \notin L$ in Eq. (3.4) defining $C_{L,\mathsf{RMFE.ct}}$ with $C_L(\mathsf{lb}) = 1$. The main difference is that we have to assume sub-eponential LWE assumption instead of (polynomial) LWE assumption for both secret key and public key settings here, because we need adaptive security for the underlying kpABE. We give further details in the following.

- In the public key setting, indistinguishability of $\mathsf{Hyb}_{6,a}$ and Hyb_7 shown in Claim 3.17, where post-challenge key queries are dealt with, cannot be proven any more if we only assume selective security for kpABE. The reason why the original proof does not go through is that we have to deal with the kpABE queries in the order of key first and ciphertext later. With polynomial size L, this does not pose a problem because when the adversary chooses L, all the labels for which we use kpABE security are in L and we can perform a hybrid argument over these labels. However, this is not possible for super-polynomial size L. To deal with the queries

[27] Originally, we start from the revocation list L and then construct the circuit C_L that hardwires L into it. Here, we directly use the circuit C_L that efficiently represents the super-polynomial set L.

in this order, we assume adaptive security (as per Definition 2.9) for kpABE. Then, the indistinguishability of the games can easily be shown by changing the post-challenge kpABE ciphertexts associated with lb with $C_L(\mathsf{lb}) = 0$ one by one.

- In the secret key setting, we also encounter similar difficulty. In particular, indistinguishability of Hyb_1 and Hyb_2 shown in Claim 3.30 cannot be proven any more if we only assume selective security for kpABE by exactly the same reason. We can overcome the problem by assuming adaptive security for kpABE.

Finally, we briefly discuss the parameter size of the resulting EITR scheme. The parameter size of the resulting scheme is the same as the case of polynomial size revocation list except that they rely on $\bar{d}$. Notably, they are independent from the size of the circuits being supported inheriting the succinctness properties of the underlying kpABE and cpABE.

CHAPTER 4

ATTRIBUTE BASED ENCRYPTION FOR TURING MACHINES FROM LATTICES

4.1 INTRODUCTION

Attribute based encryption (ABE) [157, 116] is a fundamental primitive in cryptography that enables fine-grained access control on encrypted data. In an ABE scheme, the ciphertext encodes a secret message m and a public *attribute* $\mathbf{x}$, the secret key encodes a (public) function f, and decryption outputs m if and only if $f(\mathbf{x}) = 1$. Security posits that an adversary cannot distinguish between an encryption of $(m_0, \mathbf{x})$ and $(m_1, \mathbf{x})$ given secret keys that do not decrypt the challenge. ABE has two variants – "key-policy" where the function f (typically representing an access control policy) is embedded in the key, or "ciphertext-policy" where it is embedded in the ciphertext, as the names suggest. These are denoted by KP-ABE and CP-ABE respectively.

Traditionally, the function f is represented by a circuit and while there has been fantastic progress in building ABE supporting general circuits [116, 107, 43, 26, 169, 122], there are inherent limitations to the circuit model. Circuits force the size of the input to be fixed ahead of time and also incur *worst case* running time on every input – this is dissatisfying from both the theoretical and practical perspective. To overcome these limitations, a line of works [165, 102, 32, 15, 136, 103, 104, 17, 18, 142] has studied ABE for uniform models of computation but success has been much more limited. Without relying on heavy hammers such as indistinguishability obfuscation or compact functional encryption, the state of art ABE in this regime[1] supports non-deterministic log space Turing machines from pairings [142]. In the post-quantum regime, the situation is even less satisfactory – the state of the art construction supports non-deterministic finite state

[1] In this work, we only consider ABE with unbounded collusion resistance.

automata from learning with errors (LWE) but only in the *symmetric* key setting [17]. In the public key setting, even ABE for DFA – supporting unbounded lengths and with provable security – is not known, to the best of our knowledge. Thus, an outstanding open question is:

Can we extend ABE for uniform computation beyond NL?

4.1.1 Our Results

In this work, we take a leap forward and provide the first ABE scheme for Turing machines supporting unbounded collusions from lattice assumptions. In more detail, the encryptor encodes an attribute $\mathbf{x}$ together with a bound t on the machine running time and a message m into the ciphertext, the key generator embeds a Turing machine M into the secret key and decryption returns m if and only if $M(\mathbf{x}) = 1$. Crucially, the input $\mathbf{x}$ and machine M can be of unbounded size, the time bound t can be chosen dynamically for each input and decryption runs in input specific time.

In more detail, our results are:

1. We construct the first ABE for NL from the LWE, evasive LWE [169, 163] and Tensor LWE [169] assumptions. This yields the first (conjectured) post-quantum ABE for NL.

2. Relying on LWE, evasive LWE and a new assumption called *circular tensor* LWE, we construct ABE for all Turing machines. At a high level, the circular tensor LWE assumption incorporates circularity into the tensor LWE (Wee, Eurocrypt 2022) assumption.

Towards our ABE for Turing machines, we obtain the first CP-ABE for circuits of unbounded depth and size from the same assumptions – this may be of independent interest.

Our Assumptions. Below, we describe the evasive and circular tensor LWE assumptions. Below, we adopt the convention by Wee [169] and let the underline denote

that noise is added to the term, whose exact value is not important.

Evasive LWE. The evasive LWE assumption [169] states that if

$$\begin{aligned} \text{if } & \mathbf{B},\ \mathbf{A},\ \mathbf{P},\ \underline{\mathbf{s}^\intercal\mathbf{B}},\ \underline{\mathbf{s}^\intercal\mathbf{A}},\ \underline{\mathbf{s}^\intercal\mathbf{P}},\ \mathsf{aux} && \approx \mathbf{B},\ \mathbf{A},\ \mathbf{P},\ \$,\ \$,\ \$,\ \mathsf{aux} \\ \text{then } & \mathbf{B},\ \mathbf{A},\ \mathbf{P},\ \underline{\mathbf{s}^\intercal\mathbf{B}},\ \underline{\mathbf{s}^\intercal\mathbf{A}},\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux} && \approx \mathbf{B},\ \mathbf{A},\ \mathbf{P},\ \$,\ \$,\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux} \end{aligned}$$

Above $\mathbf{B}^{-1}(\mathbf{P})$ is a low norm Gaussian matrix such that $\mathbf{B}\ \mathbf{B}^{-1}(\mathbf{P}) = \mathbf{P}$, and $\mathbf{A}, \mathbf{P}$ are sampled in a correlated way. In the public coin version of this assumption, the auxiliary information aux above contains random coins using during the sampling. We rely on the private coin version of this assumption, similarly to prior work [22, 164].

Tensor Circular LWE. Similar to how circularity was incorporated into the evasive LWE assumption by [122], we will need to incorporate it in the tensor LWE assumption introduced by Wee [169]. The tensor LWE assumption states that for all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, we have

$$①\colon \mathbf{A}, \underline{\left\{(\mathbf{s}(\mathbf{I}\otimes\mathbf{r}_i))^\intercal(\mathbf{A} - \mathbf{x}_i\otimes\mathbf{G}), \mathbf{r}_i\right\}_{i\in[Q]}} \approx_c \ ②\colon\ \mathbf{A}, \{\$, \mathbf{r}_i\}_{i\in[Q]}$$

In the original formulation $\mathbf{s}$ is uniform and $\mathbf{r}$ is sampled from a discrete Gaussian. We will require $\mathbf{s}$ to be small, i.e. also sampled from a discrete Gaussian.

Our *tensor circular* LWE *assumption* basically incorporates the circular terms into the assumption. For notational brevity, we denote $\mathbf{s}(\mathbf{I}\otimes\mathbf{r}_i)$ by $\mathbf{s}_{\mathbf{r}_i}$. In more detail, for all $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \{0,1\}^L$,

$$\mathbf{A}_{\mathsf{circ}}, \left\{①,\ \ \mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}},\ \mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}},\ \underline{\mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}})\otimes\mathbf{G})},\ \mathbf{r}_i\right\}_{i\in[Q]} \approx_c \mathbf{A}_{\mathsf{circ}}, \left\{②,\ \ \$,\ \$,\ \$,\ \mathbf{r}_i\right\}_{i\in[Q]} \tag{4.1}$$

Above $\mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}}$ is the FHE public key corresponding to secret $\mathbf{s}_{\mathbf{r}_i}$, and $\mathbf{S}_{\mathbf{s}_{\mathbf{r}_i}} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathbf{s}_{\mathbf{r}_i})$ is an FHE ciphertext.

The rationale of security for tensor LWE does not change with the addition of the circular terms, to the best of our knowledge. While we do not see any attack on this strengthening of tensor LWE, we note that the usage of tensor LWE in our construction is inherited from the construction by Wee [169]. Any improvements to Wee's construction are likely to lead to improvements in our setting as well.

4.1.2 Technical Overview

Next, we outline the main technical ideas developed in our work.

Using KP-ABE and CP-ABE for circuits to build ABE for TM. The starting point of our work is the compiler by Agrawal et al. [16] that shows how to construct public key functional encryption (FE) for Turing machines using two circuit FE schemes in tandem, one ciphertext-policy and one key-policy. In more detail, their construction, building upon techniques developed in [17], relies on two restricted FE schemes: one that supports decryption in the case $|(\mathbf{x}, 1^t)| > |M|$ and one that supports the case where $|(\mathbf{x}, 1^t)| \leq |M|$. Here, $\mathbf{x}$ is the input chosen by the encryptor and is of unbounded length, M is the Turing machine chosen by the key generator and t is an upper bound on the runtime of the Turing machine on a given input $\mathbf{x}$. We note that t can be chosen by the encryptor dynamically for each x and is not a-priori bounded. These restricted schemes are then run in parallel so that the first scheme can be used to decrypt the ciphertext when $|(\mathbf{x}, 1^t)| > |M|$ and the second scheme can be used otherwise.

It was shown by [16] that the first sub-scheme can be instantiated using a CP-FE that supports unbounded size and unbounded depth circuits while the second can be instantiated using a KP-FE for unbounded size but bounded depth circuits. If the first scheme only supports bounded depth (but still supports unbounded size), this still yields an FE for a smaller class of computation NL. While the techniques of [16] were developed for the setting of bounded key FE, they can be adapted to the setting of unbounded key ABE as well. We skip the details here and refer the reader to Section 4.5 for the detailed

compiler. Since KP-ABE for unbounded size but bounded depth circuits has been known since 2013 [107, 43], it remains to find a CP-ABE for unbounded size, unbounded depth circuits to instantiate the compiler.

CP-ABE for Unbounded Size Circuits. Traditionally ciphertext-policy schemes have been much harder to construct than their key-policy cousins – for instance, while KP-ABE for circuits has been known for over a decade from the standard LWE assumption, it was only very recently that the first CP-ABE for circuits was discovered [169], and that too by relying on new lattice assumptions called the *evasive* and *tensor* LWE assumptions. Moreover, of these, the evasive LWE is actually a fairly strong, non-falsifiable assumption. Yet, these assumptions have generated significant excitement in the community since they enabled progress on problems that had remained "stuck" for a while.

Wee's construction supports circuits of unbounded size but only bounded depth. While this will turn out to be a significant barrier, this construction is all that we have to begin with. Let us recap this construction below and then try to adapt it to suit our needs.

Overview of Wee's CP-ABE. Wee's construction can be seen as broadly following a two step outline: (i) start with a CP-ABE which satisfies only single key security, (ii) add collusion resistance via randomization of keys. Indeed, this two-step recipe for CP-ABE has been used before [27, 64], where the first step is common to all works (barring minor syntactic differences) and can be based on LWE, while the second step has been implemented using pairings in [27] and using heuristic lattice methods in [64]. Wee's construction achieves randomization of keys via *tensors*.

In more detail, from prior work on lattice based KP-ABE [43], it is known that given an input $\mathbf{x} \in \{0,1\}^\ell$, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on an "input encoding" matrix of form $\mathbf{A} - \mathbf{x} \otimes \mathbf{G}$ by multiplying

on the right by a low norm matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ to obtain the term $\mathbf{A}_f - f(x)\mathbf{G}$. Here, $\mathbf{G}$ is a special gadget matrix defined as follows. Let $\mathbf{g} = [1, 2, 2^2, \ldots, 2^{\log q}]^\intercal$ and $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^\intercal$.

Towards CP-ABE, let the public key be the matrix $\mathbf{A}$ and the ciphertext for function f and message μ be of the form $\underline{\mathbf{s}^\intercal\mathbf{A}_f + \mu\lceil q/2\rceil}$ for some randomly sampled LWE secret $\mathbf{s}$. To decrypt correctly, we would ideally want the secret key encoding attribute $\mathbf{x}$ to be $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ – this would let us recover the mask $\mathbf{s}^\intercal\mathbf{A}_f$ via homomorphic computation and subtract it from the ciphertext to recover the message (after rounding out a suitably bounded noise).

Of course, the key generator cannot know the encryption randomness $\mathbf{s}$, and indeed this randomness will change for every ciphertext (among an unbounded number) so the above does not work [2]. At this stage, to enable key generation, the evasive LWE assumption described above comes to our rescue. Loosely speaking, the evasive LWE assumption can be seen as a kind of "secret sharing" mechanism to generate our desired term $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ when the encryptor holds randomness $\mathbf{s}$ and the key generator holds attribute $\mathbf{x}$. The encryptor will now additionally provide an extra LWE sample $\underline{\mathbf{s}^\intercal\mathbf{B}}$ under a public matrix $\mathbf{B}$ and the key generator will provide a low norm Gaussian matrix $\mathbf{R}$ such that $\mathbf{B}\,\mathbf{R} = (\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$. For readability, $\mathbf{R}$ is referred to as $\mathbf{B}^{-1}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$ in the literature. Now, the decryptor can compute:

$$(\underline{\mathbf{s}^\intercal\mathbf{B}})\mathbf{B}^{-1}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) = \underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$$

and decryption can proceed as desired. However, while we achieve correctness, the above scheme is not secure. It is easy to see that an attacker, given keys for any $\mathbf{x}$ and $\bar{\mathbf{x}}$ can trivially break security. To achieve collusion resistance, it is necessary to randomize keys so that mix and match attacks are thwarted.

[2] A simple twist to this scheme yields a construction satisfying single-key security though by [156]

Randomizing keys. Wee [169] observed that the correctness of the evaluation algorithm is preserved even under tensoring with random Gaussian vectors $\mathbf{r}$ – in more detail, one can evaluate f even on the randomized $(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}$ to recover $(\mathbf{A}_f - f(x)\mathbf{G}) \otimes \mathbf{r}$ by simply augmenting the low norm multiplication matrix to $\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \otimes \mathbf{I}$. This is just a direct application of the mixed product property on tensors.

Armed with this technique, a modification of the above construction that is not immediately broken is:

$$\mathsf{ct}_f = \underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{I})} + \mu \cdot \mathbf{g}, \;\; \underline{\mathbf{s}^\mathsf{T}\mathbf{B}},$$
$$\mathsf{sk}_\mathbf{x} = \mathbf{B}^{-1}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}), \;\; \mathbf{r}$$

To decrypt, we compute

$$(\underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{I})} + \mu \cdot \mathbf{g})(\mathbf{I} \otimes \mathbf{r}) \approx \mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{r}) + \mu\mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r})$$
$$\underline{\mathbf{s}^\mathsf{T}\mathbf{B}}(\mathbf{B}^{-1}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r})) \cdot (\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \otimes \mathbf{I}) \approx \mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{r})$$

and subtract the second from the first to recover μ. Now, for different keys, the random vectors $\mathbf{r}_i$ are different and mix and match attacks no longer apply.

The above scheme needs to be refined further to admit a proof, but we skip the details here for ease of exposition. We only remark that to support arbitrary (bounded) depth, Wee needs to make an additional assumption called *tensor* LWE as discussed above.

Overcoming bounded depth. Wee's construction only supports circuits of bounded depth, primarily because the evaluation algorithms used on the ciphertext and public key only support bounded depth. In this context, the recent exciting work by Hsieh et al. [122] provides hope since they adapted the public key and ciphertext evaluation algorithms to support circuits of unbounded depth, and provided the first KP-ABE for circuits from a generalization of the evasive LWE assumption, called the *circular small*

secret evasive LWE assumption.

One may hope that plugging in the algorithms of [122] (henceforth HLL) into the CP-ABE construction of [169] can give a CP-ABE of unbounded depth, but as we shall see this approach runs into multiple difficulties. While this approach is what we will finally make work, it is via several new ideas, new assumptions and an involved security proof. We proceed to outline these next.

Unbounded Evaluation by HLL. The work of HLL supports unbounded homomorphism via two steps: (i) noise removal and (ii) bootstrapping. Suppose we evaluate on the lattice encoding $\underline{\mathbf{s}^\mathsf{T}(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$ as described previously and obtain $\mathbf{s}^\mathsf{T}(\mathbf{A}_f - f(\mathbf{x})\mathbf{G}) + \mathbf{e}_{\mathsf{large}}$ where $\mathbf{e}_{\mathsf{large}}$ is large noise, as the name suggests. We would like to reduce this noise, so that we can continue homomorphic evaluation. HLL provide a noise removal procedure inspired by the modulus reduction technique developed in the context of levelled FHE [61, 55], which allows to perform public operations and recover a *noiseless* encoding of the desired function $f(\mathbf{x})$. Unfortunately, the recovered encoding does not have the right structure and disallows further homomorphic computation. In more detail, their noise removal procedure allows to obtain a term

$$\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}) - f(\mathbf{x})\ \mathbf{s}^\mathsf{T}\mathbf{G}$$

where the second summand is what we want but the first is an ill-formed mask that is not amenable to homomorphism. The exact details of what this term looks like and how one obtains it are not too important here, and we will defer these to the technical sections. However, having reached this point, the obvious question is whether and how one can proceed to obtain a well formed encoding of $f(\mathbf{x})$ which has noise within the desired bound and permits further computation.

Bootstrapping or Structure Restoration. HLL provide a clever way out of the predicament by using circular security of LWE based FHE. At a high level, circular security posits that an FHE ciphertext remains pseudorandom even if it encrypts its own secret key. HLL first observe that if there was a way to compute $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$ then this could be combined with the term obtained above to recover $\underline{(\mathbf{s}^\mathsf{T}\mathbf{A}'_f - f(\mathbf{x})\ \mathbf{s}^\mathsf{T}\mathbf{G})}$ which has the right shape and allows to continue homomorphic evaluation. Thus, it suffices to compute the term $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$.

To do so, they leverage the algebraic structure of the GSW FHE scheme [99] together with an elegant "automatic decryption" technique by Brakerski et al. [60]. In more detail, recall that in the GSW FHE scheme, the secret key is $\mathbf{s}^\mathsf{T}$, a ciphertext for message $\mathbf{y}^\mathsf{T}$ is a matrix $\mathbf{C}$ and decryption computes $\mathbf{s}^\mathsf{T}\mathbf{C}$ to recover $\underline{\mathbf{y}^\mathsf{T}}$. The work of [60] proposes to use the same secret $\mathbf{s}$ for encoding the attribute, and set the attribute to be an FHE encryption under $\mathbf{s}$. As we will see, this causes "automatic decryption" which will be useful to recover the term we are after. In more detail, HLL additionally provide the following elements in their ciphertext:

$$\mathbf{S} = \mathsf{hct}_\mathbf{s}(\mathbf{s}), \quad \underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})} \tag{4.2}$$

Given this, one can evaluate the circuit $\mathsf{RndPad}_f(\cdot)$ homomorphically on the second term to obtain $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f} - \mathbf{s}^\mathsf{T}\mathsf{hct}(\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}))$ for some matrix $\mathbf{A}'_f$. Next, given the structure of GSW ciphertext, $\mathbf{s}^\mathsf{T}\mathsf{hct}(\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}))$ decrypts *automatically* to yield $\mathsf{RndPad}_f(\mathbf{s})$ giving the overall term $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}'_f - \mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s})}$ as desired.

Randomizing Unbounded Homomorphic Evaluation. Let us now try to plug in the unbounded evaluation procedures of HLL into Wee's construction. As described above, decryption in Wee's CP-ABE recovers the terms

$$\underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_f \otimes \mathbf{r})} + \mu\mathbf{g} \cdot (\mathbf{I} \otimes \mathbf{r}), \quad \underline{\mathbf{s}^\mathsf{T}((\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r})}$$

By the mixed product property of tensors, this can be written as

$$\underline{\mathbf{s}^\mathsf{T}(\mathbf{I}\otimes\mathbf{r})\mathbf{A}_f + \mu\mathbf{g}\cdot(\mathbf{I}\otimes\mathbf{r})}, \quad \underline{\mathbf{s}^\mathsf{T}(\mathbf{I}\otimes\mathbf{r})(\mathbf{A}-\mathbf{x}\otimes\mathbf{G})}$$

Denoting $\mathbf{s}^\mathsf{T}(\mathbf{I}\otimes\mathbf{r})$ as $\mathbf{s}_\mathbf{r}^\mathsf{T}$, we obtain well formed ABE encodings

$$\underline{\mathbf{s}_\mathbf{r}^\mathsf{T}\mathbf{A}_f + \mu\mathbf{g}\cdot(\mathbf{I}\otimes\mathbf{r})}, \quad \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}-\mathbf{x}\otimes\mathbf{G})}$$

In particular, we can perform the bounded depth evaluation of [43] as done by Wee [169] to obtain $\underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_f - f(\mathbf{x})\mathbf{G}))}$ and plug this into the noise removal procedure of HLL. Taking appropriate care, this works out to yield

$$\underline{\mathbf{s}_\mathbf{r}{}^\mathsf{T}(\mathsf{RndPad}_{\mathbf{A}_f}(\mathbf{s}_\mathbf{r}) - f(\mathbf{x})\mathbf{G})}$$

which appears suitable for further processing. To continue further, HLL requires the terms

$$\mathbf{S}_\mathbf{r} = \mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r}), \quad \mathbf{E}_\mathbf{r} = \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_\mathbf{r}\otimes\mathbf{G})} \tag{4.3}$$

Above, the subscript in hct denotes the secret key under which the ciphertext can be decrypted. Thus, by HLL, it suffices to compute the above terms so as to bootstrap and continue.

Randomized Bootstrapping. Unfortunately, careful examination shows how demanding the above requirement is. Note that $\mathbf{r}$ is chosen by the key generator while $\mathbf{s}$ (and hence $\mathbf{S} = \mathsf{hct}_\mathbf{s}(\mathbf{s})$) by the encryptor. To randomize $\mathbf{S}$ to $\mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r})$ might seem intuitively possible at first – after all, this is an FHE ciphertext and can be evaluated upon using knowledge of $\mathbf{r}$ to obtain $\mathsf{hct}_\mathbf{s}(\mathbf{s}_\mathbf{r})$. However, while modifying the underlying message is indeed easy, modifying the underlying *secret key* is anything but! In its full generality, modifying the secret key of an FHE scheme should not even be possible since it violates semantic security. Yet, there are amazing ways already known to modify the underlying secret key in FHE schemes – key shrinking in single key FHE [61, 55] as well as key *expansion* in multi-key FHE [76, 149]. Sadly, neither approach (nor anything

related) works for us since they require providing some "advice" terms that cannot be computed in our setting. Note that our situation is additionally complicated by the fact that we are also required to provide a circular encoding of $\mathbf{S_r}$ under the secret $\mathbf{s_r}$.

Another difficulty is that if we wish to randomize the circular encoding $\underline{\mathbf{s}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})}$ using tensors as discussed earlier, our key generator would be required to give a term that looks like $\mathbf{B}^{-1}((\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G}) \otimes \mathbf{r})$, which it cannot because unlike before, the inner quantity is not fixed and public but depends on $\mathbf{S}$ which is only known to the encryptor.

Our Approach. We overcome both hurdles together by relying on the following obvious fact, which previously seemed like a barrier – FHE ciphertexts are good for computing on the underlying messages, not on the underlying keys. We observe that nothing forces us to provide an FHE ciphertext of $\mathbf{s}$ under the secret key $\mathbf{s}$ itself! We can have our encryptor sample a new FHE scheme with unrelated secret, $\mathbf{t}$ (say) and provide an FHE encryption of $\mathbf{s}$ under the public key corresponding to $\mathbf{t}$. Now, given knowledge of $\mathbf{r}$, an evaluator can easily compute any complicated circuit, including those necessary to compute $\mathbf{S_r}$ and $\mathbf{E_r}$ described in Eq. 1. We can also provide an ABE encoding using the same trick of reusing $\mathbf{t}$ as the randomness to cause automatic decryption as was done in HLL.

In more detail, the encryptor can provide

$$\mathbf{T} = \mathsf{hct}_{\mathbf{t}}(\mathbf{s}, \mathsf{sd}), \quad \mathbf{D} = \underline{\mathbf{t}^\mathsf{T}(\mathbf{A}_1 - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G})}$$

where sd is a PRF seed, and $\mathbf{A}_1$ is a public matrix of appropriate dimensions. Now, one can homomorphically evaluate on the encoding $\mathbf{D}$ in *bounded* depth, using knowledge of $\mathbf{T}$, to obtain

$$\underline{\mathbf{t}^\mathsf{T}\mathbf{A}'_{\mathbf{r}} + \mathbf{t}^\mathsf{T}\mathsf{hct}_{\mathbf{t}}(\mathbf{S_r}, \mathbf{E_r})} = \underline{\mathbf{t}^\mathsf{T}\mathbf{A}'_{\mathbf{r}} + (\mathbf{S_r}, \mathbf{E_r})}$$

where $\mathbf{A}'_{\mathbf{r}}$ is some $\mathbf{r}$ dependent matrix and the equality follows by automatic decryption.

Recall that:

$$\mathbf{S_r} = \mathsf{hct}(\mathbf{A}_{\mathbf{s_r}}, \mathbf{s_r}), \quad \mathbf{E_r} = \underline{\mathbf{s}_\mathbf{r}^\top (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S_r})) \otimes \mathbf{G})}$$

which we require to plug into the HLL bootstrapping procedure.

Now, our desired output is masked by $\mathbf{t}^\top \mathbf{A}'_\mathbf{r}$ so it is unclear we have achieved anything. However, note that we now have the happy situation that $\mathbf{t}$ is known to the encryptor and $\mathbf{A}'_\mathbf{r}$ can be computed by the key generator! So we can plug back evasive LWE so that the encryptor gives $\underline{\mathbf{t}^\top \mathbf{C}}$ for some fixed matrix $\mathbf{C}$ and the key generator gives $\mathbf{C}^{-1}(\mathbf{A}'_\mathbf{r})$ so that we can recover the term $\underline{\mathbf{t}^\top \mathbf{A}'_\mathbf{r}}$ and cancel it out. Please see Section 4.3 for the detailed description.

Onward to Unbounded CP-ABE. The core idea for randomizing the bootstrapping discussed above applied carefully to Wee's construction allows us to obtain a correct scheme supporting unbounded depth circuits. However, proving security must still contend with several hurdles. While the high level structure of our proof resembles the proof of Wee's CP-ABE, our distributions are significantly more complex and need more work to analyze. To argue indistnguishability of hybrids, we must rely on the new assumptions described previously. We refer the reader to Section 4.4 for a detailed description of the assumptions, the scheme and the proof.

4.2 PRELIMINARIES

4.2.1 Garbled Circuits

Our definition of garbled circuits is based upon the one considered in [110]. For $\lambda \in \mathbb{N}$, let C_{inp} denote a family of circuits with inp bit inputs and $C = \{C_{\mathsf{inp}(\lambda)}\}_{\lambda \in \mathbb{N}}$. A garbling scheme for C consists of two algorithms $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ with the following syntax.

$\mathsf{Garble}(1^\lambda, C) \to \{\mathsf{lab}_{i,b}\}_{i \in [\mathsf{inp}], b \in \{0,1\}}$. The garbling algorithm takes as input the security parameter λ and a circuit $C \in C_{\mathsf{inp}(\lambda)}$. It outputs a set of labels

$\{\mathsf{lab}_{i,b}\}_{i\in[\mathsf{inp}],b\in\{0,1\}}$.

$\mathsf{Eval}(\{\mathsf{lab}_i\}_{i\in[\mathsf{inp}]}) \rightarrow y$. The evaluation algorithm takes as input an inp labels $\{\mathsf{lab}_i\}_{i\in[\mathsf{inp}]}$ and outputs y.

Definition 4.1 (Correctness). A garbling scheme GC for circuit family $\{C_{\mathsf{inp}(\lambda)}\}_{\lambda\in\mathbb{N}}$ is correct if for all $C \in C_{\mathsf{inp}(\lambda)}$ and all $x \in \{0,1\}^{\mathsf{inp}(\lambda)}$, we have

$$\Pr\left[\mathsf{Eval}\left(\{\mathsf{lab}_{i,x_i}\}_{i\in[\mathsf{inp}]}\right) \neq C(x) : \left(\{\mathsf{lab}_{i,b}\}_{i\in[\mathsf{inp}],b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C)\right] = \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of Garble.

Definition 4.2 (Security). A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for $C = \{C_{\mathsf{inp}(\lambda)}\}_{\lambda\in\mathbb{N}}$ is said to be a secure garbling scheme if there exists a PPT simulator Sim such that for all $\lambda \in \mathbb{N}, \mathsf{inp}, C \in C_{\mathsf{inp}(\lambda)}$ and $x \in \{0,1\}^{\mathsf{inp}(\lambda)}$, we have

$$\begin{aligned} & \left\{ \{\mathsf{lab}_i\}_{i\in[\mathsf{inp}]} \ : \ \{\mathsf{lab}_i\}_{i\in[\mathsf{inp}]} \leftarrow \mathsf{Sim}\left(1^\lambda, 1^{\mathsf{inp}}, 1^{|C|}, C(x)\right) \right\} \\ \approx_c \ & \left\{ \{\mathsf{lab}_{i,x_i}\}_{i\in[\mathsf{inp}]} \ : \ \{\mathsf{lab}_{i,b}\}_{i\in[\mathsf{inp}],b\in\{0,1\}} \leftarrow \mathsf{Garble}\left(1^\lambda, C\right) \right\} \end{aligned}$$

Garbled circuits are known to exist from one-way functions [173, 36]. Though the above definition is not as general as one in [36], it suffices for our constructions in this paper.

Remark 11. Note that in the standard syntax of garbling scheme [145, 35], the output of the garbling algorithm consists of labels along with a garbled circuit. On the other hand, the output only consists of labels in our syntax. However, the former can easily be converted into the latter by including the garbled circuit into a label for example. This change in syntax is made for simplifying our constructions and notations. We also note that the same primitive is called decomposable randomized encoding in [106].

Remark 12 (Multi-instance security definition)**.** The above definition allows to argue security of a single instance of GC. In the multi-instance variant, the adversary can adaptively make polynomial number of garbling queries to the challenger. All queries are answered by honestly generated labels in the real world whereas they are all simulated in the ideal world. If both worlds are computationally indistinguishable, we say that GC

satisfies multi-instance security. Note that such *multi-instance* security follows from the standard single instance security (from Definition 4.2 above) by a set of simple hybrid arguments.

4.2.2 Identity-Based Encryption

We define identity-based encryption (IBE) [137, 42, 77] in this section. An IBE scheme IBE with identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the unary representation of security parameter and outputs the master public and secret keys mpk and msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm takes as input the master secret key msk and an identity $\mathsf{id} \in \mathcal{I}_\lambda$. It outputs a corresponding secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the master public parameter mpk, an identity $\mathsf{id} \in \mathcal{I}_\lambda$ and a message $m \in \{0,1\}^*$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to m'/\perp$. The decryption algorithm takes as input the secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext ct, and outputs either a message $m' \in \{0,1\}^*$ or $\perp$.

Definition 4.3 (Correctness). For correctness, we require that for all $\lambda \in \mathbb{N}, m \in \{0,1\}^*, \mathsf{id} \in \mathcal{I}_\lambda$,

$$\Pr\left[\begin{array}{l} \mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) = m \mid (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^L), \\ \quad \mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}), \mathsf{ct}_{\mathsf{id}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \end{array}\right] = 1,$$

where the probability is taken over the random coins of $\mathsf{Setup}, \mathsf{KeyGen}$ and Enc.

Remark 13 (About the identity space.)**.** In this paper, we set the identity space $\mathcal{I}$ to be $\{0,1\}^*$ by default. This is without loss of generality assuming collision resistant hash function and an IBE scheme with sufficiently large identity space (e.g., $\mathcal{I}_\lambda = \{0,1\}^{2\lambda}$),

since we can hash any string into a string of fixed length using collision resistant hash function.

Remark 14 (About the message length.)**.** In the above definition, we assume that the message space is $\{0,1\}^*$. This is without loss of generality if we consider IND-CPA security for IBE, since it is straightforward to extend the message space by encrypting each bit (or chunk) of the message in parallel.

Definition 4.4 (IND-CPA **Security).** An IBE scheme IBE for an identity space $\mathcal{I}_\lambda$ and message space $\{0,1\}^*$ is said to satisfy indistinguishability based security if for any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(.)$ such that

$$\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{(0)}_{\mathsf{IBE},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathsf{IBE},\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Exp}^{(b)}_{\mathsf{IBE},\mathcal{A}}$, modelled as a game between adversary and challenger, is defined as follows:

1. **Setup Phase:** On input 1^λ from the adversary $\mathcal{A}$, the challenger samples $(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and replies to $\mathcal{A}$ with mpk.

2. **Query phase:** During the game, $\mathcal{A}$ adaptively makes the following queries in any arbitrary order and unbounded many times.

   a) **Key Queries:** $\mathcal{A}$ chooses an identity $\mathsf{id} \in \mathcal{I}_\lambda$ and sends it to the challenger. For each such query, the challenger replies with $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk},\mathsf{id})$.

   b) **Encryption Queries:** $\mathcal{A}$ submits to the challenger, an identity $\mathsf{id}^* \in \mathcal{I}_\lambda$ and a pair of equal length messages (m_0, m_1). The challenger replies with $\mathsf{ct}_{\mathsf{id}^*} \leftarrow \mathsf{Enc}(\mathsf{mpk},\mathsf{id}^*, m_b)$.

3. **Output phase:** $\mathcal{A}$ outputs it's guess b' as output of the experiment.

We say an adversary $\mathcal{A}$ is legitimate if during the challenge phase, it is restricted to query for ciphertexts corresponding to identities $\mathsf{id}^* \in \mathcal{I}_\lambda$ whose secret keys are not queried during any key query.

Remark 15. Since we are in the public key setting, we can simplify the above experiment so that the encryption query is allowed for only once. This simplified definition is shown to equivalent to the definition above by a simple hybrid argument. We adopt the above

multi-challenge security definition since it is convenient for our purpose in this paper.

4.2.3 Turing Machines

Here, we recall the definition of a Turing machine (TM) following [142]. The definition considers Turing machines with two tapes, namely, input tape and working tape.

Definition 4.5 (Turing Machine)**.** A (deterministic) TM M is represented by the tuple $M = (Q, \delta, F)$ where Q is the number of states (we use $[Q]$ as the set of states and 1 as the initial state), $F \subset [Q]$ is the set of accepting state and

$$\delta : [Q] \times \{0,1\} \times \{0,1\} \to [Q] \times \{0,1\} \times \{0,\pm 1\} \times \{0,\pm 1\}$$
$$(q, b_1, b_2) \mapsto (q', b_2', \Delta i, \Delta j)$$

is the state transition function, which, given the current state q, the symbol b_1 on the input tape under scan, and the symbol b_2 on the work tape under scan, specifies the new state q', the symbol b_2', overwriting b_2, the direction Δi to which the input tape pointers moves, and the direction Δj to which the work tape pointer moves. The machine is required to hang (instead of halting) once it reaches an accepting state, i.e., for all $q \in [Q]$ such that $q \in F$ and $b_1, b_2 \in \{0,1\}$, it holds that $\delta(q, b_1, b_2) = (q, b_2, 0, 0)$.

For input length $n \geq 1$ and space complexity bound $s \geq 1$, the set of internal configurations of M is

$$\mathcal{Q}_{M,n,s} = [n] \times [s] \times \{0,1\}^s \times [Q]$$

where $(i, j, W, q) \in \mathcal{Q}_{M,n,s}$ specifies the input tape pointer $i \in [n]$, the work tape pointer $j \in [s]$, the content of the work tape $W \in \{0,1\}^s$ and the machine state $q \in [Q]$.

For any bit-string $x \in \{0,1\}^n$ for $n \geq 1$ and time/space complexity bounds $t, s \geq 1$, the machine M accepts x within time t and space s if there exists a sequence of internal configurations (computation path of t steps) $c_0, \ldots, c_t \in \mathcal{Q}_{M,n,s}$ with $c_k = (i_k, j_k, W_k, q_k)$

such that $(i_0, j_0, W_0, q_0) = (1, 1, 0^s, 1)$ (initial configuration),

$$\text{for all } 0 \le k < t\text{:} \quad \begin{cases} \delta(q_k, x[i_k], W_k[j_k]) = (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k) \\ W_{k+1}[j] = W_k[j] \quad \text{for all } j \neq j_k \qquad \text{(valid transitions);} \end{cases}$$

where $x[i]$ is the i the bit of the string x and $W_k[j]$ is the j the bit of the string W_k, and $q_t \in F$ (accepting). We also say M accepts x within time t (without the space bound) if M accepts x within time t and space $s = t$.

Next, we define time/space bounded computation with *non-deterministic* Turing machines. The definition is the same as Definition 4.5, except with the following changes:

- The transition criterion δ can be any relation between (i.e., any subset of the Cartesian product of) $[Q] \times \{0,1\}^2$ and $[Q] \times \{0,1\} \times \{0, \pm q\}^2$, where $((q, b_1, b_2), (b_2', b_2', \Delta i, \Delta j)) \in \delta$ means that if the current state is q, the input tape symbol under scan is b_1 and the work tape symbol under scan is b_2, then it is valid to transit into state q', overwrite b_2 with b_2', and move the input and work tape pointer by offsets Δi and Δj respectively.

- The definition of hanging in accepting states is that for all $q \in [Q]$ such that $q \in F$ and all $b_1, b_2 \in \{0,1\}$,

$$\delta \cap \Big(\{(q, b_1, b_2)\} \times ([Q] \times \{0,1\} \times \{0, \pm 1\}^2) \Big) = \{(q, b_1, b_2), (q, b_2, 0, 0)\}\,.$$

- In the definition of acceptance

$$\delta(q_k, x[i_k], W_k[j_k]) = (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k)$$

is changed to $((q_k, x[i_k], W_k[j_k]), (q_{k+1}, W_{k+1}[j_k], i_{k+1} - i_k, j_{k+1} - j_k)) \in \delta$.

The following lemma can be obtained by a simple argument on emulating a Turing machine on Boolean circuits. While we have many other clever methods of simulating Turing machines on circuits (e.g., [152]), we use the following simple one because it evaluates the depth of the circuit as a function on the size of Turing machine, which is usually regarded as a constant and ignored.

Lemma 4.1 (Emulating a Turing Machine on Circuit)**.** *Consider a circuit that takes as input a description of a (deterministic) Turing machine $M = (Q, \delta, F)$, input x to M, a configuration $(i, j, W, q) \in Q_{M,|x|,|W|}$ and outputs the next configuration (i', j', W', q').*

We can implement such a circuit with depth $\mathrm{poly}(\log|x|, \log|W|, \log|M|)$ *and size* $\mathrm{poly}(|x|, |W|, |M|)$.

Proof. The circuit is implemented as follows. We focus on the depth of the circuits, since the bound on the size will be clear from the description. Given the input, it first retrieves the i-th bit $x[i] \in \{0, 1\}$ of the input tape. This can be done by a circuit with depth $O(\log|x|)$, which checks whether $i = \nu$ or not for each position ν of the string x in parallel and returns $x[\nu]$ for ν such that $\nu = i$. Similarly, it can retrieve $W[j]$ with depth $O(\log|W|)$. Given $x[i]$ and $W[j]$, it then retrieves $\delta(q, x[i], W[j])$ from δ, which can be done with depth $O(\log|Q|)$ similarly to the above. Given $\delta(q, x[i], W[j]) = (q', b', \Delta i, \Delta j)$, the update of i and j can be done in depth $O(\log|x|)$ and $O(\log|W|)$, respectively. Writing back the new value b' can also be done in depth $O(\log|W|)$ by finding the right place to write in the tape and change the value there. From the above discussion, the total depth of the circuit is $\mathrm{poly}(\log|x|, \log|W|, \log|M|)$ as desired. ∎

We also need the following lemma, which can be obtained by a simple observation.

Lemma 4.2 (Checking Transition for Non-deterministic Turing Machine)**.** *Consider a circuit that takes as input a description of a non-deterministic Turing machine* $M = (Q, \delta, F)$*, input* x *to* M*, two configurations* $(i, j, W, q) \in Q_{M,|x|,|W|}$ *and* $(i', j', W', q') \in Q_{M,|x|,|W|}$ *and outputs whether* $((i, j, W, q), (i', j', W', q')) \in \delta$ *or not. We can implement such a circuit with depth* $\mathrm{poly}(\log|x|, \log|W|, \log|M|)$ *and size* $\mathrm{poly}(|x|, |W|, |M|)$.

Proof. The circuit is implemented as follows. We focus on the depth of the circuits, since the bound on the size will be clear from the description. Given the input, it first checks whether $i' - i \in \{0, \pm 1\}$, $j' - j \in \{0, \pm 1\}$. Clearly, this can be done in depth $\mathrm{poly}(\log|x|, \log|W|)$. It then checks whether $W'[k] = W[k]$ for all $k \in [|W|]\backslash\{k\}$, which can be done in depth $\mathrm{poly}(\log|W|)$. It then checks whether $((q, x[i], W[j]), (q', W'[j], i' - i, j' - j)) \in \delta$ or not. This can be checked in depth

$\mathrm{poly}(\log |Q|)$. Therefore, the total depth of the circuit is $\mathrm{poly}(\log |x|, \log |W|, \log |M|)$ as desired. ■

4.2.4 Attribute Based Encryption

In the following, we recall definitions of various ABEs by specifying the relation. We refer to Section 2.2 for the definition of an ABE scheme.

Key-policy Attribute Based encryption (kpABE) for circuits. To define kpABE for circuits, we set $\mathcal{X} = \{0,1\}^*$ and $\mathcal{Y}$ as the set of all circuits and define $R(\mathbf{x}, C) = C(\mathbf{x})$. In this paper, we consider circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$ and depth $\mathsf{dep} := \mathsf{dep}(\lambda)$. To do so, we set $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}})$, $\mathcal{X}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$, and $\mathcal{Y}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp},\mathsf{dep}}$.

Ciphertext-policy Attribute Based encryption (cpABE) for circuits. To define cpABE for circuits, we set $\mathcal{Y} = \{0,1\}^*$ and $\mathcal{X}$ as the set of all circuits and define $R(C, \mathbf{x}) = C(\mathbf{x})$. In this paper, we consider circuit class $\mathcal{C}_{\mathsf{inp},\mathsf{dep}}$ that consists of circuits with input length $\mathsf{inp} := \mathsf{inp}(\lambda)$ and depth $\mathsf{dep} := \mathsf{dep}(\lambda)$. To do so, we set $\mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}})$, $\mathcal{X}_{\mathsf{prm}} = \mathcal{C}_{\mathsf{inp},\mathsf{dep}}$, and $\mathcal{Y}_{\mathsf{prm}} = \{0,1\}^{\mathsf{inp}}$.

ABE for Turing Machines. To define ABE for Turing machines, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all Turing machine, and define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\bot\}$ as

$$R((\mathbf{x}, 1^t), M) = \begin{cases} 0 & \text{if } M \text{ accepts } \mathbf{x} \text{ in } t \text{ steps} \\ 1 & \text{otherwise.} \end{cases}$$

ABE for NL. To define ABE for NL, we set $\mathcal{X} = \{0,1\}^*$, $\mathcal{Y}$ to be set of all non-deterministic Turing machines with two tapes, one of which encodes the input and can only be read, whereas the other tape can be read as well as written. When we measure

the space complexity of the computation, we consider the space being used for the latter tape. We define $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \cup \{\perp\}$ as

$$R((\mathbf{x}, 1^t, 1^{2^s}), M) = \begin{cases} 0 & \text{if } M \text{ accepts } \mathbf{x} \text{ within } t \text{ steps and space } s \\ 1 & \text{otherwise.} \end{cases}$$

Note that here, s is in the exponent to reflect the idea that the space for the computation is logarithmically bounded.

We will use the kpABE scheme given by [43] and the cpABE scheme by [169] for our constructions in this chapter. We refer to Theorem 3.3 and Theorem 3.4 for the properties of [43] and [169] schemes, respectively.

4.2.5 Tensors

In this work, similarly to [169], we use the tensor product techniques. Let $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{B} \in \mathbb{Z}_q^{s \times t}$. The tensor product is defined as:

$$\mathbf{A} \otimes \mathbf{B} \stackrel{\text{def}}{=} \begin{pmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \cdots & a_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{Z}_q^{ms \times nt}.$$

Throughout the paper, we will heavily use the mixed-product equality, stated as follows. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{B} \in \mathbb{Z}_q^{s \times t}$, $\mathbf{C} \in \mathbb{Z}_q^{n \times u}$ and $\mathbf{D} \in \mathbb{Z}_q^{t \times v}$,

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \in \mathbb{Z}_q^{ms \times uv}.$$

The mixed-product can be naturally generalized following

$$(\mathbf{A}^1 \otimes \cdots \otimes \mathbf{A}^k) \cdot (\mathbf{B}^1 \otimes \cdots \otimes \mathbf{B}^k) = (\mathbf{A}^1\mathbf{B}^1) \otimes \cdots \otimes (\mathbf{A}^k\mathbf{B}^k).$$

Note that we adopt the same convention as in [169] where matrix multiplication takes precedence over tensor products, i.e. $\mathbf{A} \otimes \mathbf{BC} = \mathbf{A} \otimes (\mathbf{BC})$.

4.2.6 Hardness Assumptions

Assumption 4.3 (Circular Small Secret LWE)**.** [122] Let n, m, m', q, χ, χ' be functions of λ and

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \bar{\mathbf{A}}' \leftarrow \mathbb{Z}_q^{n\times m'},\ \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^n,\ \mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal,\ \ \mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^m,\ \ \mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m'},$$
$$\mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m},\ \ \delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^m,\ \ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \ \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$$

The circular small-secret LWE assumption $\mathsf{csLWE}_{n,m,m',q,\chi,\chi'}$ states that

$$\left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal\end{pmatrix}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal\end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s})\otimes\mathbf{G}, \bar{\mathbf{A}}', \mathbf{r}^\intercal\bar{\mathbf{A}}' + (\mathbf{e}')^\intercal\right)\right\}_{\lambda\in\mathbb{N}}$$
$$\approx \left\{\left(1^\lambda, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \delta_{\mathsf{fhe}}^\intercal\end{pmatrix}, \quad \Delta, \quad \bar{\mathbf{A}}', \ \ (\delta_i)^\intercal\right)\right\}_{\lambda\in\mathbb{N}}$$

4.2.7 GSW Homomorphic Encryption and Evaluation

We recall the format (without the distribution) of the (leveled fully) homomorphic encryption [99] and the correctness property. We adapt the syntax from [122].

Lemma 4.4. *The leveled FHE scheme works as follows:*

- *The keys are*

$$\textit{(public)} \quad \mathbf{A}_{\mathsf{fhe}} = \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \bar{\mathbf{s}}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal\end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}, \quad \textit{(secret)} \quad \mathbf{s}^\intercal = (\bar{\mathbf{s}}^\intercal, -1),$$

  where $\bar{\mathbf{s}} \in \mathbb{Z}^n$,$\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}$, *and* $\mathbf{e}_{\mathsf{fhe}}^\intercal \in \mathbb{Z}^m$.

- *A ciphertext of* $x \in \{0,1\}$ *is* $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - x\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m}$, *where* $\mathbf{R} \in \mathbb{Z}_q^{m\times m}$ *is the encryption randomness. The decryption equation is*

$$\mathbf{s}^\intercal\mathbf{X} = -\mathbf{e}_{\mathsf{fhe}}^\intercal\mathbf{R} - x\mathbf{s}^\intercal\mathbf{G} \in \mathbb{Z}_q^m, \tag{4.4}$$

  which can be used to extract x *via multiplication by* $\mathbf{G}^{-1}(\lfloor q/2\rfloor\iota_{n+1})$.

- *There is an efficient algorithm*

$$\mathit{MakeHEvalCkt}(1^n, 1^m, q, C) = \mathsf{HEval}_C$$

that takes as input n, m, q and a circuit $C : \{0,1\}^L \to \{0,1\}$ *and outputs a circuit*

$$\mathsf{HEval}_C(\mathbf{X}_1, \ldots, \mathbf{X}_L) = \mathbf{C}$$

taking L ciphertexts as input and outputting a new ciphertext $\mathbf{C}$*.*

- *The depth of* HEval_C *is* $dO(\log m \log\log q)$*, where d is the depth of C.*
- *Suppose* $\mathbf{X}_\ell = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ *for* $\ell \in [L]$ *with* $\mathbf{x} \in \{0,1\}^L$*, then*

$$\mathbf{C} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_C - C(\mathbf{x})\mathbf{G},$$

where $\left\|\mathbf{R}_C^\intercal\right\| \le (m+2)^d \mathsf{max}_{\ell\in[L]} \left\|\mathbf{R}_\ell^\intercal\right\|$.

Additionally, in the circular version, ciphertexts of $\mathsf{bits}(\mathbf{s})$ *are published.*

Lemma 4.5. *(homomorphic evaluation for vector-valued functions [122]) There is an efficient algorithm*

$$\mathit{MakeVEvalCkt}(1^n, 1^m, q, C) = \mathsf{VEval}_C$$

that takes as input n, m, q and a vector-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{1\times m'}$ *and outputs a circuit*

$$\mathsf{VEval}_C(\mathbf{X}_1, ..., \mathbf{X}_L) = \mathbf{C},$$

taking L ciphertexts as input and outputting a new ciphertext $\mathbf{C}$ *of different format.*

- *The depth of* VEval_C *is* $d \cdot O(\log m \log\log q) + O(\log^2 \log q)$ *for C of depth d.*
- *Suppose* $\mathbf{X}_\ell = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_\ell - \mathbf{x}[\ell]\mathbf{G}$ *for* $\ell \in [L]$ *with* $\mathbf{x} \in \{0,1\}^L$*, then*

$$\mathbf{C} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_C - \begin{pmatrix} \mathbf{0}_{n\times m'} \\ C(\mathbf{x}) \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m'},$$

where $\left\|\mathbf{R}_C^\intercal\right\| \le (m+2)^d \lceil \log q \rceil \mathsf{max}_{\ell\in[L]} \left\|\mathbf{R}_\ell^\intercal\right\|$*. The new decryption equation is*

$$\mathbf{s}^\intercal \mathbf{C} = -\mathbf{e}_{\mathsf{fhe}}^\intercal \mathbf{R}_C + C(\mathbf{x}) \in \mathbb{Z}_q^{1\times m'}.$$

4.2.8 BGG+ Homomorphic Evaluation Procedures

In this section we describe the properties of the attribute encoding and its homomorphic evaluation. We adapt the syntax from [122].

- For L-bit input, the public parameter is $\mathbf{A}_{\mathsf{att}} \in \mathbb{Z}_q^{(n+1)\times(L+1)m}$.

- The encoding of $\mathbf{x} \in \{0,1\}^L$ is

$$\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal,$$

  where $\mathbf{s}^\intercal = (\bar{\mathbf{s}}^\intercal, -1)$ with $\bar{\mathbf{s}} \in \mathbb{Z}^n$ and $\mathbf{e}_{\mathsf{att}}^\intercal \in \mathbb{Z}^{(L+1)m}$.

- There are efficient deterministic algorithms [43]

$$\mathsf{EvalC}(\mathbf{A}_{\mathsf{att}}, C) = \mathbf{H}_C \quad \text{and} \quad \mathsf{EvalCX}(\mathbf{A}_{\mathsf{att}}, C, \mathbf{x}) = \mathbf{H}_{C,\mathbf{x}}$$

  that take as input $\mathbf{A}_{\mathsf{att}}$, a circuit $C : \{0,1\}^L \to \{0,1\}$, and (for EvalCX) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m \times m}$.

  - Suppose C is of depth d, then $\left\|\mathbf{H}_C^\intercal\right\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \leq (m+2)^d$.
  - They satisfy encoding homomorphism, $(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})\mathbf{G}$.

- There are efficient deterministic algorithms [60]

$$\mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, C) = \mathbf{H}_C \quad \text{and} \quad \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, C, \mathbf{x}) = \mathbf{H}_{C,x}$$

  that take as input $\mathbf{A}_{\mathsf{att}}$, a matrix-valued circuit $C : \{0,1\}^L \to \mathbb{Z}_q^{n+1 \times m'}$, and (for $\mathsf{MEvalCX}$) some $\mathbf{x} \in \{0,1\}^L$, and output some matrix in $\mathbb{Z}^{(L+1)m \times m'}$.

  - Suppose C is of depth d, then $\left\|\mathbf{H}_C^\intercal\right\|, \|\mathbf{H}_{C,\mathbf{x}}^\intercal\| \leq (m+2)^d \lceil \log q \rceil$.
  - The matrix encoding homomorphism is $(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G})\mathbf{H}_{C,\mathbf{x}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}_C - C(\mathbf{x})$.

Dual-Use Technique and Extension. In [60], the attribute encoded with secret $\mathbf{s}^\intercal$ is FHE ciphertexts under key $\mathbf{s}^\intercal$ (the same, "dual-use") and the circuit being $\mathsf{MEvalCX}$'ed is some HEval_C. This leads to automatic decryption. Let C be a circuit with Boolean output, $\mathbf{x}$ an input, $\mathbf{X}$ a bunch of FHE ciphertexts of $\mathsf{bits}(\mathbf{x})$ under $\mathbf{s}^\intercal$, and $\mathbf{e}_{\mathsf{att}}$, $\mathbf{e}'$, $\mathbf{e}''$ some unspecified noises, then

$$\begin{aligned}
&\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathsf{HEval}_C,\mathbf{X}} \\
&= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal \mathsf{HEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \quad (\mathsf{MEvalCX})
\end{aligned}$$

$$= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - \mathbf{s}^\intercal C(\mathbf{x})\mathbf{G} + (\mathbf{e}'')^\intercal \quad (\mathsf{HEval}\text{ decryption })$$

$$= \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{att}} \mathbf{H}_{\mathsf{HEval}_C} - C(\mathbf{x})\mathbf{G}) + (\mathbf{e}'')^\intercal .$$

To extend the dual-use technique to vector-valued circuits, let the codomain of C be $\mathbb{Z}_q^{1\times m'}$, then VEval_C is $\mathbb{Z}_q^{(n+1)\times m'}$-valued and

$$\mathbf{s}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathsf{bits}(\mathbf{X})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathsf{VEval}_C, \mathbf{X}}$$

$$= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - \mathbf{s}^\intercal \mathsf{VEval}_C(\mathbf{X}) + (\mathbf{e}')^\intercal \quad (\mathsf{MEvalCX})$$

$$= \mathbf{s}^\intercal \mathbf{A}_{\mathsf{att}} \mathsf{VEval}_C - C(\mathbf{x}) + (\mathbf{e}'')^\intercal \quad (\mathsf{VEval}\text{ decryption }).$$

Extension to circular encryption means setting $\mathbf{x} = \mathbf{s}$, for which we say $\mathbf{S}$, $\mathbf{A}_{\mathsf{circ}}$ in place of $\mathbf{X}$, $\mathbf{A}_{\mathsf{att}}$.

4.3 BOOTSTRAPPING RANDOMIZED HOMOMORPHIC EVALUATION

In this section, we show how to achieve unbounded homomorphism for a randomized attribute encoding of the form

$$\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}^\intercal (\mathbf{I}_{n+1} \otimes \mathbf{r})(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal$$

where $\mathbf{s} \in \mathbb{Z}^{m(n+1)}, \mathbf{r} \in \mathbb{Z}^m, \mathbf{A}_{\mathsf{att}} \in \mathbb{Z}_q^{(n+1)\times m}, x \in \{0,1\}^{\mathrm{L}}, \mathbf{e}_{\mathsf{att}}^\intercal \in \mathbb{Z}^m$. We let $\|\mathbf{e}_{\mathsf{att}}\|$ and $\|(\mathbf{s}^\intercal (\mathbf{I}_{n+1} \otimes \mathbf{r}))^\intercal\|$ be bounded by B. Here B denotes the bound of removable noise. We achieve unbounded homomorphism for the above randomized attribute encoding using the following steps.

1. **Noise removal for randomized encoding.** First we transform $\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal$ into a noiseless encoding using the procedure $\mathsf{RemoveNoise}(\cdot)$ from HLL to achieve $\mathsf{RemoveNoise}(\mathbf{c}_{\mathsf{att}}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) - C(\mathbf{x})\mathbf{s}_{\mathbf{r}}^\intercal \mathbf{G}$ where $\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) = \mathsf{RemoveNoise}(\mathbf{s}_{\mathbf{r}}^\intercal \mathbf{A})$.

2. **Structure restoration.** Here we transform the noiseless encoding achieved above

to an attribute encoding with a smaller noise, following the blueprint of [122].

Before describing the above steps in detail, we first state a few tools, theorems and lemmas that will be useful later.

4.3.1 Preparation

Rounding Function. Let $\mathsf{Rnd} : \mathbb{Z}_q \to \mathbb{Z}_q$ be a rounding function defined as $\mathsf{Rnd}(x) = (q/p)\lfloor p/q \cdot x \rceil$, where p divides q. We make the following observations for the function Rnd.

Claim 4.6. For $x \in \mathbb{Z}_q$, $\mathsf{Rnd}(x) = x + e_{\mathsf{Rnd}}$ where $|e_{\mathsf{Rnd}}| \leq \frac{q}{2p}$.

Proof. The rounding function $\mathrm{Rnd}(x)$ first rounds $\frac{p}{q} \cdot x$ to the nearest integer, then scales it up to $\mathbb{Z}_q$ by multiplying with $\frac{q}{p}$. Let m be the nearest integer to $\frac{p}{q} \cdot x$. Then, $\mathrm{Rnd}(x) = m \cdot \frac{q}{p}$. The error e_{Rnd} introduced by the rounding process is defined as $e_{\mathrm{Rnd}} = \mathrm{Rnd}(x) - x$. When $\frac{p}{q} \cdot x$ is rounded to m, the error introduced is at most $\frac{1}{2}$. Scaling this error up to $\mathbb{Z}_q$ by multiplying with $\frac{q}{p}$, the maximum error becomes $\frac{q}{p} \cdot \frac{1}{2} = \frac{q}{2p}$. Therefore, we have $|e_{\mathrm{Rnd}}| \leq \frac{q}{2p}$. ∎

Claim 4.7. For random $x, x' \in \mathbb{Z}_q$ such that $|x - x'|$ is negligibly smaller than $\frac{q}{p}$, we have $\mathsf{Rnd}(x) - \mathsf{Rnd}(x') = 0$ with all but negligible probability.

Proof. Let ϵ represent the negligible quantity, such that $|x - x'| \leq \epsilon \cdot \frac{q}{p}$. The function Rnd operates by scaling x to the $\mathbb{Z}_p$ space, rounding it, and then scaling back to $\mathbb{Z}_q$. This creates intervals in $\mathbb{Z}_q$ of size $\frac{q}{p}$ that map to each integer in $\mathbb{Z}_p$. The probability that x and x' fall into different rounding intervals is related to how close x is to the boundary of a rounding interval. If x and x' are within $\epsilon \times \frac{q}{p}$ of each other, for a small ϵ, and they straddle a rounding boundary, then they round to different values.

Given a uniform distribution of x and x' over $\mathbb{Z}_q$, the probability of a number falling within $\epsilon \times \frac{q}{p}$ of a rounding boundary is approximately 2ϵ because there are two boundaries (upper and lower) for each interval, and each boundary has a "risk zone" of $\epsilon \times \frac{q}{p}$ around

it.

Therefore, $\mathsf{Pr}[\mathsf{Rnd}(x) \neq \mathsf{Rnd}(x') \mid |x - x'| \leq \epsilon \cdot (q/p)] \leq 2\epsilon$, which is negligible. ■

Theorem 4.8 (Noise Removal: [122] Construction 1)**.** *There exists a deterministic procedure* $\mathsf{RemoveNoise}(\cdot)$ *such that for an input* $\mathbf{u}^\intercal = \mathbf{s}^\intercal(\mathbf{A} - x\mathbf{G}) + \mathbf{e} \in \mathbb{Z}_q^{1\times m}$*, where* $\mathbf{A} \in \mathbb{Z}_q^{n+1\times m}, x \in \{0,1\}, \|\mathbf{s}\| \leq B, \|\mathbf{e}\| \leq B$*,*

$$\mathsf{RemoveNoise}(\mathbf{u}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) - x\mathbf{s}^\intercal\mathbf{G} \in \mathbb{Z}_q^{1\times m}$$

where $\mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) = \mathsf{RemoveNoise}(\mathbf{s}^\intercal\mathbf{A})$.

Lemma 4.9. *A canonical Boolean circuit of* $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ *is of depth* $O(\log n \log\log q)$ *and can be efficiently generated from* $\mathbf{A} \in \mathbb{Z}_q^{(n+1)\times m}$.

Theorem 4.10 (Bootstrapping: [122] Theorem 12)**.** *It works as follows:*

- *The secret is* $\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal \in \mathbb{Z}^{n+1}$ *with* $\mathsf{bits}(\mathbf{s}) \in \{0,1\}^{(n+1)\lceil\log q\rceil}$*. The circular ciphertext is*

$$\mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} (\mathbf{R}_1, \cdots, \mathbf{R}_{(n+1)\lceil\log q\rceil}) - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil\log q\rceil}$$

  where $\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{(n+1)\times m}$*,* $\mathbf{e}_{\mathsf{fhe}}^\intercal \in \mathbb{Z}^m$*, and* $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$ *for* $1 \leq i \leq (n+1)\lceil\log q\rceil$*. Let* $L_S = m(n+1)^2\lceil\log q\rceil^2$*, so that* $\mathsf{bits}(\mathbf{S}) \in \{0,1\}^{1\times L_S}$.

- *The circular encoding is* $\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$ *where* $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1)\times(L_S+1)m}$ *and* $\mathbf{e}_{\mathsf{circ}} \in \mathbb{Z}^{(L_S+1)m}$.

- *There are efficient deterministic algorithms*

$$\mathsf{EvalRndPad}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}) = H_{\mathbf{A}}^{\mathsf{RndPad}} \quad \textit{and} \quad \mathsf{EvalRndPad}S(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}, \mathbf{S}) = H_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}}$$

  such that

$$\left\|(H_{\mathbf{A}}^{\mathsf{RndPad}})^\intercal\right\|, \left\|(H_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}})^\intercal\right\| \leq 2^{O(\log^5\lambda)}.$$

  Moreover, when $\mathbf{S}$ *is indeed of the correct form for* $\mathsf{RndPad}_{\mathbf{A}}(\cdot)$ *defined in Theorem 4.8,*

$$\begin{aligned} &\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G})H_{\mathbf{A},\mathbf{S}}^{\mathsf{RndPad}} \\ &= \mathbf{s}^\intercal\mathbf{A}_{\mathsf{circ}}H_{\mathbf{A}}^{\mathsf{RndPad}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}) + \mathbf{e}_{\mathsf{fhe}}^\intercal\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}} \end{aligned}$$

  where $\left\|\mathbf{R}_{\mathsf{RndPad}_{\mathbf{A}}}^\intercal\right\| \leq 2^{O(\log^4\lambda)}$.

Theorem 4.11 (Unbounded Homomorphic Evaluation: [122] Construction 2)**.** *Let*

$C : \{0,1\}^L \to \{0,1\}$ *and* $\mathbf{x} \in \{0,1\}^L$. *Suppose*

$$\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal, \ , \mathbf{R} \in \{0,1\}^{m \times m(n+1)\log q}, \ \ \mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}) \otimes \mathbf{G}$$

$$\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{att}} + (1, \mathbf{x}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal, \ \mathbf{c}_{\mathsf{circ}}^\intercal = \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal,$$

where $\|\mathbf{s}\| \le B, \|\mathbf{e}_{\mathsf{att}}\|, \left\|\mathbf{e}_{\mathsf{fhe}}^\intercal\right\|, \|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda} B$.

There are two efficient algorithms

$$\mathsf{UEvalC}(\mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, C) = \mathbf{A}_C, \qquad \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{ct}_{\mathsf{att}}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathsf{ct}_{\mathsf{circ}}^\intercal, C, \mathbf{x}, \mathbf{S}) = \mathbf{c}_{C,\mathbf{x}}^\intercal$$

satisfying, with all but negligible probability, $\mathbf{c}_{C,\mathbf{x}}^\intercal = \mathbf{s}^\intercal (\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^\intercal$ *and*

$$\left\|\mathbf{e}_{C,\mathbf{x}}\right\| \le (\|\mathbf{e}_{\mathsf{fhe}}\| + \|\mathbf{e}_{\mathsf{circ}}\|) \cdot 2^{\mathrm{poly}(\log q)} \le B.$$

4.3.2 Noise Removal for Randomized Encoding

Here we use the $\mathsf{RemoveNoise}(\cdot)$ procedure from Theorem 4.8 to remove noise from a randomized attribute encoding $\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal (\mathbf{A} - C(\mathbf{x})\mathbf{G}) + \mathbf{e}^\intercal \in \mathbb{Z}_q^{1 \times m}$ with $\|\mathbf{s}_{\mathbf{r}}\|, \|\mathbf{e}\| \le B$. Here we have $\mathbf{s}_{\mathbf{r}}^\intercal = \mathbf{s}^\intercal (\mathbf{I}_{n+1} \otimes \mathbf{r})$, $\mathbf{s} \in \mathbb{Z}^{m(n+1)}, \mathbf{r} \in \mathbb{Z}^m, \mathbf{A} \in \mathbb{Z}_q^{(n+1) \times m}$, and $\mathbf{e} \in \mathbb{Z}^m$.

We have the following using Theorem 4.8.

$$\mathsf{RemoveNoise}(\mathbf{s}_{\mathbf{r}}^\intercal (\mathbf{A} - C(\mathbf{x})\mathbf{G}) + \mathbf{e}^\intercal) = \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) - C(\mathbf{x})\mathbf{s}_{\mathbf{r}}^\intercal \mathbf{G}.$$

4.3.3 Randomized Bootstrapping a.k.a Structure Restoration

To support unbounded evaluation, we transform the noiseless encoding achieved in Section 4.3.2 back to an attribute encoding with smaller noise, following the blueprint of [122]. In particular, we want to compute $\mathbf{s}_{\mathbf{r}}^\intercal \mathbf{A}'_C - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) + (\mathbf{e}')^\intercal$ for some publicly computable matrix $\mathbf{A}'_C$ and a small noise $\mathbf{e}'$ whose norm is within the bound of removable noise. Note that, as discussed in the technical overview, this cannot be directly computed during the key generation or encryption process as it requires the knowledge of both the keygen randomness $\mathbf{r}$ and the encryption randomness $\mathbf{s}$. We divide the structure

restoration into two parts.

Step 1: Computing Advice for HLL. We define a new algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$ that enables us to compute an extra *noisy* FHE circular encryption $\mathbf{S}'_{\mathbf{r}} = \mathsf{hct}_{\mathbf{s_r}}(\mathbf{s_r}) + \mathbf{err_S}$ using key $\mathbf{s_r}$ and a circular encoding $\mathbf{E}'_{\mathbf{r}} = \mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_{\mathbf{r}})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{E}}^{\intercal}$ where $\mathbf{s}_{\mathbf{r}}^{\intercal} = \mathbf{s}^{\intercal}(\mathbf{I}_{n+1} \otimes \mathbf{r}) = (\bar{\mathbf{s}}_{\mathbf{r}}^{\intercal}, -1)$[3] is the FHE secret key.

Step 2: HLL Bootstrapping. We use $\mathbf{S}'_{\mathbf{r}}$ and $\mathbf{E}'_{\mathbf{r}}$ obtained above to compute $\mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{A} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s_r})$ similarly to HLL.

Notations. We set $m_{\mathrm{T}} = m(m(n+1)\lceil \log q \rceil + \lambda)$, $\mathrm{L}_{\mathrm{T}} = m(n+1)(m(n+1)\lceil \log q \rceil + \lambda)\lceil \log q \rceil$, $m_{\mathrm{S}} = m(n+1)\lceil \log q \rceil$, $\mathrm{L}_{\mathrm{S}} = m(n+1)^2\lceil \log q \rceil^2$, and $\ell = \mathrm{L}_{\mathrm{S}}(1 + m \log q)$.

Ingredients. We require the following tools.

1. A rounding function $\mathsf{Rnd} : \mathbb{Z}_q \to \mathbb{Z}_q$ where $\mathsf{Rnd}(x) = \frac{q}{p}\left\lfloor \frac{px}{q} \right\rceil$, where p is an integer that divides q.
2. Three pseudorandom functions: $\mathsf{PRF}_1 : \{0,1\}^\lambda \times \mathbb{Z}^m \to [-\sigma', \sigma']^{1\times m}$, $\mathsf{PRF}_2 : \{0,1\}^\lambda \times \mathbb{Z}^m \to \{0,1\}^{m \times m_{\mathrm{S}}}$ and $\mathsf{PRF}_3 : \{0,1\}^\lambda \times \mathbb{Z}^m \to [-\sigma', \sigma']^{1\times(\mathrm{L}_{\mathrm{S}}+1)m}$.

Next, we describe our $\mathsf{ComputeAdvice}_{\mathbf{r}}$ procedure.

Step 1: Procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^{\intercal})$. The algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$, with $\mathbf{r} \in \mathbb{Z}_q^m$ hardwired, has the following functionality.

Input: It takes as input $(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^{\intercal})$, where $\bar{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{A}_{\mathsf{circ}} \in \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{S}}+1)m}$,

$$\mathbf{A}_{\mathsf{path}} \in \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{T}}+1)m},\ \mathbf{T} = \begin{pmatrix} \bar{\mathbf{A}}'_{\mathsf{fhe}} \\ \bar{\mathbf{t}}^{\intercal}\bar{\mathbf{A}}'_{\mathsf{fhe}} + (\mathbf{e}'_{\mathsf{fhe}})^{\intercal} \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \in \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}}$$

[3] We sample $\mathbf{s}$ and $\mathbf{r}$ in such a way that $\mathbf{s}_{\mathbf{r}}^{\intercal} = (\bar{\mathbf{s}}_{\mathbf{r}}^{\intercal}, -1)$ holds.

$$\mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^\intercal \in \mathbb{Z}_q^{(\mathrm{L_T}+1)m},\ \mathbf{c}_1^\intercal = \mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}}) + \mathbf{e}_1^\intercal \in \mathbb{Z}_q^{\ell},$$

for $\mathbf{t} = (\bar{\mathbf{t}}^\intercal, -1)^\intercal \in \mathbb{Z}^{n+1}$, $\bar{\mathbf{A}}'_{\mathsf{fhe}} \in \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{e}'_{\mathsf{fhe}} \in \mathbb{Z}^m$, $\mathbf{R} \in \{0,1\}^{m\times m_{\mathrm{T}}}$, and $\mathbf{A}_{\mathbf{r}} = \mathbf{A}_{\mathsf{path}}\mathbf{H}_{\mathrm{F}} \in \mathbb{Z}_q^{(n+1)\times\ell}$. Here $\mathbf{H}_{\mathrm{F}}$ is a short matrix computed w.r.t. the function F defined in Figure 4.1.

Output: It outputs $(\mathbf{S}'_{\mathbf{r}} \in \mathbb{Z}_q^{(n+1)\times m_{\mathrm{S}}}, \mathbf{E}'_{\mathbf{r}} \in \mathbb{Z}_q^{1\times(\mathrm{L_S}+1)m})$, where

$$\mathbf{S}'_{\mathbf{r}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}_{\mathbf{r}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}_{\mathbf{r}}) \otimes \mathbf{G} + \mathbf{err}_{\mathbf{S}},\quad \mathbf{E}'_{\mathbf{r}} = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_{\mathbf{r}})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$$

for $\mathbf{s}_{\mathbf{r}}^\intercal = (\bar{\mathbf{s}}_{\mathbf{r}}^\intercal, -1) \in \mathbb{Z}^{n+1}$, $\mathbf{e}_{\mathsf{fhe}} \in \mathbb{Z}^m$, $\mathbf{err}_{\mathbf{S}}$[4] $\in \mathbb{Z}^{(n+1)\times m_{\mathrm{S}}}$, and $\mathbf{e}_{\mathsf{circ}} \in \mathbb{Z}_q^{1\times(\mathrm{L_S}+1)m}$.

If the terms $\mathbf{e}'_{\mathsf{fhe}}, \mathbf{e}_{\mathbf{D}}$ and $\mathbf{e}_1$ are short, then $\mathbf{e}_{\mathsf{fhe}}$, $\mathbf{err}_{\mathbf{S}}$, and $\mathbf{e}_{\mathsf{circ}}$ are short. This will be formally proven in Theorem 4.12.

The algorithm $\mathsf{ComputeAdvice}_{\mathbf{r}}$ works as follows.

1. *Defining function to evaluate.* Define function $\mathrm{F} = \mathrm{F}_{\mathbf{r}}(\cdot,\cdot)$ as in Figure 4.1. Using the fact that the computation of modular inner product and the PRF is in NC_1, we analyse the depth of F as follows.
   Step 1 and 3 can be implemented by a circuit of depth $O(\log m \log\log q + \log n \log\log q)$. Step 2 can be implemented by a circuit of depth $O(\log n \log\log q) + O(\log(m \log q)) \leq O(\log m \log\log q + \log n \log\log q)$, where the first component is w.r.t. inner product and second for the addition. Similarly step 4 can be implemented by a circuit of depth $O(\log n \log\log q)$. Thus the entire computation can be implemented with depth $d_{\mathrm{F}} = O(\log m \log\log q + \log n \log\log q)$.

2. *Defining circuits for homomorphic evaluation.* Here we define homomorphic evaluation circuits for function F.

   – Define $\mathsf{VEval}_{\mathrm{F}} = \mathrm{MakeVEvalCkt}(n, m, q, \mathrm{F}) \in \mathbb{Z}_q^{n+1\times\ell}$. From Theorem 4.5, the depth of $\mathsf{VEval}_{\mathrm{F}}$ is

   $$(d_{\mathrm{F}} O(\log m \log\log q) + O(\log^2 \log q)) = O(\log^2 m \log^2 \log q + \log m \log n \log^2 \log q).$$

[4] This is an extra noise term in the fhe ciphertext. However, we bound it such that it is still correctly decryptable by secret key $\mathbf{s}_{\mathbf{r}}$.

Function F

Hardwired constants: $\mathbf{r} \in \mathcal{D}^m_{\mathbb{Z},\sigma_{\mathbf{r}}}$.
On input $(\mathbf{s} \in \mathbb{Z}^{m(n+1)}, \mathsf{sd} \in \{0,1\}^\lambda)$, proceed as follows:

a) Set $\mathbf{s}_{\mathbf{r}}^\intercal := \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r})$ and parse $\mathbf{s}_{\mathbf{r}} = (\bar{\mathbf{s}}_{\mathbf{r}}^\intercal, -1)^\intercal$.

b) Compute $\underline{\mathbf{a}}_{\mathbf{r}}^\intercal = \bar{\mathbf{s}}_{\mathbf{r}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})$.

c) Compute $\mathbf{S}_{\mathbf{r}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \underline{\mathbf{a}}_{\mathbf{r}}^\intercal \end{pmatrix} \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}) - \mathsf{bits}(\mathbf{s}_{\mathbf{r}}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log q \rceil}$.
Parse $\mathbf{S}_{\mathbf{r}} \in \mathbb{Z}_q^{1\times m(n+1)^2\lceil \log q \rceil}$ and set $\bar{\mathbf{S}}_{\mathbf{r}} = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}})$.

d) Compute $\mathbf{E}_{\mathbf{r}} := \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_{\mathbf{r}})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}) \in \mathbb{Z}_q^{1\times(\mathrm{L_S}+1)m}$.

e) Output $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}}) \in \mathbb{Z}_q^{1\times\ell}$.

Figure 4.1: Function F

– Compute $\mathbf{H}_{\mathrm{F}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_{\mathrm{F}})$, $\mathbf{H}_{\mathrm{F},\mathbf{T}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_{\mathrm{F}}, \mathbf{T}) \in \mathbb{Z}^{(\mathrm{L_T}+1)m\times\ell}$.

Using the depth bound from Section 4.2.8, we have

$$\begin{aligned}\left\|(\mathbf{H}_{\mathrm{F}})^\intercal\right\|, \left\|\left(\mathbf{H}_{\mathrm{F},\mathbf{T}}\right)^\intercal\right\| &\le (m+2)^{\mathsf{d}_{\mathsf{VEval}_{\mathrm{F}}}\lceil \log q \rceil} \\ &= (m+2)^{O(\log^2 m \log^2 \log q + \log m \log n \log^2 \log q)}\lceil \log q \rceil \\ &\le 2^{\log^5 \lambda}.\end{aligned}$$

3. *Compute the circuit homomorphically on* $\mathbf{D}$. Here we use the homomorphic evaluation circuits to compute on $\mathbf{D}$.

$$\begin{aligned}\mathbf{D}\cdot\mathbf{H}_{\mathrm{F},\mathbf{T}} &= (\mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^\intercal)\mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^\intercal \mathbf{A}_{\mathsf{path}}\mathbf{H}_{\mathrm{F}} - \mathbf{t}^\intercal \mathsf{VEval}_{\mathrm{F}}(\mathsf{bits}(\mathbf{T})) + \mathbf{e}_{\mathrm{D}}^\intercal \mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^\intercal \mathbf{A}_{\mathbf{r}} - \mathrm{F}(\mathbf{s}, \mathsf{sd}) + (\mathbf{e}'_{\mathsf{fhe}})^\intercal \mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathrm{D}}^\intercal \mathbf{H}_{\mathrm{F},\mathbf{T}} \\ &= \mathbf{t}^\intercal \mathbf{A}_{\mathbf{r}} - (\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}}) + \mathbf{e}_{\mathrm{F}}^\intercal.\end{aligned}$$

where $\mathbf{e}_{\mathrm{F}}^\intercal = (\mathbf{e}'_{\mathsf{fhe}})^\intercal \mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathrm{D}}^\intercal \mathbf{H}_{\mathrm{F},\mathbf{T}}$ and by Theorem 4.5, we have

$$\begin{aligned}\left\|\mathbf{R}_{\mathrm{F}}^\intercal\right\| &\le (m+2)^{d_{\mathrm{F}}}\lceil \log q \rceil \cdot \mathsf{max}_{i\in[m(n+1)\lceil \log q \rceil + \lambda]} \left\|\mathbf{R}_i^\intercal\right\| \\ &\le (m+2)^{d_{\mathrm{F}}}\lceil \log q \rceil \cdot m = (m+2)^{d_{\mathrm{F}}}\lceil \log q \rceil \cdot 3(n+1)\lceil \log q \rceil \\ &\le (m+2)^{d_{\mathrm{F}}} O(\log q) \le 2^{O(\log^4 \lambda)}.\end{aligned}$$

4. *Cancel out masking term.* Compute $\mathbf{c}_1^\intercal - \mathbf{D} \cdot \mathbf{H}_{\mathrm{F},\mathbf{T}}$.

$$\begin{aligned}
&\mathbf{c}_1^\intercal - \mathbf{D} \cdot \mathbf{H}_{\mathrm{F},\mathbf{T}} \\
&= \mathbf{t}^\intercal(\mathbf{A}_\mathbf{r}) + \mathbf{e}_1^\intercal - (\mathbf{t}^\intercal \mathbf{A}_\mathbf{r} - (\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r}) + \mathbf{e}_\mathrm{F}^\intercal) \\
&= (\mathbf{S}_\mathbf{r}, \mathbf{E}_\mathbf{r}) + \mathbf{e}^\intercal = (\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal \ , \ \mathbf{E}_\mathbf{r} + \mathbf{e}_\mathbf{E}^\intercal)
\end{aligned}$$

   where $\mathbf{e}^\intercal = \mathbf{e}_1^\intercal - \mathbf{e}_\mathrm{F}^\intercal \in \mathbb{Z}^{1\times\ell}$ and $\|\mathbf{e}_\mathbf{S}\|, \|\mathbf{e}_\mathbf{E}\| \le \|\mathbf{e}\|$.

5. *Output* Output $\mathbf{S}'_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ and $\mathbf{E}'_\mathbf{r} = \mathbf{E}_\mathbf{r} + \mathbf{e}_\mathbf{E}^\intercal$.

We encapsulate the property of our algorithm $\mathsf{ComputeAdvice}_\mathbf{r}$ in the following lemma.

Lemma 4.12 ($\mathsf{ComputeAdvice}_\mathbf{r}$)**.** *If $\sigma', \left\|\mathbf{e}'_{\mathsf{fhe}}\right\|, \|\mathbf{e}_\mathbf{D}\| \le 2^{-2\lambda}B', \|\mathbf{e}_1\| \le 2^{-\lambda/2}B', \ q = B'p\lambda^{\omega(1)}, \|\mathbf{s}_\mathbf{r}\| \le 2^{-\lambda}B'$, for a bound $B' = 2^{-4\lambda}B$ where $B < q/4$ is the bound on the removable error, then for procedure $\mathsf{ComputeAdvice}_\mathbf{r}$ defined above, the output $(\mathbf{S}'_\mathbf{r}, \mathbf{E}'_\mathbf{r})$ is suitable for HLL Bootstrapping. Specifically, we have*

- $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}'_\mathbf{r}$ *with overwhelming probability.*
- $\bar{\mathbf{S}}_\mathbf{r}$ *and* $\mathbf{S}'$ *are valid fhe ciphertexts encrypting* $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ *using key* $\mathbf{s}_\mathbf{r}$ *with decryption error* $\le 2^{-\lambda}B$
- *The error in* $\mathbf{E}'_\mathbf{r}$ *is bounded as* $\|\mathbf{e}_{\mathsf{circ}}\| \le 2^{-\lambda}B$.

Proof. We show that the output of $\mathsf{ComputeAdvice}_\mathbf{r}$ is valid for HLL bootstrapping.

First we note that since $\left\|\mathbf{e}'_{\mathsf{fhe}}\right\|, \|\mathbf{e}_\mathbf{D}\| \le 2^{-2\lambda}B'$ and $\left\|\mathbf{R}_\mathrm{F}^\intercal\right\|, \left\|(\mathrm{H}_{\mathrm{F},\mathbf{T}})^\intercal\right\| \le 2^{\mathrm{poly}(\log\lambda)}$, from Step 3 of $\mathsf{ComputeAdvice}_\mathbf{r}$ we have $\|\mathbf{e}_\mathrm{F}\| \le 2^{-3\lambda/2}B'$. Also, since $\|\mathbf{e}_1\| \le 2^{-\lambda/2}B'$, from Step 4 we have $\|\mathbf{e}_\mathbf{S}\|, \|\mathbf{e}_\mathbf{E}\| \le \|\mathbf{e}\| \le \|\mathbf{e}_1\| + \|\mathbf{e}_\mathrm{F}\| \le B'$.

- Equality of $\bar{\mathbf{S}}_\mathbf{r}$ and $\mathbf{S}'_\mathbf{r}$. We have $\bar{\mathbf{S}}_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r})$ and $\mathbf{S}'_\mathbf{r} = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$. From Claim 4.7, we know that $\mathsf{Rnd}(\mathbf{S}_\mathbf{r}) = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ with overwhelming probability if $\|(\mathbf{S}_\mathbf{r} - (\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal))^\intercal\| = \|\mathbf{e}_\mathbf{S}\| \le \epsilon\frac{q}{p}$ for some negligible parameter ϵ. Note that $\|\mathbf{e}_\mathbf{S}\| \le B' \le \frac{1}{\lambda^{\omega(1)}}q/p$. Thus we have $\mathsf{Rnd}(\mathbf{S}_\mathbf{r}) = \mathsf{Rnd}(\mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S}^\intercal)$ with overwhelming probability.
- Validity of $\bar{\mathbf{S}}_\mathbf{r}$ and $\mathbf{S}'_\mathbf{r}$.
  First we show that $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}_\mathbf{r} + \mathbf{err}_1 \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil\log q\rceil}$ is a valid fhe encryption of

$\mathbf{s}_\mathbf{r}$. Consider the decryption equation

$$\begin{aligned}\mathbf{s}_\mathbf{r}^\intercal \bar{\mathbf{S}}_\mathbf{r} &= \mathbf{s}_\mathbf{r}^\intercal \mathbf{S}_\mathbf{r} + \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 \\ &= -\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})\mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}) - \mathbf{s}_\mathbf{r}(\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 \\ &= -\mathbf{s}_\mathbf{r}(\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + \mathbf{err}_{\mathsf{fhe}}^\intercal.\end{aligned}$$

which can be used to extract $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ via tensoring by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor \iota_{n+1})$ if $\|\mathbf{err}_{\mathsf{fhe}}\| \leq 2^{-\lambda}B < q/4$, where $\mathbf{err}_{\mathsf{fhe}} = \mathbf{s}_\mathbf{r}^\intercal \mathbf{err}_1 - \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})\mathsf{PRF}_2(\mathsf{sd}, \mathbf{r})$.
Note that Claim 4.6 implies $\|\mathbf{err}_1\| \leq \frac{q}{2p} \leq B'\lambda^{\omega(1)}$. Then, we have $\|\mathbf{err}_{\mathsf{fhe}}\| \leq B' \cdot B'\lambda^{\omega(1)} + O(m)\sigma' \leq \lambda^{\omega(1)}(B')^2 + O(m)2^{-\lambda}B' \leq 2^{-\lambda}B$.
Next, we show that $\mathbf{S}'_\mathbf{r} = \mathbf{S}_\mathbf{r} + \mathbf{e}_\mathbf{S} + \mathbf{err}_2 \in \mathbb{Z}_q^{(n+1)\times m(n+1)\lceil \log q \rceil}$ is a valid fhe encryption of $\mathbf{s}_\mathbf{r}$. Consider the decryption equation

$$\begin{aligned}\mathbf{s}_\mathbf{r}^\intercal \mathbf{S}'_\mathbf{r} &= \mathbf{s}_\mathbf{r}^\intercal \mathbf{S}_\mathbf{r} + \mathbf{s}_\mathbf{r}^\intercal (\mathbf{e}_\mathbf{S} + \mathbf{err}_2) \\ &= -\mathbf{s}_\mathbf{r}^\intercal (\mathsf{bits}(\mathbf{s}_\mathbf{r}) \otimes \mathbf{G}) + (\mathbf{err}'_{\mathsf{fhe}})^\intercal.\end{aligned}$$

which can be used to extract $\mathsf{bits}(\mathbf{s}_\mathbf{r})$ via tensoring by $\mathbf{G}^{-1}(\lfloor q/2 \rfloor \iota_{n+1})$ if $\left\|\mathbf{err}'_{\mathsf{fhe}}\right\| \leq 2^{-\lambda}B < q/4$, where $(\mathbf{err}'_{\mathsf{fhe}})^\intercal = \mathbf{s}_\mathbf{r}^\intercal (\mathbf{e}_\mathbf{S} + \mathbf{err}_2) - \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})\mathsf{PRF}_2(\mathsf{sd}, \mathbf{r})$ and $\left\|\mathbf{err}'_{\mathsf{fhe}}\right\| \leq 2^{-\lambda}B$. The analysis is similar to $\|\mathbf{e}_{\mathsf{fhe}}\|$, hence omitted.

- Error bound in $\mathbf{E}'_\mathbf{r}$. We have $\mathbf{E}'_\mathbf{r} = \mathbf{s}_\mathbf{r}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_\mathbf{r})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$, where $\mathbf{e}_{\mathsf{circ}}^\intercal = \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_\mathbf{E}^\intercal$. We have $\|\mathbf{e}_{\mathsf{circ}}\| \leq 2^{-\lambda}B' + B' \leq 2^{-\lambda}B$.

■

Step 2: HLL Bootstrapping On input $\mathbf{A}_{\mathsf{circ}}, \mathbf{S}'_\mathbf{r} = \mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r}) + \mathbf{err}_\mathbf{S}$, and $\mathbf{E}'_\mathbf{r} = \mathbf{s}_\mathbf{r}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_\mathbf{r})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$[5], it does the following[6].

1. *Defining circuits for homomorphic evaluation*. Define the following evaluation circuits/matrices corresponding to the function $\mathsf{RndPad}_\mathbf{A}(\cdot)$.

   - Define $\mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}} = \mathsf{MakeVEvalCkt}(\mathsf{RndPad}_\mathbf{A})$.

   - Compute $\mathrm{H}_\mathbf{A}^{\mathsf{RndPad}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}}) \in \mathbb{Z}^{(\mathrm{L}_\mathrm{S}+1)m\times m}$ and $\mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{circ}}, \mathsf{VEval}_{\mathsf{RndPad}_\mathbf{A}}, \mathbf{S}'_\mathbf{r}) \in \mathbb{Z}^{(\mathrm{L}_\mathrm{S}+1)m\times m}$.

   Here $\left\|(\mathrm{H}_\mathbf{A}^{\mathsf{RndPad}})^\intercal\right\|, \left\|(\mathrm{H}_{\mathbf{A},\mathbf{S}'_\mathbf{r}}^{\mathsf{RndPad}})^\intercal\right\| \leq 2^{O(\log^5 \lambda)}$.

[5] We replace $\bar{\mathbf{S}}_\mathbf{r}$ with $\mathbf{S}'_\mathbf{r}$ in the $\mathbf{E}'_\mathbf{r}$, since $\bar{\mathbf{S}}_\mathbf{r} = \mathbf{S}'_\mathbf{r}$ with overwhelming probability.
[6] Note that this doesn't straightforward follows from HLL. The error terms are different in the circular ciphertext $\mathbf{S}'_\mathbf{r}$. However, we can still get the final error with the desired bounds as analysed below.

2. *Compute the matrix homomorphically on* $\mathbf{E}'_{\mathbf{r}}$. We have

$$\begin{aligned}\mathbf{E}'_{\mathbf{r}} \cdot \mathrm{H}^{\mathsf{RndPad}}_{\mathbf{A},\mathbf{S}'_{\mathbf{r}}} &= \left(\mathbf{s}_{\mathbf{r}}^{\intercal}\left(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}'_{\mathbf{r}})) \otimes \mathbf{G}\right) + \mathbf{e}_{\mathsf{circ}}^{\intercal}\right) \mathrm{H}^{\mathsf{RndPad}}_{\mathbf{A},\mathbf{S}'_{\mathbf{r}}} \\ &= \mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{A}_{\mathsf{circ}}\mathrm{H}^{\mathsf{RndPad}}_{\mathbf{A}} - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) + \mathbf{err}^{\intercal}\end{aligned}$$

where, using the fact that depth of RndPad is $O(\log n \log\log q)$, we have

$$\begin{aligned}\|\mathbf{err}\| \leq &(m+2)^{O(\log n \log\log q)} \cdot m \cdot (\|\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})^{\intercal}\| + \|(\mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2))^{\intercal}\|) \\ &+ \|\mathbf{e}_{\mathsf{circ}}\| \cdot 2^{O(\log^5 \lambda)}.\end{aligned}$$

Assuming $\|\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})^{\intercal}\|, \|(\mathbf{s}_{\mathbf{r}}^{\intercal}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2))^{\intercal}\|, \|\mathbf{e}_{\mathsf{circ}}\| \leq 2^{-\lambda}B$, we get $\|\mathbf{err}\| \leq 2^{-\lambda/2}B$.

3. *Output* $\mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{A}_C - \mathsf{RndPad}_{\mathbf{A}}(\mathbf{s}_{\mathbf{r}}) + \mathbf{err}^{\intercal}$, where $\mathbf{A}_C = \mathbf{A}_{\mathsf{circ}}\mathrm{H}^{\mathsf{RndPad}}_{\mathbf{A}}$.

Restoring the structure. We note that combining the outputs from Section 4.3.2 and 4.3.3, we get

$$\mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{A}_C - C(\mathbf{x})\mathbf{s}_{\mathbf{r}}^{\intercal}\mathbf{G} + \mathbf{err}^{\intercal}, \quad \text{where } \|\mathbf{err}\| \leq 2^{-\lambda/2}B.$$

4.4 CIPHERTEXT POLICY ABE FOR UNBOUNDED DEPTH CIRCUITS

In this section we present CP-ABE for unbounded depth and size circuits.

4.4.1 Construction

We construct a CPABE scheme for circuit class $C = \{C : \{0,1\}^{\mathrm{L}} \to \{0,1\}\}$.

$\mathsf{Setup}(1^{\lambda}, 1^{\mathrm{L}})$. The setup algorithm does the following.

1. Set $\mathrm{L_S} = m(n+1)^2\lceil\log q\rceil^2$, $\mathrm{L_T} = m(n+1)(m(n+1)\lceil\log q\rceil + \lambda)\lceil\log q\rceil$, and $\ell = \mathrm{L_S}(1 + m\log q)$.

2. Sample: $(\mathbf{B}, \tau) \leftarrow \mathsf{TrapGen}(1^{(n+1)(m+2)}, 1^{(n+1)(m+2)w}, q)$, $\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{A}_0 \leftarrow \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{A}_{\mathsf{path}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L_T}+1)m}$, $\mathbf{A}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}+1)m}$, $\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L_S}+1)m}$, $\mathbf{u} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma_{\mathbf{r}}}$, where $w \in O(\log q)$.

3. Output $\mathsf{mpk} := (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u})$[7] and $\mathsf{msk} := \tau$.

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{x})$. The key generation algorithm does the following.

1. For $1 \le j \le m-1$, sample $\mathbf{r}_j \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{r}}}$. Set $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_m)$ such that $\sum_{j\in[m]} \mathbf{r}_j = 1$.

2. Define function F as in Figure 4.1 using $\mathbf{r}$, $\mathsf{VEval}_{\mathrm{F}} = \mathsf{MakeVEvalCkt}(n, m, q, \mathrm{F}) \in \mathbb{Z}_q^{n+1\times\ell}$ and compute $\mathbf{H}_{\mathrm{F}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{path}}, \mathsf{VEval}_{\mathrm{F}})$. Set $\mathbf{A}_{\mathbf{r}} = \mathbf{A}_{\mathsf{path}}\mathbf{H}_{\mathrm{F}}$.

3. Sample
$$\mathbf{K} \leftarrow \mathbf{B}_\tau^{-1}\begin{pmatrix}\mathbf{A}_0\mathbf{r} \\ (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G})\otimes\mathbf{r} \\ \mathbf{A}_{\mathbf{r}}\end{pmatrix}$$

4. Output $\mathsf{sk} := (\mathbf{K}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, C, \mu)$. The encryption algorithm does the following.

1. Sample $\mathbf{s}_0 \leftarrow \mathcal{D}^{(n+1)}_{\mathbb{Z},\sigma_{\mathbf{s}}}$, $\mathbf{s}_j \leftarrow \mathcal{D}^{m}_{\mathbb{Z},\sigma_{\mathbf{s}}}$ for $1 \le j \le n$, $\bar{\mathbf{t}} \leftarrow \mathcal{D}^{n}_{\mathbb{Z},\sigma_{\mathbf{s}}}$, $\mathbf{e}'_0 \leftarrow \mathcal{D}^{(n+1)w}_{\mathbb{Z},\sigma_0}, \mathbf{e}'_{\mathsf{att}} \leftarrow \mathcal{D}^{m(n+1)w}_{\mathbb{Z},\sigma_{\mathsf{att}}}, \mathbf{e}'_1 \leftarrow \mathcal{D}^{(n+1)w}_{\mathbb{Z},\sigma_1}$ $\mathbf{e}_{\mathsf{msg}} \leftarrow \mathcal{D}^{m}_{\mathbb{Z},\sigma_{\mathsf{msg}}}$. Set $\mathbf{s} = (\mathbf{s}_1^\intercal, \ldots, \mathbf{s}_n^\intercal, -\mathbf{1}_{\mathbf{m}})^\intercal$, $\mathbf{t} = (\bar{\mathbf{t}}^\intercal, -1)^\intercal, \mathbf{e}_{\mathbf{B}}^\intercal = ((\mathbf{e}'_0)^\intercal, (\mathbf{e}'_{\mathsf{att}})^\intercal, (\mathbf{e}'_1)^\intercal)$.

2. Compute
$$\mathbf{c}_{\mathbf{B}}^\intercal := (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \quad \mathsf{ct}_{\mathsf{msg}}^\intercal := \mathbf{s}_0^\intercal\mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u}\otimes\mathbf{I}_m) + \mu\cdot\mathbf{g}^\intercal + \mathbf{e}_{\mathsf{msg}}^\intercal$$
where $\mathbf{A}_C \leftarrow \mathsf{UEvalC}(\mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{att}}, C)$.

3. For $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ and $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_{m(n+1)\lceil\log q\rceil+\lambda})$, compute fhe ciphertext as follows.
$$\bar{\mathbf{A}}'_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}, \quad \mathbf{e}'_{\mathsf{fhe}} \leftarrow \mathcal{D}^{m}_{\mathbb{Z},\sigma'}, \quad \mathbf{A}_{\mathbf{t}} := (\bar{\mathbf{A}}'_{\mathsf{fhe}} \;\; \bar{\mathbf{t}}^\intercal\bar{\mathbf{A}}'_{\mathsf{fhe}} + (\mathbf{e}'_{\mathsf{fhe}})^\intercal)^\intercal,$$
$$\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd})\otimes\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(m(n+1)\lceil\log q\rceil+\lambda)}$$
where $\mathbf{R}_i \leftarrow \{0,1\}^{m\times m}$ for all $1 \le i \le (m(n+1)\lceil\log q\rceil + \lambda)$.
Set $\mathrm{L_T} = m(n+1)(m(n+1)\lceil\log q\rceil + \lambda)\lceil\log q\rceil$ so that $\mathsf{bits}(\mathbf{T}) \in \{0,1\}^{\mathrm{L_T}}$.

[7] All the algorithms take mpk implicitly.

4. Computes circular encoding as follows.

$$\mathbf{e}_{\mathrm{D}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}^{(\mathrm{L}_{\mathrm{T}}+1)m}, \quad \mathbf{D} = \mathbf{t}^{\intercal}\left(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}\right) + \mathbf{e}_{\mathrm{D}}^{\intercal}.$$

5. Output $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{msg}}, \mathbf{T}, \mathbf{D})$.

$\mathsf{Dec}(\mathsf{sk}, \mathbf{x}, \mathsf{ct}, C)$. The decryption algorithm does the following.

1. Parse $\mathsf{sk} = (\mathbf{K}, \mathbf{r}^{\intercal})$ and $\mathsf{ct} = (\mathbf{A}_{\mathbf{t}}, \mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{msg}}, \mathbf{T}, \mathbf{D})$.
2. Compute $\mathsf{ct}_{\mathbf{B}}^{\intercal} \cdot \mathbf{K}$ and parse it as $(\mathbf{c}_0^{\intercal} \quad \mathbf{c}_{\mathsf{att}}^{\intercal} \quad \mathbf{c}_1^{\intercal})$ where $\mathbf{c}_0 \in \mathbb{Z}_q$, $\mathbf{c}_{\mathsf{att}} \in \mathbb{Z}_q^{(\mathrm{L}+1)m}$, $\mathbf{c}_1 \in \mathbb{Z}_q^{\ell}$.
3. Compute $(\mathbf{S}'_{\mathbf{r}}, \mathbf{E}'_{\mathbf{r}}) \leftarrow \mathsf{ComputeAdvice}_{\mathbf{r}}(\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{A}_{\mathsf{path}}, \mathbf{T}, \mathbf{D}, \mathbf{c}_1^{\intercal})$.
4. Set $\mathbf{c}_{\mathsf{circ}}^{\intercal} \coloneqq \mathbf{E}'_{\mathbf{r}}$ and compute $\mathbf{c}_{C,\mathbf{x}}^{\intercal} \leftarrow \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}_{\mathsf{att}}^{\intercal}, \mathbf{A}_{\mathsf{circ}}, \mathbf{c}_{\mathsf{circ}}^{\intercal}, C, \mathbf{x}, \mathbf{S}'_{\mathbf{r}})$.
5. Output $\mathsf{ct}_{\mathsf{msg}}^{\intercal} \cdot \mathbf{r} - \mathbf{c}_0^{\intercal} - \mathbf{c}_{C,\mathbf{x}}^{\intercal} \cdot \mathbf{u}$

Parameters. We set our parameters as follows.

$$n = \mathrm{poly}(\lambda),\ q = 2^{14\lambda}\lambda^{\omega(1)},\ p = 2^{10\lambda},\ m = O(n \log q),\ B = 2^{7\lambda},$$

$$\sigma_{\mathbf{r}} = \mathrm{poly}(\lambda),\ \sigma_{\mathbf{s}} = \sigma' = 2^{\lambda},\ \sigma_0 = 2^{7\lambda},\ \sigma_1 = 2^{2\lambda}\lambda^{\omega(1)},\ \sigma_{\mathsf{msg}} = 2^{5\lambda},\ \sigma_{\mathsf{att}} = 2^{5\lambda},$$

$$\tau = O(\sqrt{(n+1)(m+2)\log q}),\ \chi_0 = 2^{6\lambda}\lambda^{\omega(1)},\ \chi_1 = 2^{2\lambda}\lambda^{\omega(1)},\ \chi_{\mathsf{att}} = 2^{3\lambda}$$

where $\chi_0, \chi_1, \chi_{\mathsf{att}}$ appear only in the security proof.

Efficiency. Using the above set parameters, we have

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, L), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, L), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda).$$

Correctness We show the correctness step by step below.

– First, we note that

$$\mathsf{ct}_{\mathbf{B}} \cdot \mathbf{K} = \left((\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal\right)\mathbf{B}_\tau^{-1}\begin{pmatrix}\mathbf{A}_0\mathbf{r} & & \\ & (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G})\otimes\mathbf{r} & \\ & & \mathbf{A}_{\mathbf{r}}\end{pmatrix}$$

$$(\mathbf{c}_0^\intercal \;\; \mathbf{c}_{\mathsf{att}}^\intercal \;\; \mathbf{c}_1^\intercal) = (\mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}) + \mathbf{e}_0^\intercal \qquad \mathbf{s}^\intercal((\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G})\otimes\mathbf{r}) + \mathbf{e}_{\mathsf{att}}^\intercal \qquad \mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}}) + \mathbf{e}_1^\intercal)$$

where $\|\mathbf{e}_0\| \le \tau\left\|\mathbf{e}_0'\right\|, \|\mathbf{e}_{\mathsf{att}}\| \le \tau\left\|\mathbf{e}_{\mathsf{att}}'\right\|, \|\mathbf{e}_1\| \le \tau\left\|\mathbf{e}_1'\right\|$.

– Next, we have $\|\mathbf{s}_{\mathbf{r}}\| \le \sigma_{\mathbf{s}}\sigma_{\mathbf{r}} \cdot m \le 2^\lambda \cdot \mathrm{poly}(\lambda) \cdot m \le B$ and $q/p = 2^{3\lambda}\lambda^{\omega(1)}$.

– Correctness of $\mathsf{ComputeAdvice}_{\mathbf{r}}$. From our parameter setting, we have $\|\mathbf{e}_1\| \le \tau \cdot 2^{2\lambda}\lambda^{\omega(1)}\sqrt{\lambda}, \|\mathbf{e}_{\mathbf{D}}\|, \left\|\mathbf{e}_{\mathsf{fhe}}'\right\| \le 2^\lambda\sqrt{\lambda}$, where $\mathbf{e}_1, \mathbf{e}_{\mathbf{D}}, \mathbf{e}_{\mathsf{fhe}}'$ are the error terms present in the input to procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}$. Next, we observe the following.

  – The error term in Step 3 is $\mathbf{e}_{\mathrm{F}}^\intercal = (\mathbf{e}')^\intercal\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathrm{D}}^\intercal\mathbf{H}_{\mathrm{F},\mathbf{T}}$. We have $\|\mathbf{e}_{\mathrm{F}}\| \le 2^{\lambda+\mathrm{poly}(\log\lambda)}\sqrt{\lambda}$.

  – The error term in Step 4 is $\mathbf{e}^\intercal = \mathbf{e}_1^\intercal - \mathbf{e}_{\mathrm{F}}^\intercal \in \mathbb{Z}^{1\times\ell}$. We have $\|\mathbf{e}\| \le 2^{3\lambda}$. This implies $\|\mathbf{e}_{\mathbf{S}}\|, \|\mathbf{e}_{\mathbf{E}}\| \le 2^{3\lambda}$.

  – The decryption error in $\bar{\mathbf{S}}_{\mathbf{r}}$ is $\mathbf{err}_{\mathsf{fhe}} = \mathbf{s}_{\mathbf{r}}^\intercal\mathbf{err}_1 - \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})\mathsf{PRF}_2(\mathsf{sd}, \mathbf{r})$. We have $\|\mathbf{err}_{\mathsf{fhe}}\| \le O(m^2) \cdot \mathrm{poly}(\lambda)2^{5\lambda}\lambda^{\omega(1)} + m \cdot 2^\lambda \le 2^{6\lambda}$, where we use the fact that the error introduced by $\mathsf{Rnd}(\cdot)$ is bounded by $q/2p$.

  – The decryption error in $\mathbf{S}_{\mathbf{r}}'$ is $(\mathbf{err}_{\mathsf{fhe}}')^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2) - \mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})\mathsf{PRF}_2(\mathsf{sd}, \mathbf{r})$. We have $\left\|\mathbf{err}_{\mathsf{fhe}}'\right\| \le 2^{5\lambda} + O(m^2) \cdot \mathrm{poly}(\lambda)2^{5\lambda}\lambda^{\omega(1)} + m \cdot 2^\lambda \le 2^{6\lambda}$, where we use the fact that the error introduced by $\mathsf{Rnd}(\cdot)$ is bounded by $q/2p$ and $\|\mathbf{s}_{\mathbf{r}}\| \cdot \|\mathbf{e}_{\mathbf{S}}\| \le 2^{5\lambda}$.

  – Next, we note that $\|\mathbf{e}_{\mathbf{S}}\| \le 2^{3\lambda} \le \frac{1}{2^\lambda} \cdot \frac{q}{p}$. This implies that $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}}) = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}} + \mathbf{e}_{\mathbf{S}}^\intercal)$ with overwhelming probability.

  – The error term in $\mathbf{E}_{\mathbf{r}}'$ is $\mathbf{e}_{\mathsf{circ}}^\intercal = \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{E}}^\intercal$. We have $\|\mathbf{e}_{\mathsf{circ}}\| \le 2^\lambda + 2^{3\lambda} \le 2^{-\lambda}B$.

  Using the above observations, we note that the procedure $\mathsf{ComputeAdvice}_{\mathbf{r}}$ outputs $\mathbf{S}_{\mathbf{r}}'$ and $\mathbf{E}_{\mathbf{r}}'$ where $\mathbf{E}_{\mathbf{r}}'$ encode $\mathbf{S}_{\mathbf{r}}'$ with all but negligible probability.

– Next, note that for $\mathbf{c}_{\mathsf{att}}^\intercal = \mathbf{s}^\intercal((\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G})\otimes\mathbf{r}) + \mathbf{e}_{\mathsf{att}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal)\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal$, where $\|\mathbf{s}_{\mathbf{r}}\|, \|\mathbf{e}_{\mathsf{att}}\| \le B$. So, by the correctness of $\mathsf{UEvalCX}$, we have

$$\mathsf{ct}_{C,\mathbf{x}}^\intercal = \mathbf{s}_{\mathbf{r}}^\intercal(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^\intercal = \mathbf{s}(\mathbf{I}_{n+1}\otimes\mathbf{r})(\mathbf{A}_C - C(\mathbf{x})\mathbf{G}) + \mathbf{e}_{C,\mathbf{x}}^\intercal,$$

where, from Step 2 of procedure HLL Bootstrapping, it follows that $\left\|\mathbf{e}_{C,\mathbf{x}}\right\| \le (m +$

$2)^{O(\log n \log\log q)} \cdot m \cdot (\|\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r})^\intercal\| + \|(\mathbf{s}_\mathbf{r}^\intercal(\mathbf{e_S} + \mathbf{err}_2))^\intercal\|) + \|\mathbf{e}_{\mathsf{circ}}\| \cdot 2^{O(\log^5 \lambda)}$. From our parameter setting and the error analysis above, $\left\|\mathbf{e}_{C,\mathbf{x}}\right\| \leq 2^{6\lambda} \leq 2^{-\lambda/2}B$.

– Next, for $C, \mathbf{x}$ such that $C(\mathbf{x}) = 0$, we have the following.

$$\begin{aligned}
&\mathsf{ct}_{\mathsf{msg}}^\intercal \cdot \mathbf{r} - \mathbf{c}_0^\intercal - \mathsf{ct}_{C,\mathbf{x}}^\intercal \cdot \mathbf{u} \\
&= (\mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mu \cdot \mathbf{g}^\intercal + \mathbf{e}_{\mathsf{msg}}^\intercal) \cdot \mathbf{r} - \mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}) - \mathbf{e}_0^\intercal \\
&- \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r})\mathbf{A}_C\mathbf{u} - \mathbf{e}_{C,\mathbf{x}}^\intercal\mathbf{u} \\
&= \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m)\mathbf{r} + \mu \cdot \mathbf{g}^\intercal\mathbf{r} + \mathbf{e}_{\mathsf{msg}}^\intercal\mathbf{r} - \mathbf{e}_0^\intercal - \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m)(\mathbf{1} \otimes \mathbf{r}) - \mathbf{e}_{C,\mathbf{x}}^\intercal\mathbf{u} \\
&\approx \mu\lfloor q/2 \rceil + \mathbf{e}^\intercal.
\end{aligned}$$

where $\mathbf{e}^\intercal = \mathbf{e}_{\mathsf{msg}}^\intercal\mathbf{r} - \mathbf{e}_0^\intercal - \mathbf{e}_{C,\mathbf{x}}^\intercal\mathbf{u}$ and $\|\mathbf{e}\| \leq 2^{5\lambda}\sqrt{\lambda} \cdot m \cdot \mathrm{poly}(\lambda) - O(m) \cdot 2^\lambda\sqrt{\lambda} - 2^{6\lambda} \cdot \mathrm{poly}(\lambda) \leq 2^{7\lambda} < q/4$, which is within the bounds of rounding. Hence the decryption outputs μ with all but negligible probability.

4.4.2 Our Assumption

We detail our assumptions next. We introduce a circular version of the Tensor LWE assumption [169] in similar spirit to the Evasive Circular assumption [122].

Assumption 4.13 (Circular Tensor LWE). Let $\mathbf{s} = (\mathbf{s}_1^\intercal, \cdots \mathbf{s}_n^\intercal, -\mathbf{1}_m)^\intercal$ where $\mathbf{s}_i \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_\mathbf{s}}^m$, $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_m)$, where $\mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_\mathbf{r}}$ such that $\sum_{i\in[m]} \mathbf{r}_i = 1$. For $i \in [Q]$, set $\mathbf{s}_{\mathbf{r}_i}^\intercal = \mathbf{s}^\intercal(\mathbf{I}_{n+1} \otimes \mathbf{r}_i) = (\bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal, -1) \in \mathbb{Z}^{n+1}$. Suppose

$$\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \mathbf{a}_{\mathsf{att}}^\intercal \end{pmatrix} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}+1)m},\ \bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \mathbf{A}_{\mathsf{circ}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{circ}} \\ \mathbf{a}_{\mathsf{circ}}^\intercal \end{pmatrix} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathsf{circ}}}$$

and for $i \in [Q]$, $\sigma_{\mathsf{att}}, \sigma' << q$, suppose

$$\mathbf{e}_{\mathsf{att},i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{att}}}^{(\mathrm{L}+1)m},\ \mathbf{e}_{\mathsf{fhe},i} \leftarrow [-\sigma',\ \sigma']^m,\ \mathbf{e}_{\mathsf{circ},i} \leftarrow [-\sigma',\ \sigma']^{m_{\mathsf{circ}}}$$

$$\boldsymbol{\delta}_{\mathsf{att},i} \leftarrow \mathbb{Z}_q^{(\mathrm{L}+1)m},\ \boldsymbol{\delta}_{\mathsf{fhe},i} \leftarrow \mathbb{Z}_q^m,\ \boldsymbol{\delta}_{\mathsf{circ},i} \leftarrow \mathbb{Z}_q^{m_{\mathsf{circ}}},\ \Delta_i \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathbf{S}}$$

$$\mathbf{R}_i \leftarrow \{0,1\}^{m\times m_\mathbf{S}}, \mathbf{S}_{\mathbf{r}_i} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe},i}^\intercal \end{pmatrix}\mathbf{R}_i - \mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m_\mathbf{S}}$$

where the parameters are functions of λ.

For all $\mathbf{x}_1, \ldots, \mathbf{x}_Q \in \{0,1\}^L$, the *circular tensor* LWE *assumption* states that

$$\left(1^\lambda, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \left\{\begin{matrix} \bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal(\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^\intercal, \ \bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe},i}^\intercal, \\ \mathbf{S}_{\mathbf{r}_i}, \quad \bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal(\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_i)) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^\intercal \ , \ \mathbf{r}_i \end{matrix}\right\}_{i\in[Q]}\right)_{\lambda\in\mathbb{N}} \tag{4.5}$$

$$\approx \left(1^\lambda, \ \mathbf{A}_{\mathsf{att}}, \ \mathbf{A}_{\mathsf{circ}}, \ \bar{\mathbf{A}}_{\mathsf{fhe}}, \ \left\{\boldsymbol{\delta}_{\mathsf{att},i}^\intercal, \ \ \boldsymbol{\delta}_{\mathsf{fhe},i}^\intercal, \ \Delta_i, \ \ \boldsymbol{\delta}_{\mathsf{circ},i}^\intercal\right\}_{i\in[Q]}\right)_{\lambda\in\mathbb{N}} \tag{4.6}$$

Remark 16. Here, we choose $\mathbf{e}_{\mathsf{fhe}}$ and $\mathbf{e}_{\mathsf{circ}}$ from uniform distribution over an interval rather than from usual Gaussian distribution. This change makes little difference to the assumption, since when the modulus q is super-polynomial, we can reduce the problem with Gaussian distribution to the one with uniform distribution with (superpolynomially) larger width and vice versa, due to the smudging (Lemma 2.3).

4.4.3 Security Proof

Theorem 4.14. *Assuming evasive LWE (assumption 2.6) for the sampler class $\mathcal{SC}$ induced by the sampler as defined in Equation (4.7), circular tensor LWE (assumption 4.13) and LWE (assumption 2.5), there exists a CP-ABE scheme for unbounded depth and unbounded size circuits which satisfies very selective security (Definition 2.7).*

Proof. Suppose the ABE adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries C and $\mathbf{x}_1, \ldots, \mathbf{x}_Q$ such that $C(\mathbf{x}_1) = \cdots = C(\mathbf{x}_Q) = 1$. The view of the adversary when bit μ is encoded is:

$$\left(\begin{matrix} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\ \mathbf{c}_{\mathbf{B}}^\intercal = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \ \ \mathsf{ct}_{\mathsf{msg}}^\intercal = \mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C \mathbf{u} \otimes \mathbf{I}_m) + \mu \cdot \mathbf{g}^\intercal + \mathbf{e}_{\mathsf{msg}}^\intercal \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \ \ \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^\intercal, \quad \{\mathbf{K}_i, \quad \mathbf{r}_i\}_{i\in[Q]} \end{matrix}\right)$$

where for $i \in [Q]$, we have

$$\mathbf{K}_i = \mathbf{B}_\tau^{-1} \begin{pmatrix} \mathbf{A}_0 \mathbf{r}_i & & \\ & (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^\intercal) \otimes \mathbf{G}) \otimes \mathbf{r}_i & \\ & & \mathbf{A}_{\mathbf{r}_i} \end{pmatrix}$$

Note that to prove $\mathsf{ct}_{\mathsf{msg}}^\intercal = \mathbf{s}_0^\intercal \mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C \mathbf{u} \otimes \mathbf{I}_m) + \mu \cdot \mathbf{g}^\intercal + \mathbf{e}_{\mathsf{msg}}^\intercal$ is indistinguishable

from random, it suffices to show that $\mathbf{c}_{\mathsf{msg}}^{\intercal} = \mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\intercal}$. We prove the pseudorandomness of the above distribution by replacing $\mathsf{ct}_{\mathsf{msg}}^{\intercal}$ with $\mathbf{c}_{\mathsf{msg}}^{\intercal}$.

We invoke the Theorem 2.7 variant of evasive LWE hardness assumption for a matrix $\mathbf{B}$ with Gaussian parameter τ and a sampler Samp that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2))$ defined as follows.

$$\mathsf{aux}_1 = \left(\begin{array}{c} \mathsf{ct}_{\mathsf{msg}} = (\mathbf{s}_0,\ \mathbf{s}) \begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m \end{pmatrix},\ \mathbf{T} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{t}}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \bar{\mathbf{e}}_{\mathsf{fhe}}^{\intercal} \end{pmatrix} \mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \\ \mathbf{D} = \mathbf{t}^{\intercal}(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{D}}^{\intercal} \end{array} \right)$$

$$\mathsf{aux}_2 = (\mathbf{x}_1, \ldots, \mathbf{x}_Q, C, \mathsf{coins}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_Q, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{u})$$

$$\mathbf{S} = (\mathbf{s}_0^{\intercal} \mid \mathbf{s}^{\intercal} \mid \mathbf{t}^{\intercal})$$

$$\mathbf{P}_0 = (\mathbf{A}_0\mathbf{r}_1\|, \ldots, \|\mathbf{A}_0\mathbf{r}_Q)$$

$$\mathbf{P}_1 = ((\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^{\intercal}) \otimes \mathbf{G}) \otimes \mathbf{r}_1\|, \ldots, \|(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}^{\intercal}) \otimes \mathbf{G}) \otimes \mathbf{r}_Q)$$

$$\mathbf{P}_2 = (\mathbf{A}_{\mathbf{r}_1}\|, \ldots, \|\mathbf{A}_{\mathbf{r}_Q})$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_0 & & \\ & \mathbf{P}_1 & \\ & & \mathbf{P}_2 \end{pmatrix} \tag{4.7}$$

We set $\mathsf{aux}_0 = (\mathbf{x}_1, \ldots, \mathbf{x}_Q, C, \mathsf{coins}_{\mathcal{A}})$.

By applying evasive LWE, it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}) \\ \mathbf{c}_{\mathbf{B}}^{\intercal} = (\mathbf{s}_0^{\intercal} \mid \mathbf{s}^{\intercal} \mid \mathbf{t}^{\intercal})\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\intercal},\ \mathbf{c}_{\mathsf{msg}}^{\intercal} = \mathbf{s}_0^{\intercal}\mathbf{A}_0 + \mathbf{s}^{\intercal}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\intercal}, \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^{\intercal}(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^{\intercal}, \\ \{\mathbf{c}_{0,i}^{\intercal} = \mathbf{s}_0^{\intercal}(\mathbf{A}_0\mathbf{r}_i) + \mathbf{e}_{0,i}^{\intercal}\ ,\ \mathbf{c}_{\mathsf{att},i}^{\intercal} = \mathbf{s}_{\mathbf{r}_i}^{\intercal}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^{\intercal}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^{\intercal}\ ,\ \mathbf{c}_{1,i}^{\intercal} = \mathbf{t}^{\intercal}\mathbf{A}_{\mathbf{r}_i} + \mathbf{e}_{1,i}^{\intercal}\ ,\ \mathbf{r}_i\}_{i \in [Q]} \end{array} \right)$$

where $\mathbf{e}_{0,i}^{\intercal} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_0}$, $\mathbf{e}_{\mathsf{att},i}^{\intercal} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{att}}}^{(\mathrm{L}+1)m}$, $\mathbf{e}_{1,i}^{\intercal} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{\ell}$ for $i \in [Q]$.

Hyb_0. This is the distribution as specified in Section 4.4.3.

Hyb_1. This hybrid is same as Hyb_0, except we compute $\mathbf{c}_{1,i}^\intercal$ as a function of $\mathbf{T}, \mathbf{D}, \mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}$.

Specifically, we compute

$$\mathbf{c}_{1,i}^\intercal \coloneqq \mathbf{D}\ \mathbf{H}_{\mathrm{F},\mathbf{T}} + (\mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}) + \mathbf{e}_{1,i}^\intercal \text{ , for } i \in [Q], \tag{4.8}$$

We claim that Hyb_0 and Hyb_1 are statistically indistinguishable. To see this, note that:

- $\mathbf{t}^\intercal(\mathbf{A}_{\mathbf{r}_i}) = \mathbf{D}\ \mathbf{H}_{\mathrm{F},\mathbf{T}} + (\mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}) + \mathbf{e}_{\mathrm{F},i}^\intercal$, where $\left\|\mathbf{e}_{\mathrm{F},i}\right\| \le 2^{\lambda+\mathrm{poly}(\log\lambda)}\sqrt{\lambda} \le 2^{2\lambda}$, by our parameter setting.
- We have $\mathbf{e}_{1,i}^\intercal \approx_s \mathbf{e}_{\mathrm{F},i}^\intercal + \mathbf{e}_{1,i}^\intercal$ by noise flooding (Theorem 2.3) since $\lambda^{\omega(1)}\left\|\mathbf{e}_{\mathrm{F},i}\right\| \le \lambda^{\omega(1)}2^{2\lambda} = \chi_1$.

Thus it suffices to show pseudorandomness of the following, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\ \mathbf{c}_{\mathbf{B}}^\intercal = (\mathbf{s}_0^\intercal \mid \mathbf{s}^\intercal \mid \mathbf{t}^\intercal)\mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal,\ \mathbf{c}_{\mathsf{msg}}^\intercal = \mathbf{s}_0^\intercal\mathbf{A}_0 + \mathbf{s}^\intercal(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^\intercal, \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^\intercal, \\ \{\mathbf{c}_{0,i}^\intercal = \mathbf{s}_0^\intercal(\mathbf{A}_0\mathbf{r}_i) + \mathbf{e}_{0,i}^\intercal\ ,\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\ , \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^\intercal,\ \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_2. This hybrid is same as Hyb_1, except we compute

$$\mathbf{c}_{0,i}^\intercal \coloneqq \mathbf{c}_{\mathsf{msg}}^\intercal \cdot \mathbf{r}_i - \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}_{\mathsf{att},i}^\intercal, \mathbf{A}_{\mathsf{circ}}, \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^\intercal, C, \mathbf{x}, \mathbf{S}'_{\mathbf{r}_i})\mathbf{u} - \mathbf{s}_{\mathbf{r}_i}^\intercal\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i}$$

where $\mathbf{S}'_{\mathbf{r}_i} = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^\intercal)$. Note that since $\left\|\mathbf{e}_{\mathbf{S},i}\right\| \le \left\|\mathbf{e}_{1,i}\right\| \le 2^{2\lambda}\lambda^{\omega(1)} \le \frac{1}{2^\lambda}q/p$, Claim 4.7 implies that $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^\intercal) = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$ with overwhelming probability, where $\mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$ is encoded in $\mathbf{E}_{\mathbf{r}_i}$.

We claim that Hyb_1 and Hyb_2 are statistically indistinguishable. To see this, note

that:

- we have $\left\|\mathbf{s}_{\mathbf{r}_i}\right\| \le B$ and $\left\|\mathbf{e}_{\mathsf{att},i}\right\| \le 2^{5\lambda}\sqrt{\lambda} \le B$.
- Next, by the correctness of $\mathsf{UEvalCX}$, we have

  $$\begin{aligned}
  &\mathbf{c}_{\mathsf{msg}}^{\mathsf{T}} \cdot \mathbf{r}_i - \mathsf{UEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathbf{c}_{\mathsf{att},i}^{\mathsf{T}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}}, C, \mathbf{x}, \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^{\mathsf{T}})\mathbf{u} - \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} \\
  &= (\mathbf{s}_0^{\mathsf{T}}\mathbf{A}_0 + \mathbf{s}^{\mathsf{T}}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\mathsf{T}}) \cdot \mathbf{r}_i - \mathbf{s}(\mathbf{I}_{n+1} \otimes \mathbf{r})(\mathbf{A}_C - C(\mathbf{x})\mathbf{G})\mathbf{u} + \mathbf{e}_{C,\mathbf{x}}^{\mathsf{T}}\mathbf{u} - \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} \\
  &= (\mathbf{s}_0^{\mathsf{T}}\mathbf{A}_0 + \mathbf{s}^{\mathsf{T}}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\mathsf{T}}) \cdot \mathbf{r}_i - \mathbf{s}^{\mathsf{T}}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m)(\mathbf{1} \otimes \mathbf{r}) + \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} + \mathbf{e}_{C,\mathbf{x}}^{\mathsf{T}}\mathbf{u} - \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} \\
  &= \mathbf{s}_0^{\mathsf{T}}\mathbf{A}_0\mathbf{r}_i + \mathbf{e}_{\mathsf{msg}}^{\mathsf{T}} \cdot \mathbf{r}_i + \mathbf{e}_{C,\mathbf{x}}^{\mathsf{T}}\mathbf{u}
  \end{aligned}$$

  where $\left\|\mathbf{e}_{C,\mathbf{x}}\right\| \le (m+2)^{O(\log n \log\log q)} \cdot m \cdot (\|\mathsf{PRF}_1(\mathsf{sd}, \mathbf{r}_i)^{\mathsf{T}}\| + \|(\mathbf{s}_{\mathbf{r}}^{\mathsf{T}}(\mathbf{e}_{\mathbf{S}} + \mathbf{err}_2))^{\mathsf{T}}\|) + \|\mathbf{e}_{\mathsf{circ}}\| \cdot 2^{O(\log^5 \lambda)} \le 2^{6\lambda}$, by our parameter setting.
- We have $\mathbf{e}_{0,i}^{\mathsf{T}} \approx_s \mathbf{e}_{\mathsf{msg}}^{\mathsf{T}}\mathbf{r}_i + \mathbf{e}_{C,\mathbf{x}}^{\mathsf{T}}\mathbf{u} + \mathbf{e}_{0,i}^{\mathsf{T}}$ by noise flooding (Theorem 2.3) since $\left\|\mathbf{e}_{\mathsf{msg}}\right\| \|\mathbf{r}_i\| + \left\|\mathbf{e}_{C,\mathbf{x}}\right\| \|\mathbf{u}\| \le 2^{7\lambda}\lambda^{\omega(1)} = \chi_0$.

Thus it suffices to show pseudorandomness of the following, given aux_0,

$$\left(\begin{array}{c}
\mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}), \\
\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} = (\mathbf{s}_0^{\mathsf{T}} \mid \mathbf{s}^{\mathsf{T}} \mid \mathbf{t}^{\mathsf{T}})\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}},\ \mathbf{c}_{\mathsf{msg}}^{\mathsf{T}} = \mathbf{s}_0^{\mathsf{T}}\mathbf{A}_0 + \mathbf{s}^{\mathsf{T}}(\mathbf{A}_C\mathbf{u} \otimes \mathbf{I}_m) + \mathbf{e}_{\mathsf{msg}}^{\mathsf{T}}, \\
\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\ \ \mathbf{D} = \mathbf{t}^{\mathsf{T}}(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^{\mathsf{T}}, \\
\{(\mathbf{c}_{0,i}^{\mathsf{T}})' = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}\mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^{\mathsf{T}} = \mathbf{s}_{\mathbf{r}_i}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^{\mathsf{T}}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^{\mathsf{T}},\ \mathbf{S}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{S},i}^{\mathsf{T}},\ \mathbf{E}_{\mathbf{r}_i} + \mathbf{e}_{\mathbf{E},i}^{\mathsf{T}},\ \mathbf{r}_i\}_{i \in [Q]}
\end{array}\right)$$

Hyb_3. This hybrid is same as Hyb_2, except we sample $\mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w}$, $\mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m$.

Note that the LWE secret $\mathbf{s}_0$ does not appear anywhere else. We have $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$, via the LWE assumption.

Thus it suffices to show pseudorandomness of the following distribution, given

aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}, \mathbf{D} = \mathbf{t}^\intercal(\mathbf{A}_{\mathsf{path}} - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G}) + \mathbf{e}_{\mathrm{D}}^\intercal, \\ \{(\mathbf{c}_{0,i}^\intercal)' = \mathbf{s}_{\mathbf{r}_i}^\intercal \mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}_{\mathbf{S},i}^\intercal, \\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal(\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_4. This hybrid is same as Hyb_3, except we sample $\mathbf{A}_{\mathbf{t}} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$, $\mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}$.

We have $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$, via the LWE assumption. We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible advantage, then there is a reduction $\mathcal{B}$ that breaks LWE security with non-negligible advantage. The reduction is as follows.

1. The LWE challenger sends $\mathbf{A}_{\mathsf{lwe}} \in \mathbb{Z}_q^{n\times(\mathrm{L_T}+2)m}, \mathbf{b} \in \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ to $\mathcal{B}$.
2. $\mathcal{B}$ parses $\mathbf{A}_{\mathsf{lwe}} = \begin{pmatrix} \bar{\mathbf{A}}'_{\mathsf{fhe}} & \mathbf{A}'_{\mathsf{path}} \end{pmatrix}$, where $\bar{\mathbf{A}}'_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}, \mathbf{A}'_{\mathsf{path}} \in \mathbb{Z}_q^{n\times(\mathrm{L_T}+1)m}$ and $\mathbf{b}^\intercal = [(\bar{\mathbf{b}}')^\intercal \quad (\mathbf{b}'_1)^\intercal]$, where $\bar{\mathbf{b}}' \in \mathbb{Z}_q^m, \mathbf{b}'_1 \in \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$ and does the following.
   - Sets $\mathbf{A}_{\mathbf{t}} = \begin{pmatrix} \bar{\mathbf{A}}'_{\mathsf{fhe}} \\ (\bar{\mathbf{b}}')^\intercal \end{pmatrix}$.
   - Computes $\mathbf{T} = \mathbf{A}_{\mathbf{t}}\mathbf{R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}$.
   - Sets $\bar{\mathbf{A}}_{\mathsf{path}} = \mathbf{A}'_{\mathsf{path}} + (1, \mathsf{bits}(\mathbf{T})) \otimes \bar{\mathbf{G}}$ and $\mathbf{A}_{\mathsf{path}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{path}} \\ \underline{\mathbf{a}}_{\mathsf{path}}^\intercal \end{pmatrix}$, where $\underline{\mathbf{a}}_{\mathsf{path}} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$.
   - Sends $\mathbf{T}$, $\mathbf{D} = (\mathbf{b}'_1)^\intercal - \left(\underline{\mathbf{a}}_{\mathsf{path}}^\intercal - (1, \mathsf{bits}(\mathbf{T})) \otimes \iota_{n+1}^\intercal \otimes \mathbf{g}\right)$ to $\mathcal{A}$.
3. The adversary outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the LWE challenger.

We note that if the LWE challenger sent $\mathbf{b} = \mathbf{t}\mathbf{A}_{\mathsf{lwe}} + \mathbf{e}_{\mathsf{lwe}}$, then $\mathcal{B}$ simulated Hyb_3 with $\mathcal{A}$ else if LWE challenger sent random $\mathbf{b} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ then $\mathcal{B}$ simulated Hyb_4 with $\mathcal{A}$.

To see the latter case, we note that if $\mathbf{b} \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+2)m}$ then it implies $\bar{\mathbf{b}}' \leftarrow \mathbb{Z}_q^m$ and $\mathbf{b}'_1 \leftarrow \mathbb{Z}_q^{(\mathrm{L_T}+1)m}$. Uniformity of $\bar{\mathbf{b}}' \leftarrow \mathbb{Z}_q^m$ implies $\mathbf{A_t} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$. Randomness of $\mathbf{b}'_1$ implies randomness of $\mathbf{D} = (\mathbf{b}'_1)^\intercal - \left(\underline{\mathbf{a}}_{\mathsf{path}}^\intercal - (1, \mathsf{bits}(\mathbf{T})) \otimes \iota_{n+1}^\intercal \otimes \mathbf{g}\right)$.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c_B} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} = \mathbf{A_t R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G},\quad \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}_{0,i}^\intercal)' = \mathbf{s}_{\mathbf{r}_i}^\intercal \mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}_{\mathbf{S},i}^\intercal, \\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_5. This hybrid is same as Hyb_4, except we sample $\mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathrm{T}}$. We have $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$ using leftover hash lemma. By leftover hash lemma (Theorem 2.4) we have that the statistical distance between $\mathbf{A_t R}$ and a uniform matrix $U \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is negligible. This implies that the statistical distance between $\mathbf{A_t R} - \mathsf{bits}(\mathbf{s}, \mathsf{sd}) \otimes \mathbf{G}$ and $\mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathrm{T}}$ is negligible. Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c_B} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathrm{T}},\quad \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}_{0,i}^\intercal)' = \mathbf{s}_{\mathbf{r}_i}^\intercal \mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}); \mathsf{PRF}_2(\mathsf{sd}, \mathbf{r}_i)) + \mathbf{e}_{\mathbf{S},i}^\intercal, \\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathsf{PRF}_3(\mathsf{sd}, \mathbf{r}_i) + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

Hyb_6: In this hybrid, we change all the PRF values computed using sd to random.

We claim that an adversary who can distinguish between Hyb_5 and Hyb_6 can be

used to break PRF security[8]. Intuitively, this is because, the PRF seed sd is now no longer used in the distribution.

In more detail, the reduction queries the $\mathsf{PRF} = (\mathsf{PRF}_1, \mathsf{PRF}_2, \mathsf{PRF}_3)$ challenger with inputs $\mathbf{r}_i$ for $i \in [Q]$ and obtains real or random outputs $\mathbf{y}_{1,i}^\intercal \in [-\sigma', \sigma']^m$, $\mathbf{y}_{2,i} \in \{0,1\}^{m^2 n \lceil \log q \rceil}$, $\mathbf{y}_{3,i}^\intercal \in [-\sigma', \sigma']^{(\mathrm{L_S}+1)m}$. It embeds these into the construction of $\underline{\mathbf{a}}_{\mathbf{r}_i}, \mathbf{S}_{\mathbf{r}_i}, \mathbf{E}_{\mathbf{r}_i}$ as follows:

- Computes $\underline{\mathbf{a}}_{\mathbf{r}_i}^\intercal = \bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{y}_{1,i}$.
- Computes $\mathbf{S}_{\mathbf{r}_i} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \underline{\mathbf{a}}_{\mathbf{r}}^\intercal \end{pmatrix} \mathbf{y}_{2,i} - \mathsf{bits}(\mathbf{s}_{\mathbf{r}}) \otimes \mathbf{G}$, and $\bar{\mathbf{S}}_{\mathbf{r}_i} = \mathsf{Rnd}(\mathbf{S}_{\mathbf{r}_i})$.
- Computes $\mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\bar{\mathbf{S}}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{y}_{3,i} + \mathbf{e}_{\mathbf{E},i}^\intercal$.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_{\mathrm{T}}},\ \ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}_{0,i}^\intercal)' = \mathbf{s}_{\mathbf{r}_i}^\intercal \mathbf{G}\mathbf{u} + \mathbf{e}_{0,i},\ \mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal\}_{i\in[Q]} \\ \{\mathbf{S}_{\mathbf{r}_i} = \mathsf{hct}_{\mathbf{s}_{\mathbf{r}_i}}(\mathsf{bits}(\mathbf{s}_{\mathbf{r}_i})) + \mathbf{e}_{\mathbf{S},i}^\intercal,\ \mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},i}^\intercal + \mathbf{e}_{\mathbf{E},i}^\intercal,\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

where $\mathbf{e}_{\mathsf{circ},i} \leftarrow [-\sigma'\ ,\ \sigma']^{(\mathrm{L_S}+1)m}$.

Hyb_7: In this hybrid, we apply circular tensor LWE (Assumption 4.13). We claim $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$.

To see this, first we observe the following using $\mathbf{s}_{\mathbf{r}_i}^\intercal = (\bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal, -1)$

- $\mathbf{E}_{\mathbf{r}_i} = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},i}^\intercal + \mathbf{e}_{\mathbf{E},i}^\intercal$
  $= \left(\bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal (\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^\intercal\right) - \left(\mathbf{a}_{\mathsf{circ}}^\intercal - (1, \mathsf{bits}(\mathbf{S}_{\mathbf{r}_i})) \otimes \iota_{n+1}^\intercal \otimes \mathbf{g}\right) + \mathbf{e}_{\mathbf{E},i}^\intercal$.
- $\mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{s}_{\mathbf{r}_i}^\intercal (\mathbf{A}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},i}^\intercal$
  $= \left(\bar{\mathbf{s}}_{\mathbf{r}_i}^\intercal (\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^\intercal) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^\intercal\right) - \left(\mathbf{a}_{\mathsf{att}}^\intercal - (1, \mathbf{x}^\intercal) \otimes \iota_{n+1}^\intercal \otimes \mathbf{g}\right)$.

[8] We assume that a single PRF challenger has both PRF_1 and PRF_2.

Next we note that

- We can change $\mathbf{S_r}$ to random if $\mathsf{hct}(\mathbf{A}_{\mathbf{s}_{\mathbf{r}_i}}, \mathsf{bits}(\mathbf{s}_{\mathbf{r}_i}))$ is indistinguishable from random.
- We can change $\mathbf{E_r}$ to random if $\bar{\mathbf{s}}_{\mathbf{r}_i}^\mathsf{T}(\bar{\mathbf{A}}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S}_i)) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{circ},i}^\mathsf{T}$ and $\mathbf{S}_{\mathbf{r}_i}$ are indistinguishable from random random.
- We can change $\mathbf{c}_{\mathsf{att},i}^\mathsf{T}$ to random if $\bar{\mathbf{s}}_{\mathbf{r}_i}^\mathsf{T}(\bar{\mathbf{A}}_{\mathsf{att}} - (1, \mathbf{x}_i^\mathsf{T}) \otimes \bar{\mathbf{G}}) + \mathbf{e}_{\mathsf{att},i}^\mathsf{T}$ is indistinguishable from random.

Invoking circular tensor LWE assumption with respect to secret $\mathbf{s}_{\mathbf{r}_i}, \mathbf{r}_i$, we achieve randomness of the latter terms in the above.

Thus it suffices to show pseudorandomness of the following distribution, given aux_0,

$$\left(\begin{array}{c} \mathsf{mpk} = (\bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}_0, \mathbf{A}_{\mathsf{path}}, \mathbf{A}_{\mathsf{att}}, \mathbf{A}_{\mathsf{circ}}, \mathbf{B}, \mathbf{u}),\ \mathbf{c_B} \leftarrow \mathbb{Z}_q^{(n+1)(m+2)w},\ \mathbf{c}_{\mathsf{msg}} \leftarrow \mathbb{Z}_q^m, \\ \mathbf{T} \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathrm{T}},\ \mathbf{D} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_T}+1)m}, \\ \{(\mathbf{c}_{0,i}^\mathsf{T})' \leftarrow \mathbb{Z}_q,\ \mathbf{c}_{\mathsf{att},i}^\mathsf{T} \leftarrow \mathbb{Z}_q^{(\mathrm{L}+1)m},\ \mathbf{S}_{\mathbf{r}_i} \leftarrow \mathbb{Z}_q^{(n+1)\times m_\mathrm{S}},\ \mathbf{E}_{\mathbf{r}_i} \leftarrow \mathbb{Z}_q^{1\times(\mathrm{L_S}+1)m},\ \mathbf{r}_i\}_{i\in[Q]} \end{array}\right)$$

which completes the proof.

■

Theorem 4.15. *Under the LWE assumption, evasive LWE assumption (Assumption 2.6) for the sampler class $\mathcal{SC}$ induced by the sampler as defined in Equation (4.7), and circular tensor assumption (Assumption 4.13), there exists a very selectively secure CP-ABE scheme supporting unbounded depth circuits $\{C : \{0,1\}^L \to \{0,1\}\}$ and one bit message with efficiency* $|\mathsf{mpk}| = \mathrm{poly}(\lambda, L)$, $|\mathsf{sk}| = \mathrm{poly}(\lambda, L)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda)$.

4.5 GENERIC COMPILER: ABE FOR TURING MACHINES AND NL

For conciseness, we first provide a generic construction of ABE that combines many instance of ABE together in Section 4.5.1. This compiler is adapted from [16], who gave it for FE in the bounded key setting. We then construct ABE for TM in Section 4.5.2 and ABE for NL in Section 4.5.3 by instantiating the generic construction.

4.5.1 Generalized Bundling of Functionality

Consider an ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for a parameter $\mathsf{prm} = 1^i$, a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\perp\}$ for all $i \in \mathbb{N}$ and a message space $\mathcal{M}$. Using such ABE, we will construct a new ABE with ciphertext attribute space A, key attribute space B and message space $\mathcal{M}$. We assume that there exist efficiently computable maps $\mathcal{S} : \mathbb{N} \to 2^{\mathbb{N}}$ and $\mathcal{T} : \mathbb{N} \to 2^{\mathbb{N}}$ such that $\mathsf{max}\mathcal{S}(n)$ and $\mathsf{max}\mathcal{T}(n)$ can be bounded by some fixed polynomial in n. We also assume that there exist maps f with domain A and g with domain B such that

$$f(x) \in \prod_{i \in \mathcal{S}(|x|)} \mathcal{X}_i \quad \text{and} \quad g(y) \in \prod_{i \in \mathcal{T}(|y|)} \mathcal{Y}_i,$$

where $|x|$ and $|y|$ are the lengths of x and y as binary strings. Namely, f and g are maps such that

$$f : \mathrm{A} \ni x \mapsto \{f(x)_i \in \mathcal{X}_i\}_{i \in \mathcal{S}(|x|)}, \quad g : \mathrm{B} \ni y \mapsto \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}.$$

Here, we require that the length of $|f(x)|_i$ and $|g(x)|_i$ can be computed from the length of $|x|$ alone and they do not depend on the actual value of x. In this setting, we can construct an ABE scheme $\mathsf{Bd\text{-}ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for a two input function $R^{\mathsf{bndl}} : \mathrm{A} \times \mathrm{B} \to \{0,1\}^*$ defined in the following

$$R^{\mathsf{bndl}}(x,y) = \begin{cases} 0, & \text{if } R_i(f(x)_i, g(y)_i) = 0 \text{ for all } i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|), \\ 1, & \text{if } R_i(f(x)_i, g(y)_i) = 1 \text{ for all } i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|). \end{cases} \tag{4.9}$$

where $f(x)_i \in \mathcal{X}_i$ and $g(y)_i \in \mathcal{Y}_i$ are the i-th entries of $f(x)$ and $g(x)$, respectively.

Ingredients. We now describe the underlying building blocks used to obtain our ABE construction:

1. An ABE scheme $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ for a parameter $\mathsf{prm} = 1^i$, a relation $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\perp\}$ for $i \in \mathbb{N}$ and a message space $\mathcal{M}$.

2. A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Garble}, \mathsf{GC.Eval})$. We assume that a label is represented by a binary string and denote its length by $L(\lambda, |C|)$, where C is the circuit being garbled. We can instantiate it by Yao's garbled circuit [173], which can be based on any one-way function.

3. An IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.Enc}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Dec})$ with $\mathsf{IND\text{-}CPA}$ security whose identity space and message space are $\{0,1\}^*$. We assume that the key generation algorithm is deterministic. This is without loss of generality, since we can use PRF to derandomize the key generation algorithm. We can instantiate IBE from various standard assumptions including LWE [6, 71], CDH, and Factoring [83].

Construction. Here we provide the description of the construction of $\mathsf{Bd\text{-}ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for R^{bndl} above.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$. On input the security parameter λ, do the following.

1. Run $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$.
2. Output the master key pair as $(\mathsf{mpk}, \mathsf{msk}) \coloneqq (\mathsf{IBE.mpk}, \mathsf{IBE.msk})$.

$\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}_y$. On input master secret key $\mathsf{msk} = \mathsf{IBE.msk}$, a key attribute $y \in \mathrm{B}$, do the following.

1. Compute $\mathcal{T}(|y|) \subseteq \mathbb{N}$, where $|y|$ is the length of y as a binary string.
2. Run $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
3. Compute $g(y) = \{g(y)_i \in \mathcal{Y}_i\}_{i \in \mathcal{T}(|y|)}$.
4. For $i \in \mathcal{T}(|y|)$, compute
$$\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i).$$
5. Let $\ell_i \coloneqq |\mathsf{ABE.mpk}_i|$. For all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$, generate a secret key as
$$\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$$
where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i \in \{0,1\}$ as a binary string.

6. Output

$$\mathsf{sk}_y = \left(\mathcal{T}(|y|),\ \left\{\mathsf{ABE.sk}_i, g(y)_i, \left\{\mathsf{IBE.sk}_{i,j}\right\}_{j\in[\ell_i]}\right\}_{i\in\mathcal{T}(|y|)}\right). \quad (4.10)$$

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}_x$. On input the encryption key $\mathsf{mpk} = \mathsf{IBE.mpk}$, a ciphertext attribute $x \in \mathrm{A}$, and a message $\mu \in \mathcal{M}$, do the following.

1. Compute $\mathcal{S}(|x|) \subset \mathbb{N}$, where $|x|$ is the length of x as a binary string.
2. Compute $f(x) = \{f(x)_i\}_{i\in\mathcal{S}(|x|)}$.
3. Do the following for all $i \in \mathcal{S}(|x|)$.
   a) Compute the length ℓ_i of $\mathsf{ABE.mpk}_i$. This is done without knowing $\mathsf{ABE.mpk}_i$.
   b) Sample a randomness r_i for the encryption algorithm $\mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu; r_i)$. This is done without knowing $\mathsf{ABE.mpk}_i$.
   c) Define a circuit
   $$\mathsf{E}_i(\cdot) \coloneqq \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu; r_i)$$
   that takes as input a string $\mathsf{str} \in \{0,1\}^{\ell_i}$ and outputs $\mathsf{ABE.Enc}(\mathsf{str}, f(x)_i, \mu_i; r_i)$, where str is interpreted as a master public key of the ABE.
   d) Generate a garbled circuit
   $$\left\{\mathsf{lab}_{i,j,b}\right\}_{j\in[\ell_i],b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i).$$
   e) For all $j \in [\ell_i]$ and $b \in \{0,1\}$, compute
   $$\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,b}\right).$$
4. Output

$$\mathsf{ct}_x = \left(\mathcal{S}(|x|), \{f(x)_i\}_{i\in\mathcal{S}(|x|)},\ \left\{\mathsf{IBE.ct}_{i,j,b}\right\}_{i\in\mathcal{S}(|x|),j\in[\ell_i],b\in\{0,1\}}\right). \quad (4.11)$$

$\mathsf{Dec}(\mathsf{sk}_y, y, \mathsf{ct}_x, x) \to \mu' \setminus \perp$. On input a secret key sk_y, key attribute y, a ciphertext ct_x, and ciphertext attribute x, do the following.

1. Parse the secret key sk_y as Eq. (4.10) and the ciphertext ct_x as Eq. (4.11).
2. For all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ do the following.
   a) Retrieve $\mathsf{ABE.mpk}_i$ from the $\mathsf{ABE.sk}_i$.
   b) For all $j \in [\ell_i]$ compute $\mathsf{lab}'_{i,j} \coloneqq \mathsf{IBE.Dec}(\mathsf{IBE.sk}_{i,j,\mathsf{ABE.mpk}_{i,j}}, \mathsf{IBE.ct}_{i,j,\mathsf{ABE.mpk}_{i,j}})$.
   c) Compute $c_i \coloneqq \mathsf{GC.Eval}(\{\mathsf{lab}'_{i,j}\}_{j\in[\ell_i]})$.
   d) Compute $z_i \coloneqq \mathsf{ABE.Dec}(\mathsf{ABE.sk}_i, g(y)_i, c_i, f(x)_i)$.
3. Output $\{z_i\}_{i\in\mathcal{S}(|x|)\cap\mathcal{T}(|y|)}$.

Correctness. We now show that the above construction is correct via the following theorem.

Theorem 4.16. *Suppose* ABE, GC *and* IBE *schemes are correct. Then the* Bd-ABE *scheme is correct.*

Proof. We observe that $\mathsf{lab}'_{i,j} = \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ holds for all $\mathsf{lab}'_{i,j}$ recovered in Step 2b of the decryption algorithm by the correctness of IBE, since the ciphertext and secret key are both generated with respect to the identity $(i, j, \mathsf{ABE.mpk}_{i,j})$. Then, by the correctness of GC, we have $c_i = \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu; r_i)$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ recovered in Step 2c of the decryption algorithm. Finally, if $R^{\mathsf{bndl}}(x, y) = 0$, then by the definition of R^{bndl}, we have $R_i(f(x)_i, g(y)_i) = 0$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$ and thus by the correctness of ABE, we get $z_i = \mu$ for all z_i recovered in Step 2d of the decryption algorithm as desired. ∎

Security. We prove the security of the Bd-ABE scheme via the following theorem.

Theorem 4.17. *Suppose* GC *is a secure garbled circuit scheme and* IBE *is* IND-CPA *secure identity-based encryption scheme. Then, assuming* ABE *is* Sel-IND *secure (Definition 2.5), so is* Bd-ABE. *Furthermore, if* ABE *is* VerSel-IND *secure*

(Definition 2.7), so is Bd-ABE[9].

Proof. Here we prove the statement for Sel-IND security. The proof proceeds via a sequence of hybrid games between the challenger and a PPT adversary $\mathcal{A}$.

Hyb_0. This is the real world with $\beta = 0$, i.e., the challenge ciphertext is computed using the message μ_0. We write the complete game here to set up the notations and easy reference in later hybrids.

1. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.
2. The challenger generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and sends it to $\mathcal{A}$.
3. **Key Queries:** The adversary can make key queries, before and after challenge query, in an arbitrary order. For each key query $y \in \mathrm{B}$, the challenger does the following.
   - It computes the mapping $\mathcal{T}(|y|)$ and generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
   - It computes the mapping $g(y)$ and generates $\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i)$ for all $i \in \mathcal{T}(|y|)$.
   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and computes $\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$.
   - It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i\in\mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|), j\in\ell_i})$.
4. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages $(\mu_0, \mu_1) \in \mathcal{M}^2$. The challenger does the following.
   - It computes $\mathcal{S}(|x|)$ and $f(x) = \{f(x)_i\}_{i\in\mathcal{S}(|x|)}$.
   - For all $i \in \mathcal{S}(|x|)$,
     - It samples the randomness r_i and defines the circuit $\mathsf{E}_i(\cdot) := \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_0; r_i)$ as in the construction.

[9] In the latter case, the adversary can see the public key before declaring all the key queries.

– It generates $\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell_i],b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$.

– It computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,b})$.

- It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|),j\in[\ell_i],b\in\{0,1\}})$.

5. $\mathcal{A}$ outputs a guess bit β'.

Hyb_1. In this hybrid, we change the way challenge query is answered. In particular, the challenger computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}})$, for all $i \in \mathcal{S}(|x|)$, $j \in [\ell_i]$, and $b \in \{0, 1\}$, where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i$.

Note that we encrypt the same label for both $b = 0$ and $b = 1$.

Hyb_2. In this hybrid, we further change the way challenge query is answered. In particular, we change the way $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is generated. The challenger does the following for all $i \in \mathcal{S}(|x|)$,

- It computes $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_0)$.
- It generates $\{\mathsf{lab}_{i,j}\}_{j\in[\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, \mathsf{ABE.ct}_i)$.
- It sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \{\mathsf{lab}_{i,j}\}$ for all $j \in [\ell_i]$.

Hyb_3. In this hybrid, we further change the way challenge query is answered. In particular, the challenger computes $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_1)$ for all $i \in \mathcal{S}(|x|)$.

The following hybrids are unwinding of the preceding hybrids.

Hyb_4. In this hybrid, we change the way challenge query is answered. In particular, we change the way $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is generated. The challenger does the following for all $i \in \mathcal{S}(|x|)$.

- It samples an encryption randomness r_i and define $\mathsf{E}_i(\cdot) = \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_1; r_i)$.
- It generates $\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell_i],b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$.
- It sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \{\mathsf{lab}_{i,j,b}\}$ for all $j \in [\ell_i]$, where $b = \mathsf{ABE.mpk}_{i,j}$.

Hyb_5. In this hybrid, we further change the way challenge query is answered. In particular, the challenger computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i, j, b), \mathsf{lab}_{i,j,b})$, for all $i \in \mathcal{S}(|x|)$, $j \in [\ell_i]$, and $b \in \{0, 1\}$.

This is the real world with $\beta = 1$.

Indistinguishability of hybrids We now show that the consecutive hybrids are indistinguishable.

Claim 4.18. Assume that IBE is IND-CPA secure. Then $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the IND-CPA security of IBE scheme with advantage ϵ. The reduction is as follows.

1. The IBE challenger generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ and samples a bit $\hat{\beta} \leftarrow \{0, 1\}$. It sends IBE.mpk to $\mathcal{B}$.
2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in$ A. $\mathcal{B}$ sets mpk = IBE.mpk and forwards it to $\mathcal{A}$.
3. **Key Queries:** For a key query $y \in$ B, $\mathcal{B}$ does the following.
   - It computes the mapping $\mathcal{T}(|y|)$ and generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{T}(|y|)$.
   - It computes the mapping $g(y)$ and generates $\mathsf{ABE.sk}_i \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.msk}_i, g(y)_i)$ for all $i \in \mathcal{T}(|y|)$.
   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and sends secret key query for identity $(i, j, \mathsf{ABE.mpk}_{i,j})$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$ to the IBE challenger. The

challenger returns $\{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|),j\in\ell_i}$.

- It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i\in\mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|),j\in\ell_i})$ to $\mathcal{A}$.

4. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following.

   - It sets $\mu = \mu_0$ and computes $\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell_i],b\in\{0,1\}}$, for all $i \in \mathcal{S}(|x|)$, as in the honest encryption algorithm.

   - It computes $\{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|),j\in[\ell_i],b\in\{0,1\}}$ as follows.

     – If $b = \mathsf{ABE.mpk}_{i,j}$, it honestly encrypts $\mathsf{lab}_{i,j,b}$ to obtain $\mathsf{IBE.ct}_{i,j,b}$.

     – If $b = 1 - \mathsf{ABE.mpk}_{i,j}$, $\mathcal{B}$ submits identity (i, j, b) and messages $(\mathsf{lab}_{i,j,b}, \mathsf{lab}_{i,j,1-b})$ to the IBE challenger. The challenger encrypts

     $$\mathsf{IBE.ct}_{i,j,b} \leftarrow \begin{cases} \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,b}\right), & \text{if } \hat{\beta} = 0 \\ \mathsf{IBE.Enc}\left(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,1-b}\right) & \text{if } \hat{\beta} = 1 \end{cases}$$

     and returns $\mathsf{IBE.ct}_{i,j,b}$ to $\mathcal{B}$.
     Note that the same label $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ is encrypted for identities $(i, j, 0)$ and $(i, j, 1)$ if $\beta = 1$.

   - It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|),j\in[\ell_i],b\in\{0,1\}})$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the IBE challenger.

We observe that if the IBE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated Hyb_0, else Hyb_1 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)| = |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_0) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1)| = \epsilon$ (by assumption).

Admissibility of $\mathcal{B}$. Observe that $\mathcal{B}$ submits identities of the form $(i, j, \mathsf{ABE.mpk}_{i,j})$ for secret key queries and identities of the form $(i, j, 1 - \mathsf{ABE.mpk}_{i,j})$ for encryption queries. So, $\mathcal{B}$ does not make a secret key query for an identity that is also submitted to the IBE challenger for an encryption query. This establishes the admissibility of $\mathcal{B}$. ■

Claim 4.19. Assume that GC is secure. Then $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of GC scheme with advantage ϵ. The reduction is as follows.

1. The GC challenger samples a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$.
2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.
3. $\mathcal{B}$ generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and forwards it to $\mathcal{A}$.
4. **Key Queries:** For a key query $y \in \mathrm{B}$, $\mathcal{B}$ computes $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i\in\mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i\in\mathcal{T}(|y|), j\in\ell_i})$ as in Hyb_0.
5. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following for all $i \in \mathcal{S}(|x|)$.
   - It samples r_i and defines a circuit $\mathsf{E}_i(\cdot) \coloneqq \mathsf{ABE.Enc}(\cdot, f(x)_i, \mu_0; r_i)$.
   - It submits the circuit $\mathsf{E}_i(\cdot)$ and $\mathsf{ABE.mpk}_i$ to the GC challenger. The challenger does the following.
     - If $\hat{\beta} = 0$, it computes $\{\mathsf{lab}'_{i,j,b}\}_{j\in[\ell_i], b\in\{0,1\}} \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{E}_i)$ and sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \mathsf{lab}'_{i,j,\mathsf{ABE.mpk}_{i,j}}$, where $\mathsf{ABE.mpk}_{i,j}$ is the j-th bit of $\mathsf{ABE.mpk}_i$.
     - If $\hat{\beta} = 1$, it computes $\{\mathsf{lab}'_{i,j}\}_{j\in[\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, E_i(\mathsf{ABE.mpk}_i))$ and sets $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}} = \mathsf{lab}'_{i,j}$.
     - It returns $\mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}}$ to $\mathcal{B}$.
   - It computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j,\mathsf{ABE.mpk}_{i,j}})$, for all $i \in \mathcal{S}(|x|)$, $j \in [\ell_i]$, and $b \in \{0,1\}$.
   - It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|), j\in[\ell_i], b\in\{0,1\}})$ to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the GC challenger.

We observe that if the GC challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the distribution Hyb_1 , else Hyb_2 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)|$ $= |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_2)| = \epsilon$ (by assumption). ∎

Claim 4.20. Assume that ABE is $\mathsf{Sel\text{-}IND}$ secure. Then $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. We show that if $\mathcal{A}$ can distinguish between Hyb_2 and Hyb_3 with non-negligible advantage ϵ, then there exists a PPT adversary $\mathcal{B}$ against the security of ABE scheme with advantage ϵ. In this game, the ABE challenger is the challenger corresponding to all the i-th instance of ABE such that $i \in \mathcal{S}(|x|)$. For $i \notin \mathcal{S}(|x|)$, the reduction $\mathcal{B}$ itself generates such i-the instance ABE. The reduction is as follows.

1. The ABE challenger samples a bit $\hat{\beta} \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$.

2. $\mathcal{B}$ invokes $\mathcal{A}$. $\mathcal{A}$ outputs the challenge ciphertext attribute $x \in \mathrm{A}$.

3. $\mathcal{B}$ generates $(\mathsf{IBE.mpk}, \mathsf{IBE.msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, sets $\mathsf{mpk} = \mathsf{IBE.mpk}$ and forwards it to $\mathcal{A}$. $\mathcal{B}$ also computes $\mathcal{S}(|x|)$ and $f(x) = \{f(x)_i\}_{i \in \mathcal{S}(|x|)}$.

4. $\mathcal{B}$ sends the challenge ciphertext attributes $f(x)_i$ for the i-th instance of ABE, where $i \in \mathcal{S}(|x|)$, to the ABE challenger. The ABE challenger generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$ for $i \in \mathcal{S}(|x|)$ and returns $\{\mathsf{ABE.mpk}_i\}_{i \in \mathcal{S}(|x|)}$ to $\mathcal{B}$.

5. **Key Queries:** For a key query $y \in \mathrm{B}$, $\mathcal{B}$ does the following.

   - It computes the mapping $\mathcal{T}(|y|)$ and for all $i \in \mathcal{T}(|y|) \setminus \mathcal{S}(|x|)$, it generates $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i)$.

   - It computes the mapping $g(y) = \{g(y)_i\}_{i \in \mathcal{T}(|y|)}$. For all $i \in \mathcal{T}(|y|) \cap \mathcal{S}(|x|)$, it sends a key query $g(y)_i$ for the i-th instance of ABE to its challenger and gets back $\mathsf{ABE.sk}_i$. For $i \in \mathcal{T}(|y|) \setminus \mathcal{S}(|x|)$, $\mathcal{B}$ computes $\mathsf{ABE.sk}_i$ itself.

   - It sets $\ell_i := |\mathsf{ABE.mpk}_i|$ and computes $\mathsf{IBE.sk}_{i,j} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}, (i, j, \mathsf{ABE.mpk}_{i,j}))$ for all $i \in \mathcal{T}(|y|)$ and $j \in \ell_i$.

   - It returns $\mathsf{sk}_y = (\mathcal{T}(|y|), g(y), \{\mathsf{ABE.sk}_i\}_{i \in \mathcal{T}(|y|)}, \{\mathsf{IBE.sk}_{i,j}\}_{i \in \mathcal{T}(|y|), j \in \ell_i})$. to $\mathcal{A}$.

6. **Challenge Query:** $\mathcal{A}$ outputs a pair of equal length messages (μ_0, μ_1). $\mathcal{B}$ does the following.

   - For all $i \in \mathcal{S}(|x|)$,

     – It sends challenge ciphertext query μ_0, μ_1 for the i-th instance of ABE to the challenger. The challenger computes and returns $\mathsf{ABE.ct}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, f(x)_i, \mu_{\hat{\beta}})$.

     – It generates $\{\mathsf{lab}_{i,j}\}_{j \in [\ell_i]} \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell_i}, 1^{|E_i(\cdot)|}, \mathsf{ABE.ct}_i)$ and

computes $\mathsf{IBE.ct}_{i,j,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{IBE.mpk}, (i,j,b), \mathsf{lab}_{i,j})$, for all $j \in [\ell_i]$, and $b \in \{0,1\}$.

- It returns $\mathsf{ct}_x = (\mathcal{S}(|x|), f(x), \{\mathsf{IBE.ct}_{i,j,b}\}_{i\in\mathcal{S}(|x|), j\in[\ell_i], b\in\{0,1\}})$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ forwards β' to the ABE challenger.

We observe that if the ABE challenger samples $\hat{\beta} = 0$, then $\mathcal{B}$ simulated the distribution Hyb_2 , else Hyb_3 with $\mathcal{A}$. Hence, advantage of $\mathcal{B} = |\mathsf{Pr}(\beta' = 1|\hat{\beta} = 0) - \mathsf{Pr}(\beta' = 1|\hat{\beta} = 1)|$ $= |\mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_1) - \mathsf{Pr}(\beta' = 1|\mathsf{Hyb}_2)| = \epsilon$.

Admissibility of $\mathcal{B}$. Observe that $\mathcal{B}$ issues ciphertext queries for the challenge attribute of type $\{f(x)_i\}_{i\in\mathcal{S}(|x|)}$ and key queries of the form $\{g(y)_i\}_{i\in\mathcal{T}(|y|)\cap\mathcal{S}(|x|)}$. Also, by the admissibility of $\mathcal{A}$, it can only issue key queries $y \in \mathrm{B}$ such that $R^{\mathsf{bndl}}(x,y) = 1$ for the challenge ciphertext attribute $x \in \mathrm{A}$. Then by the definition of R^{bndl}, we have $R_i(f(x)_i, g(y)_i) = 1$ for all $i \in \mathcal{S}(|x|) \cap \mathcal{T}(|y|)$. Thus all the key queries made by $\mathcal{B}$ satisfies $R_i(f(x)_i, g(y)_i) = 1$. This establishes the admissibility of $\mathcal{B}$. ■

The rest of the hybrids, Hyb_4 and Hyb_5, are simply unwinding the previous hybrids and their proofs of indistinguishability are same as their corresponding counterparts in the first set of hybrids and hence, omitted. ■

4.5.2 ABE for Turing Machines

Here, we provide the construction of ABE for Turing machines. Namely, a ciphertext is associated with $(x, 1^t)$ and a secret key is for a Turing machine M, and the decryption results to 1 if the machine accepts the input within t steps and 0 otherwise. To construct such a scheme, we start with constructing two schemes with partial functionality and then combine them. The one scheme takes care of the case where $|(x, 1^t)| > |M|$, while the other takes care of the case where $|(x, 1^t)| \leq |M|$. The idea for the construction is very similar to FE for TM in [16].

Circuit $U_{i,x,t}$

Hardwired constants: x, t.
On input $M = (Q, \delta, F) \in \{0,1\}^i$, proceed as follows:

1. Parse the input $M \in \{0,1\}^i$ as a description of a Turing machine.
2. Run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Figure 4.2: Circuit $U_{i,x,t}$.

The Case of $|(x, 1^t)| > |M|$

We first show that by applying the conversion in Section 4.5.1 to the CP-ABE scheme for unbounded depth circuits in Section 4.4, we can obtain an ABE scheme for Turing machines for the case where $|(x, 1^t)| > |M|$. Formally, we construct an ABE for $R^{>} : \mathrm{A} \times \mathrm{B} \to \{0,1\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all Turing machines, and

$$R^{>}((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ in } t \text{ steps}) \wedge \left(|(x, 1^t)| > |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of the unbounded depth and size cpABE for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\} \cup \{\bot\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, and output length 1, $\mathcal{Y}_i = \{0,1\}^i$, and $R_i(C, x) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1, 2, \ldots, i-1\}, \quad \mathcal{T}(i) = \{i\}, \quad f(x, 1^t) = \left\{U_{i,x,t}(\cdot)\right\}_{i \in [|(x,1^t)|-1]}, \quad g(M) = M$$

where $U_{i,x,t}(\cdot)$ is defined as Figure 4.2. The circuit (in particular, Step 2 of the computation) is padded so that the circuit size only depends on $|(x, 1^t)|$. Note that $U_{i,x,t}$ is in $\mathcal{X}_i$ even for x and t with unbounded length, since $\mathcal{X}_i$ contains circuits of unbounded size. Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is equivalent to $R^{>}$ except for the case of $|(x, 1^t)| \leq |M|$. In this case, we have $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. To handle this, we add the extra step to the

decryption algorithm of the former where it outputs $\perp$ if $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. Since the original scheme is VerSel-IND secure, so is the resulting scheme by Theorem 4.17.

The Case of $|(x, 1^t)| \leq |M|$

We next show that by applying the conversion in Section 4.5.1 to the KP-ABE scheme from [43], we can obtain an ABE scheme for TM for the case where $|(x, 1^t)| \leq |M|$. Formally, we construct an ABE for $R^{\leq} : \mathrm{A} \times \mathrm{B} \to \{0,1\} \cup \{\perp\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all non-deterministic Turing machines, and

$$R^{\leq}((x, 1^t), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x, 1^t)| \leq |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of kpABE from [43] for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i = \{0,1\}^i$ and $\mathcal{Y}_i$ is the set of circuits with input length i, depth $i \cdot \lambda$, and output length 1, and $R_i(x, C) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1, 2, \ldots, i\}, \quad f(x, 1^t) = (x, 1^t), \quad g(M) = \left\{U_{i,M}(\cdot)\right\}_{i \in [M]},$$

where $U_{i,M}(\cdot)$ is defined as Figure 4.3. Here, we check that $U_{i,M}(\cdot)$ is in $\mathcal{Y}_i$. Recall that even though $\mathcal{Y}_i$ supports circuits with unbounded size, it has a bound on the depth of the circuit. We therefore argue that the depth of $U_{i,M}(\cdot)$ does not exceed $i\lambda$, even for unbounded size $|M|$. We evaluate the depth of Step 2 of the circuit, since this is the only non-trivial step. By Lemma 4.1, this step can be implemented by a circuit with depth

$$t \cdot \mathrm{poly}(\log |x|, \log t, \log |M|) \leq i \cdot \mathrm{poly}(\log \lambda) \leq i \cdot \lambda$$

Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is equivalent to $R^{\leq}$ except for the case of $|(x, 1^t)| > |M|$. In this case, we have $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. To handle this, we add the extra step to the decryption algorithm of the former where it outputs $\perp$ if $\mathcal{S}(|x|) \cap \mathcal{T}(|y|) = \emptyset$. Since the original scheme is Sel-IND secure, so is the resulting scheme by Theorem 4.17.

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M.
On input $y \in \{0,1\}^i$, proceed as follows:

1. Parse the input $y \in \{0,1\}^i$ as $(x, 1^t)$.
2. Otherwise, run M on input x for t steps.
3. Output 1 if the state is in F (accept) and 0 otherwise.

Figure 4.3: Circuit $U_{i,M}$.

Putting the Pieces Together

Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathrm{A} = \{0,1\}^*$ and B to be the set of all Turing machines. We also set $R_1 = R^{>}$, $R_2 = R^{\leq}$. $\mathcal{X}_i = \mathrm{A}$, $\mathcal{Y}_i = \mathrm{B}$ for $i = 1, 2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = \{1,2\}, \quad \mathcal{T}(i) = \{1,2\}, \quad f(x,1^t) = \{(x,1^t),(x,1^t)\}, \quad g(M) = \{M, M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is

$$R^{\mathsf{bndl}}((x,1^t),M) = \begin{cases} (1,0) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x,1^t)| > |M|\right) \\ (0,1) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps}) \wedge \left(|(x,1^t)| \leq |M|\right) \\ (0,0) & \text{otherwise.} \end{cases}.$$

To obtain the ABE scheme for TM, we add an extra step for the decryption algorithm of the ABE scheme for R^{bndl} obtained above, where we output the message recovered if the decryption succeeds for either left or right slot and $\perp$ otherwise. To sum up, we have the following theorem

Theorem 4.21. *Assume a* VerSel-IND *secure* cpABE *scheme that supports circuits with unbounded depth and a* Sel-IND *secure* kpABE *scheme that supports bounded*

depth circuits. Then there exists a ABE for Turing machines, presented in Section 4.5.2, satisfying VerSel-IND *security.*

Instantiating the VerSel-IND secure cpABE scheme from Section 4.4 and Sel-IND secure kpABE scheme from [43], we get the following corollary.

Corollary 4.21.1. *Under the LWE assumption, evasive LWE assumption (Assumption 2.6) for the sampler class $\mathcal{SC}$ induced by the sampler as defined in Equation (4.7), and circular tensor LWE assumption (Assumption 4.13), there exists a very selectively secure ABE for TM with* $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(|M|, \lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |x|, t)$. *where the Turing machine M runs on input x for time step t.*

4.5.3 ABE for NL

Here, we provide the construction of ABE for NL. Namely, a ciphertext encrypts a string $(x, 1^t, 1^{2^s})$, where x is a string, t is the time bound, s is the space bound for the computation and a secret key is associated with a non-deterministic Turing machine M. The decryption is possible if M accepts x within t steps and the space used for the computation does not exceed s. The idea for the construction is very similar to FE for NL in [16]. We consider two schemes that complement each other and then combine them.

Transition Matrix. Before describing the schemes, we need some preparations. Similarly to [143], we represent the computation of $M(x)$ as a multiplication of matrices. To do so, let us enumerate all the possible internal configurations that may appear when we run $M = (Q, \delta, F)$ on input x with the space for the computation being bounded by s. As an internal configuration, we have $|x|$ and s choices for the input and work tape pointers, respectively. We also have $|Q|$ choices for the possible state, and 2^s possible choices for the contents of the work tape. Therefore, we have

$$N := s2^s \cdot |x| \cdot |Q|$$

possible internal configurations. We associate each $i \in [N]$ with such configuration and represent the configuration by a vector $\mathbf{e}_i$, which is a unit vector whose entries are all 0 except for the i-th entry that is set to be 1. We also define the matrix $\mathsf{Mat}(M, x, s)$ as

$$\mathsf{Mat}(M, x, s)_{i,j} \coloneqq \begin{cases} 1 & \text{if the configuration } i \text{ can reach } j \text{ in one step by } \delta \\ 0 & \text{otherwise} \end{cases}$$

where $\mathsf{Mat}(M, x, s)_{i,j}$ is the (i, j)-th entry of the matrix.

Furthermore, we consider a special matrix multiplication over $\{0, 1\}$, where the multiplication $\mathbf{A} \cdot \mathbf{B} \in \{0, 1\}^{N_1 \times N_3}$ of two matrices $\mathbf{A} \in \{0, 1\}^{N_1 \times N_2}$ and $\mathbf{B} \in \{0, 1\}^{N_2 \times N_3}$ is defined as

$$(\mathbf{A} \cdot \mathbf{B})_{i,j} = \bigvee_{k \in [N_2]} \left(\mathbf{B}_{i,k} \wedge \mathbf{C}_{k,j}\right)$$

where we denote the (i, j)-th entry of a matrix $\mathbf{D}$ by $\mathbf{D}_{i,j}$ above. The multiplication can be defined for any size of matrices and in particular, also defined for computations involving vectors. We then define $\mathbf{e}_{\mathsf{stt}}$ as the vector corresponding to the initial state of the computation and $\mathbf{u}_{\mathsf{acc}}$ as

$$\mathbf{u}_{\mathsf{acc}} = \sum_{i \in [N] \,:\, i \text{ encodes accepting state}} \mathbf{e}_i.$$

Then, we observe that

$$\mathbf{e}_{\mathsf{stt}}^{\top} \cdot \left(\mathsf{Mat}(M, x, s)_{i,j}\right)^t \cdot \mathbf{u}_{\mathsf{acc}} = \begin{cases} 1 & \text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s \\ 0 & \text{otherwise} \end{cases}$$

holds from the property of the transition matrix, where we use the special multiplication above.

The Case of $|(x, 1^t, 1^{2^s})| > |M|$

We first show that by applying the conversion in Section 4.5.1 to the CP-ABE scheme for circuits in [169], we can obtain an ABE scheme for NL for the case where $|(x, 1^t, 1^{2^s})| > |M|$. Formally, we construct an ABE for $R^{>} : \mathrm{A} \times \mathrm{B} \to \{0, 1\}$, where $\mathrm{A} = \{0, 1\}^*$, B is

the set of all non-deterministic Turing machines, and

$$R^{>}((x,1^t,1^{2^s}),M)=\begin{cases}1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s)\wedge\left(|(x,1^t,1^{2^s})|>|M|\right)\\ 0 & \text{otherwise.}\end{cases}.$$

To apply the conversion, we use various instances of cpABE from [169] for $\mathsf{prm}=1^i$, $R_i:\mathcal{X}_i\times\mathcal{Y}_i\to\{0,1\}\cup\{\bot\}$ where $\mathcal{X}_i$ is the set of circuits with input length i, depth λ, and output length 1, $\mathcal{Y}_i=\{0,1\}^i$, and $R_i(C,x)=C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i)=\{1,2,\ldots,i-1\},\quad \mathcal{T}(i)=\{i\},\quad f(x,1^t,1^{2^s})=\left\{U_{i,x,t,s}(\cdot)\right\}_{i\in[|(x,1^t,1^{2^s})|-1]},\quad g(M)=M$$

where $U_{i,x,t,s}(\cdot)$ is defined as Figure 4.4. The circuit may be padded so that it does not leak more information about the hardwired constants (x,t,s) beyond $|(x,1^t,1^{2^s})|$.

We now show that f is a valid map. Namely, $U_{i,x,t,s}$ is in $\mathcal{X}_i$ even for x and t with unbounded length. Recall that $\mathcal{X}_i$ has a bound on the depth of the circuits it supports, even though it does not have such a bound on the size. We argue that the depth of $U_{i,x,t,s}$ is bounded by λ. To do so, we evaluate the depth of each computation step of $U_{i,x,t,s}$. First, Step 1 of the computation is trivial. Step 2 can be implemented by a circuit of depth $\mathrm{poly}(\log|x|,\log s,\log|M|)$ by Lemma 4.2. Step 3 can be executed by $O(\log t)$ multiplication of matrices of size $N\times N$, which can be implemented with depth $O(\log N)$. Therefore, this step can be computed with depth $O(\log N\log t)$. Step 4 can also be implemented with depth $O(\log N)$. Therefore, the entire computation can be implemented with depth

$$\mathrm{poly}(\log s,\log|x|,\log t,\log N,\log|M|)\le\mathrm{poly}(\log\lambda)\le\lambda$$

as desired. Then, by inspection, we can observe that for $\mathcal{X}_i,\mathcal{Y}_i,\mathcal{S},\mathcal{T},f,g,\mathrm{A},\mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is equivalent to $R^{>}$ except for the case of $|(x,1^t,1^{2^s})|\le|M|$. In this case, we have $\mathcal{S}(|x|)\cap\mathcal{T}(|y|)=\emptyset$. To handle this, we add the extra step to the decryption algorithm of the former where it outputs $\bot$ if $\mathcal{S}(|x|)\cap\mathcal{T}(|y|)=\emptyset$. This gives us the construction of ABE for $R^{>}$. Since the original

Circuit $U_{i,x,t,s}$

Hardwired constants: x, t, s.
On input $M \in \{0,1\}^i$, proceed as follows:

1. Parse the input $M = (Q, \delta, F)$ as a description of a Turing machine.
2. Compute $\mathsf{Mat}(M, x, s)_{j,k}$ for all $j, k \in [N]$ in parallel, where $N = s2^s \cdot |x| \cdot |Q|$.
3. Compute $\mathbf{A} := \mathsf{Mat}(M, x, s)^t$.
4. Compute and output $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathbf{A} \cdot \mathbf{u}_{\mathsf{acc}}$.

Figure 4.4: Circuit $U_{i,x,t,s}$.

scheme is VerSel-IND secure, so is the resulting scheme by Theorem 4.17.

The Case of $|(x, 1^t, 1^{2^s})| \leq |M|$

We next show that by applying the conversion in Section 4.5.1 to the KP-ABE scheme from [43], we can obtain an ABE scheme for NL for the case where $|(x, 1^t, 1^{2^s})| \leq |M|$. Formally, we construct an ABE for $R^{\leq} : \mathrm{A} \times \mathrm{B} \to \{0,1\} \cup \{\bot\}$, where $\mathrm{A} = \{0,1\}^*$, B is the set of all non-deterministic Turing machines, and

$$R^{\leq}((x, 1^t, 1^{2^s}), M) = \begin{cases} 1 & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x, 1^t, 1^{2^s})| \leq |M|\right) \\ 0 & \text{otherwise.} \end{cases}.$$

To apply the conversion, we use various instances of kpABE from [43] for $\mathsf{prm} = 1^i$, $R_i : \mathcal{X}_i \times \mathcal{Y}_i \to \{0,1\}$ where $\mathcal{X}_i = \{0,1\}^i$ and $\mathcal{Y}_i$ is the set of circuits with input length i, depth λ, and output length 1, and $R_i(x, C) = C(x)$. We then set $\mathcal{S}$, $\mathcal{T}$, f, and g as

$$\mathcal{S}(i) = i, \quad \mathcal{T}(i) = \{1, 2, \ldots, i\}, \quad f(x, 1^t, 1^{2^s}) = (x, 1^t, 1^{2^s}), \quad g(M) = \left\{U_{i,M}(\cdot)\right\}_{i \in [M]},$$

where $U_{i,M}(\cdot)$ is defined as Figure 4.5.

We now show that g is a valid map. Namely, $U_{i,M}$ is in $\mathcal{Y}_i$ even for M with unbounded size. In particular, we have to show that the depth of the circuit is bounded by λ.

Circuit $U_{i,M}$

Hardwired constants: Description of a Turing Machine M.
On input $y \in \{0,1\}^i$, proceed as follows:

1. Parse the input $y \in \{0,1\}^i$ as $(x, 1^t, 1^{2^s})$.
2. Compute $\mathsf{Mat}(M, x, s)_{j,k}$ for all $j, k \in [N]$ in parallel, where $N = s2^s \cdot |x| \cdot |Q|$.
3. Compute $\mathbf{A} := \mathsf{Mat}(M, x, s)^t$.
4. Compute and output $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathbf{A} \cdot \mathbf{u}_{\mathsf{acc}}$.

Figure 4.5: Circuit $U_{i,M}$.

This can be shown by the same argument as for $U_{i,x,t,s}$, since both circuits compute $\mathbf{e}_{\mathsf{stt}}^\top \cdot \mathsf{Mat}(M, x, s)^t \cdot \mathbf{u}_{\mathsf{acc}}$ from (x, t, s, M) in exactly the same way. Then, by inspection, we can observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is equivalent to $R^{\leq}$ except for the case of $|(x, 1^t, 1^{2^s})| > |M|$. However, this case can be handled by modifying the decryption algorithm as in Section 4.5.3. This gives us the construction of ABE for $R^{\leq}$. Since the original scheme is $\mathsf{Sel\text{-}IND}$ secure, so is the resulting scheme by Theorem 4.17.

Putting the Pieces Together

Here, we combine the two schemes we considered so far to obtain the full-fledged scheme. We set $\mathrm{A} = \{0,1\}^*$ and B to be the set of all Turing machines. We also set $R_1 = R^{>}$, $R_2 = R^{\leq}$. $\mathcal{X}_i = \mathrm{A}$, $\mathcal{Y}_i = \mathrm{B}$ for $i = 1, 2$. We have already constructed schemes for R_1 and R_2 and now combine them. To do so, we set $\mathcal{S}, \mathcal{T}, f$, and g as

$$\mathcal{S}(i) = \{1,2\}, \quad \mathcal{T}(i) = \{1,2\}, \quad f(x, 1^t) = \{(x, 1^t, 1^{2^s}), (x, 1^t, 1^{2^s})\}, \quad g(M) = \{M, M\}$$

We observe that for $\mathcal{X}_i, \mathcal{Y}_i, \mathcal{S}, \mathcal{T}, f, g, \mathrm{A}, \mathrm{B}$ defined as above, R^{bndl} defined as Equation (4.9) is

$$R^{\mathsf{bndl}}((x, 1^t, 1^{2^s}), M) = \begin{cases} (1,0) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x, 1^t, 1^{2^s})| > |M|\right) \\ (0,1) & (\text{if } M \text{ accepts } x \text{ within } t \text{ steps and space } s) \wedge \left(|(x, 1^t, 1^{2^s})| \leq |M|\right) \\ (0,0) & \text{otherwise.} \end{cases}.$$

To obtain the ABE scheme for NL, we add an extra step for the decryption algorithm of the ABE scheme for R^{bndl} obtained above, where we output the message recovered if the decryption succeeds for either left or right slot and $\perp$ otherwise. To sum up, we have the following theorem:

Theorem 4.22. *Assume a* VerSel-IND *secure* cpABE *and* Sel-IND *secure* kpABE *scheme that supports circuits with bounded depth. Then the ABE scheme for NL presented in Section 4.5.3 satisfies* VerSel-IND *security.*

Instantiating the VerSel-IND secure cpABE scheme from [169] and Sel-IND secure kpABE scheme from [43], we get the following corollary.

Corollary 4.22.1. *Under the (public-coin) evasive LWE assumption (Assumption 2.6) and tensor LWE assumption (Assumption 4.13), there exists a very selectively secure ABE for NL with* $|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(|M|, \lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |x|, t, 2^s)$ *where the machine* M *runs on input* x *for time step* t *and takes space* s.

CHAPTER 5

COMPACT PSEUDORANDOM FUNCTIONAL ENCRYPTION FROM EVASIVE LWE

5.1 INTRODUCTION

Functional encryption (FE) [157, 46] generalizes public key encryption to support computation of non-trivial functions on encrypted data, beyond "all or nothing" access. More formally, in FE, a ciphertext is associated with a vector $\mathbf{x}$, a secret key is associated with a circuit f and decryption enables recovery of $f(\mathbf{x})$ and nothing else. Aside from its direct relevance to real world applications for computing on encrypted data, FE has proved to be a powerful tool in the theory of cryptography, and can be used to build a number of advanced primitives. The most prominent amongst these is Indistinguishability Obfuscation (iO), which is considered essentially "crypto-complete" by virtue of its ability to instantiate almost every known cryptographic primitive [34, 90, 129].

For FE to succeed in being bootstrapped all the way to iO, it must satisfy a strong efficiency property known as *compactness* – at a high level, this posits that the size of the ciphertext should be sublinear in the size of the circuits being supported by the scheme. There has been substantial research effort in the community for instantiating FE (or directly iO) from well-understood assumptions, leading to a sequence of exciting results [128, 129, 5, 20, 171, 96, 82, 129, 130, 154]. The breakthrough work of Jain, Lin and Sahai [129] finally obtained the first construction of compact FE for P from standard assumptions. This has been subsequently improved by [130, 154]. However, all these works rely quite crucially on pairings which is dissatisfying. Not only are pairings quantum insecure, it is also desirable to have alternate pathways for constructing such important primitives as FE and iO, even for restricted classes of functions. In the realm of conjectured quantum safety, there exist several candidates from lattices but their

security is either based on heuristics, or their underlying assumptions have been broken [5, 20, 171, 96, 82, 120, 126]. Thus, an outstanding open question in the area is:

Can we construct compact Functional Encryption for any nontrivial functionality from simple lattice assumptions?

Attribute Based Encryption. A special case of Functional Encryption is the notion of Attribute Based Encryption (ABE) [157, 116]. ABE is similar to FE but with a crucial difference – in ABE the computation is performed on *public* attribiutes encoded in the ciphertext, while the burden of privacy is only on an unchanging message. In more detail, the ciphertext encodes a public *attribute* $\mathbf{x}$ together with a secret message m, the secret key is generated for a public function f, and decryption outputs m if and only if $f(\mathbf{x}) = 1$. Security is similar to FE, except that $\mathbf{x}$ need not be hidden. This is formalized in an indistinguishability style game which asks that an adversary should be unable to distinguish between an encryption of $(m_0, \mathbf{x})$ and $(m_1, \mathbf{x})$, even given secret keys for functions f_i so long as $f_i(\mathbf{x}) = 0$ for all i. ABE comes in two avatars – "key-policy" where the function f is encoded in the secret key, or "ciphertext-policy" where it is encoded in the ciphertext. These are denoted by kpABE and cpABE respectively. An interesting generalization of ABE is the so-called "Predicate Encryption" (PE) [133] where the attribute $\mathbf{x}$ is also hidden but only against an adversary that does not receive any decrypting key, namely $f_i(\mathbf{x}) = 0$ for all f_i queried by the adversary.

Prior Work in ABE. There has been significant progress in constructing ABE for circuits over the last several years [116, 107, 43, 26, 169, 122] from well-understood assumptions. Here, there has been asymmetry between the key and ciphertext policy variants – while kpABE can be constructed from the standard Learning With Errors (LWE) assumption, it's cpABE counterpart additionally requires a relatively new, strong assumption called *Evasive* LWE [169, 163].

Recently, an important focus area in ABE research has been to achieve asymptotic optimality for both kpABE and cpABE schemes [170, 127]. The very recent work of Wee [170] constructs kpABE for *bounded* depth circuits using a new assumption called L-succinct LWE, which is weaker than evasive LWE. Assuming compact FE, [127] achieve optimality even for *unbounded* depth circuits, but this assumption necessitates the reliance on pairings as discussed above. We summarize the state of the art in Table 5.1. Thus, an outstanding open question in ABE research is:

Can we construct kp/cp ABE *for* P *with optimal parameters from lattices?*

Reference	KP/CP	|mpk|	|sk|	|ct|	Depth	Assumptions
[127]	KP	$O(1)$	$O(1)$	$O(1)$	Unbdd	Compact FE for P
[127]	CP	$O(1)$	$O(1)$	$O(1)$	Unbdd	Compact FE for P
[122]	KP	$O(L)$	$O(1)$	$O(L)$	Unbdd	evasive circular LWE + circular LWE
[13]	CP	$O(L)$	$O(L)$	$O(1)$	Unbdd	LWE + evasive LWE + circular tensor LWE
[170]	KP	$L^2 \cdot O(d)$	$O(d)$	$O(d)$	Bdd	L-succinct LWE
[123]	CP	$O(d)$	$L \cdot O(d)$	$L \cdot O(d)$	Bdd	LWE + evasive learning with structured errors
This Work	KP	$O(1)$	$O(1)$	$O(1)$	Unbdd	LWE + evasive LWE
This Work	CP	$O(1)$	$O(1)$	$O(1)$	Unbdd	LWE + evasive LWE

Table 5.1: State of the Art: Attribute Based Encryption

5.1.1 Our Results

In this work, we make progress on both the above questions and achieve the following: (i) we provide the first construction of compact FE from private-coin evasive LWE (and LWE) for a nontrivial class of *pseudorandom* functionalities, (ii) we use our new FE to construct kpABE and cpABE achieving *optimal* parameters for *unbounded depth* circuits, also from evasive LWE and LWE. We summarize our results as informal theorems below.

The first theorem shows that we can obtain partially hiding pseudorandom FE (PHprFE) with optimal parameter sizes. In PHprFE, an input $\mathbf{x}$ to the function f is divided into a public part $\mathbf{x}_{\mathsf{pub}}$ and private part $\mathbf{x}_{\mathsf{priv}}$. While we consider the same correctness requirement as usual FE, we consider a relaxed security notion where the public part is allowed to leak to the adversary. The advantage of allowing part of the input to be public

is that it leads to shorter ciphertexts whose size does not depend on the length of $\mathbf{x}_{\mathsf{pub}}$, when we follow the convention of ignoring the length of the public part when measuring ciphertext size.

Theorem 5.1 (Partially Hiding prFE for Unbounded Depth)**.** *Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for function class* $\mathcal{F} = \{f : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$*, that satisfies very selective security (more accurately, security as per Definition 5.20) whose master public key and the secret key sizes are fixed polynomial* $\mathrm{poly}(\lambda)$*. Furthermore, the size of the ciphertext is* $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Here, very selective security refers to the security notion where the adversary has to choose its challenge query and key queries before seeing the public parameter.

Note that in particular, this implies a plain FE scheme for pseudorandom functionalities with optimal parameters (by setting the public part of the input to $\perp$). We then leverage the above FE to construct Attribute Based and Predicate Encryption (ABE and PE respectively) for unbounded depth circuits with optimal parameters, in both the KP and CP setting. In more detail, we prove the following.

Theorem 5.2 (Optimal KP-ABE)**.** *[Theorem 5.30] Under the LWE and Evasive LWE assumptions, there exists very selectively secure KP-ABE scheme supporting circuits* $\{C : \{0,1\}^{\ell} \to \{0,1\}\}$ *with unbounded depth and message space* $\{0,1\}^{\lambda}$ *with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda).$$

Theorem 5.3 (Optimal KP-PE)**.** *[Theorem 5.32] Under the LWE and Evasive LWE assumptions, there exists very selectively secure KP-PE scheme supporting circuits* $\{C : \{0,1\}^{\ell} \to \{0,1\}\}$ *with unbounded depth and message space* $\{0,1\}^{\lambda}$ *with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}|$$

where $\mathbf{x} \in \{0,1\}^{\ell}$.

Theorem 5.4 (Optimal CP-ABE)**.** *[Theorem 5.34] Under the LWE and Evasive LWE*

assumptions, there exists very selectively secure CP-ABE scheme supporting circuits with unbounded depth $\{C : \{0,1\}^\ell \to \{0,1\}\}$ *and message space* $\{0,1\}^\lambda$ *with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda)$$

where $\mathbf{x} \in \{0,1\}^\ell$.

We observe that since our schemes support message space $\{0,1\}^\lambda$, they can be used to support arbitrary message space by using the hybrid encryption framework. We note that our unbounded CP-ABE scheme Theorem 5.34 and unbounded KP-ABE scheme Theorem 5.30 can be used to instantiate ABE for Turing Machines ([13]). We obtain the following corollary, which improves both the assumptions and the parameters of [13].

Corollary 5.4.1 (KP-ABE for TM)**.** *[Corollary 5.35.1] Under the LWE and evasive LWE assumptions, there exists a very selectively secure ABE for TM with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = |M| \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = |\mathbf{x}| \cdot t \cdot \mathrm{poly}(\lambda)$$

where the Turing machine M *runs on input* $\mathbf{x}$ *for time step* t.

In contrast, [13] uses LWE, evasive LWE and circular tensor LWE and achieves $|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}| = \mathrm{poly}(|M|, \lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

5.1.2 Additional Prior Work

The notion of iO for pseudorandom functionalities was considered implicitly by the work of Mathialagan, Peters and Vaikuntanathan [146] where they used subexponential LWE and evasive LWE to construct adaptively sound zero-knowledge SNARKs for UP. In a previous version of their work [147], which was in private circulation and shared with us, this notion was defined explicitly and leveraged to obtain unlevelled fully homomorphic encryption. However, an explicit construction of iO for pseudorandom functionalities was not provided. Moreover, FE or MIFE for any class of functionalities was not considered.

Our Companion Paper. In a companion work [14], we bootstrap our compact FE for pseudorandom functionalities to the multi-input setting. This yields the first multi-input FE for pseudorandom functionalities from LWE and (a stronger variant of) private-coin evasive LWE. Using the techniques of [29], this multi-input FE can be naturally deployed to construct the first iO for pseudorandom functionalities. We then leverage our pseudorandom multi-input FE and iO for applications – we refer the reader to [14] for details.

In the present work, our construction of ABE with optimal parameters (Section 5.5) uses a restricted iO (called pPRIO) as a building block which is developed in the companion work [14]. This creates a dependence between the two works. The rationale for our choice is primarily to have the most concise presentation and avoid duplication of content. We chose to split the papers across the axis of single input and multi-input since this seems most natural. The notion of pPRIO uses MIFE for constant arity, so fits naturally in the companion work. By *not* using it in the present work, we would achieve ABE schemes with sub-optimal parameters which nevertheless outperform the state of the art. We would then improve these schemes in the companion work, which was undesirable. Another alternative was to provide the multi-input compiler for constant arity in the present work and generalize this to polynomial arity (from a stronger assumption) in the companion work. However this would lead to duplicating the MIFE construction in both works. Therefore, we believe that using the pPRIO black-box as a tool in the ABE application developed here is the best option.

5.1.3 Recent Attacks on Evasive LWE and Repercussions.

Subsequent to the first online appearance of the present work, some counter-examples for evasive LWE were developed. We discuss these and their impact on the present work below. To begin, we recall the definition of evasive LWE.

The evasive LWE assumption [169, 163] roughly says that if

$$(\mathbf{B},\ \mathbf{P},\ \ \mathbf{s}^\top\mathbf{B}+\mathbf{e_B},\ \ \mathbf{s}^\top\mathbf{P}+\mathbf{e_P},\ \ \mathsf{aux}\ ) \approx_c (\ \mathbf{B},\ \mathbf{P},\ \$,\ \ \$,\ \ \mathsf{aux})$$

where \$ represents random, then

$$(\ \mathbf{B},\ \mathbf{P},\ \mathbf{s}^\top\mathbf{B}+\mathbf{e_B},\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux}) \approx_c (\ \mathbf{B},\ \mathbf{P},\ \ \$,\ \ \mathbf{B}^{-1}(\mathbf{P}),\ \ \mathsf{aux})$$

Above $\mathbf{B}^{-1}(\mathbf{P})$ refers to a low norm matrix, say $\mathbf{K}$, such that $\mathbf{BK} = \mathbf{P} \mod q$. It is clear that given $\mathbf{B}^{-1}(\mathbf{P})$ and $\mathbf{s}^\top\mathbf{B} + \mathbf{e_B}$, the adversary can compute $\mathbf{s}^\top\mathbf{P} + \mathbf{e_B}\mathbf{K}$. Evasive LWE intuitively says that the adversary cannot exploit $\mathbf{B}^{-1}(\mathbf{P})$ in any other way. The rationale behind the assumption, discussed at length in [169], is that the final value obtained by the adversary after decryption, represented above by $\mathbf{s}^\top\mathbf{P} + \mathbf{e_B}\mathbf{K}$, is large and completely avoids the so-called "zeroizing" regime, namely the situation where the attacker obtains low norm polynomial equations over the integers and can solve these to obtain harmful leakage. Thus, evasive LWE should allow us to construct schemes where the decryption yields a pseudorandom output, as required by the pre-condition. Indeed, this is the motivation for the name "evasive" LWE – it should only support construction of evasive functionalities where decryption of challenge ciphertexts is not allowed, such as ABE, but not FE/iO.

Evasive LWE has been studied in two main regimes, namely "public-coin" and "private-coin", where the former means that the randomness used to sample $\mathbf{P}$ and auxiliary information aux, is made available to the adversary, and the latter means that this information needs to be hidden. The private-coin version was known to have some contrived counter-examples since the work of [164], but this was not considered too problematic as it relied on highly unnatural auxiliary information which contained obfuscations that would output secrets given $\mathbf{B}^{-1}(\mathbf{P})$ but not otherwise. No attacks were known in the public-coin setting used by Wee's original formulation or its extensions, such as the circular evasive LWE by Hseih, Lin and Luo [122]. Thus, evasive LWE has been seen as a meaningful "middle point" in the land between LWE on one hand, and

lattice assumptions used for iO on the other. The community has tried to make progress on long standing "evasive" problems in the world of lattice based cryptography using this assumption, while simultaneously trying to improve the assumption (see for instance [170]).

New Attacks. In a very exciting, very recent development subsequent to the first online posting of the present work, some surprising new counter-examples against evasive LWE became known [19, 121, 85]. In a nutshell, these attacks show that the intuition that evasive LWE evades the zeroizing regime is not always true, even in the public-coin setting. However, by taking suitable precautions, security can be recovered as discussed below.

Malicious Sampler Attacks. The works of [19, 121, 85] demonstrated that the general formulation of evasive LWE, which allows for arbitrary malicious samplers, is *false as stated*. In particular, as related to the present work, the works of [19, 121] showed (via essentially the same attack) that by carefully crafting a contrived circuit to implement a PRF which is used (non black-box) in our scheme, the pre-condition can still be argued true while the post-condition can be shown false. Additionally, besides attacking private coin versions of Evasive LWE that are prevalent in the literature, the attacks by [19] also affect some public-coin variants, including the "circular, small-secret evasive LWE" by [122].

We view these attacks as an important step forward in our understanding of evasive LWE. Note that evasive LWE should be seen as a family of assumptions parametrized by the description of the sampler and choice of error distributions, which, if invoked in full generality, is now known to be false, even in the public-coin setting. However, as discussed in [19], the original intuition by Wee [169] and Tsabary [163] about the security of evasive LWE can be recovered by taking precautions to identify and respect a

"safe zone" for evasive LWE – thus, by refining/restraining the formulation of evasive LWE, even the original construction (presented in the first online posting of this work) is secure. In addition, we modify our original construction of prFE to implement a modulus reduction step, which negates the effect of choosing contrived circuits in the construction – for this modified construction, we do not even know of any attack using malicious samplers. We also remark that in the real world, the circuit representation of functions is chosen by the key generator who is an honest party (it holds the master secret key), and we view the counter-examples emerging from such circuit choices as indicating the limits of the assumption rather than the security of the existing constructions.

Contrived Functionality Attacks. The work of [65] and [19] show that there exists a contrived "self-referential" functionality for which pseudorandom functional encryption or pseudorandom obfuscation cannot exist. As discussed in [19], this result may be seen as analogous to the impossibilities known for the random oracle model [68] or virtual black box (VBB) obfuscation [34]. In more detail, despite impossibilities known for ROM and VBB obfuscation [68, 34], the meaningfulness of ROM for practical security, and of VBB obfuscation for restricted functionalities [167, 69] is accepted widely. The pseudorandom functionalities that are useful for our applications, such as computing blind garbled circuits or FE ciphertexts, are quite natural and do not fall prey to such attacks.

Attacks by withholding information about **B** *or* **P**. The elegant work of [66] presents attacks against classes of evasive LWE such that either **B** or **P** are not known to the adversary. In our case, both **B** and **P** are known to the adversary – indeed **P** is publicly computable using auxiliary information.

5.1.4 Technical Overview

In this section, we present the core ideas that we develop in this work. Below $\underline{X}$ denotes a noisy version of X where the exact value of noise is not important.

KP-ABE for Unbounded Depth by HLL. The seminal work of Boneh et al. [43] developed algorithms for evaluating arithmetic functions on the ciphertext as well as the public key of an ABE scheme, which form the cornerstone of several subsequent constructions. Their core technique is as follows: given an input $\mathbf{x} \in \{0,1\}^\ell$, and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on an "input encoding" matrix of form $\mathbf{A} - \mathbf{x} \otimes \mathbf{G}$ by multiplying on the right by a low norm matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ to obtain the term $\mathbf{A}_f - f(x)\mathbf{G}$. Here, $\mathbf{G}$ is a special gadget matrix defined as follows. Let $\mathbf{g} = [1, 2, 2^2, \ldots, 2^{\log q}]^\intercal$ and $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^\intercal$. In key evaluation, one can homomorphically evaluate a circuit $f : \{0,1\}^\ell \to \{0,1\}$ on $\mathbf{A}$ to obtain $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_{\mathbf{A},f}$ for some low norm matrix $\mathbf{H}_{\mathbf{A},f}$. In ciphertext evaluation, given an attribute $\mathbf{x}$ and corresponding attribute encoding of the form $\underline{\mathbf{s}^\intercal(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})}$, which we refer to as BGG^+ encoding, right multiplication by $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ yields $\underline{\mathbf{s}^\intercal(\mathbf{A}_f - f(\mathbf{x})\mathbf{G})}$ without substantially blowing up the noise in the encoding since $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ is low norm. Skipping several details, since the key generator can compute $\mathbf{A}_f = \mathbf{A} \cdot \mathbf{H}_{\mathbf{A},f}$, it can provide a matching key which allows the decryptor to cancel out the masking term $\mathbf{s}^\intercal\mathbf{A}_f$ and proceed with decryption. We refer to $\mathbf{H}_{\mathbf{A},f}$ and $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ as the PK and CT evaluation matrices respectively.

Providing Advice for Bootstrapping. The essential barrier in supporting circuits of unbounded depth for homomorphic computation is that the norm of the matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ grows exponentially with the depth of the circuit being computed, causing the the noise in the ciphertext encoding to blow out of control after some number of evaluations. The same barrier was encountered while constructing fully homomorphic encryption (FHE), to resolve which, Gentry proposed the beautiful "bootstrapping" idea. Roughly speaking, bootstrapping suggests performing homomorphic evaluation of the decryption circuit on

the large-noise ciphertext – this has the effect of internally throwing away the large noise accumulated in the ciphertext and "refreshing" it with smaller, more manageable noise. Since this procedure can be performed every time the ciphertext has accumulated large noise, the evaluator can keep on going! However, to execute this approach, one needs to assume that the scheme satisfies circular security, namely it should be safe to provide a ciphertext encrypting the scheme's own secret key, as this is required for homomorphic decryption described above.

While it has been long known how to use circular security in the context of unbounded depth FHE [97], its utility in the context of ABE was uncovered only very recently, in an elegant work by Hseih, Lin and Luo [122] (HLL). In more detail, HLL supports unbounded homomorphism in ABE via two steps: (i) noise removal, and (ii) bootstrapping. The noise removal step is via modulus reduction à la BV/BGV [62, 55] which destroys the algebraic structure of the BGG^+ encoding required for further evaluation, while the bootstrapping step makes use of a circular encoding, or "advice", which enables to restore the structure of the BGG^+ encoding, making it suitable for further evaluation. In more detail, the HLL advice has the following structure:

$$\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s}), \quad \mathbf{E} = \underline{\mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathbf{S} \otimes \mathbf{G})} \tag{5.1}$$

Above, $\mathsf{hct}_{\mathbf{s}}(\cdot)$ is an FHE ciphertext decryptable by secret key $\mathbf{s}$ denoted in the subscript – thus $\mathbf{S}$ is a circular FHE ciphertext, and $\mathbf{E}$ is a BGG^+ encoding with attribute $\mathbf{S}$ and *re-using* the FHE secret $\mathbf{s}$ as the LWE secret! The trick of reusing the FHE secret as the LWE secret in the BGG^+ encoding of attribute $\mathsf{hct}_{\mathbf{s}}(\cdot)$, was introduced by Brakerski et al. [60] and can lead to "automatic decryption" of the FHE ciphertext, as described next. Recall that in the GSW FHE scheme [99], the secret key is $\mathbf{s}^\intercal$, a ciphertext for message $\mathbf{y}^\intercal$ is a matrix $\mathbf{C}$ and decryption computes $\mathbf{s}^\intercal\mathbf{C}$ to recover (a noisy version of) $\mathbf{y}^\intercal$. Brakerski et al. [60] suggested "vectorizing" the BGG^+ ciphertext evaluation procedure so that homomorphic evaluation on the encoding produces a term of the form $\underline{\mathbf{s}^\intercal(\mathbf{A}_f - \mathsf{hct}_{\mathbf{s}}(\mathbf{y}^\intercal))}$ (i.e. without $\mathbf{G}$). Now, the inner product of $\mathbf{s}$ and $\mathsf{hct}_{\mathbf{s}}(\mathbf{y}^\intercal)$ causes

FHE decryption to occur automatically and we obtain the encoding $\underline{\mathbf{s}^\mathsf{T}\mathbf{A}_f + \mathbf{y}^\mathsf{T}}$, where the noise in the encoding is low. It turns out that this term is exactly what is needed to restore the structure of the ill-formed encoding obtained by step (i) of HLL – this allows to create a low noise BGG^+ encoding which can be used to evaluate further. Put together, HLL prove the following:

Theorem 5.5. *Assuming circular evasive LWE and LWE, there exists a very selectively secure* kpABE *scheme for circuits of unbounded depth and attribute length ℓ with* $|\mathsf{mpk}| = \mathrm{poly}(\lambda, \ell)$, $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, \ell)$.

Above *circular evasive LWE* is a new assumption introduced by HLL which "mixes" circularity into the recently introduced evasive LWE assumption [169, 163].

Randomizing Advice for CP-ABE. Recently, Agrawal, Kumari and Yamada [13] (AKY) built upon the construction by HLL to obtain the first ABE for Turing machines from lattice assumptions. A key technical contribution of the AKY construction is a way to randomize the advice provided in the HLL ciphertext, making it suitable for integration with Wee's bounded depth CP-ABE. These techniques led to the first CP-ABE for unbounded depth circuits, which they further leveraged to construct a (KP-)ABE for Turing machines. In more detail, the AKY transformation requires computation of the following randomized HLL terms:

$$\mathbf{S}_\mathbf{r} = \mathsf{hct}_{\mathbf{s}_\mathbf{r}}(\mathbf{s}_\mathbf{r}), \quad \mathbf{E}_\mathbf{r} = \underline{\mathbf{s}_\mathbf{r}^\mathsf{T}(\mathbf{A}_{\mathsf{circ}} - \mathbf{S}_\mathbf{r} \otimes \mathbf{G})} \tag{5.2}$$

Above, $\mathbf{s}_\mathbf{r} = \mathbf{s}^\mathsf{T}(\mathbf{I} \otimes \mathbf{r})$ where $\mathbf{r}$ is chosen by the key generator while $\mathbf{s}$ is chosen by the encryptor.

Evidently, neither party can provide the encodings directly, and while randomizing the message inside an FHE ciphertext from $\mathbf{s}$ to $\mathbf{s}_\mathbf{r}$ is easy given knowledge of $\mathbf{r}$, randomizing the secret key of FHE ciphertext is much more challenging. To get around this difficulty,

AKY suggest that the structure of the advice provided by the encryptor be changed, so that the true power of FHE – which is to transform encoded messages rather than underlying secret keys – be further leveraged. Thus, they provide:

$$\mathbf{T} = \mathsf{hct}_{\mathbf{t}}(\mathbf{s}, \mathsf{sd}), \quad \mathbf{D} = \underline{\mathbf{t}^{\intercal}(\mathbf{A}_1 - (1, \mathsf{bits}(\mathbf{T})) \otimes \mathbf{G})}$$

where sd is a PRF seed, $\mathbf{t}$ is the secret of a fresh FHE scheme and $\mathbf{A}_1$ is a public matrix of appropriate dimensions. Now, one can homomorphically evaluate on the encoding $\mathbf{D}$ in *bounded* depth, using knowledge of $\mathbf{T}$, to obtain

$$\underline{\mathbf{t}^{\intercal}\mathbf{A}'_{\mathbf{r}} + \mathbf{t}^{\intercal}\mathsf{hct}_{\mathbf{t}}(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})} = \underline{\mathbf{t}^{\intercal}\mathbf{A}'_{\mathbf{r}} + (\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})}$$

where $\mathbf{A}'_{\mathbf{r}}$ is some $\mathbf{r}$ dependent matrix and the equality follows by automatic decryption. To get rid of the masking term $\mathbf{t}^{\intercal}\mathbf{A}'_{\mathbf{r}}$, the encryptor additionally provides $\underline{\mathbf{t}^{\intercal}\mathbf{C}}$ for some fixed matrix $\mathbf{C}$ and the key generator provides $\mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}})$ where $\mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}})$ is not a true matrix inverse but rather a low norm matrix so that $\mathbf{C} \cdot \mathbf{C}^{-1}(\mathbf{A}'_{\mathbf{r}}) = \mathbf{A}'_{\mathbf{r}}$. Together these allow the decryptor to compute the term $\underline{\mathbf{t}^{\intercal}\mathbf{A}'_{\mathbf{r}}}$ and cancel it out from the encoding above, to recover $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ in the clear.

The security of the above construction, relies on evasive LWE (aside from other assumptions), and depends crucially on the fact that the computed terms $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ are pseudorandom. Digging deeper into the AKY proof, the term $\underline{\mathbf{s}^{\intercal}\mathbf{P}}$ in Evasive LWE can be essentially simplified to the advice terms $(\mathbf{S}_{\mathbf{r}}, \mathbf{E}_{\mathbf{r}})$ which, therefore need to be pseudorandom for invoking the assumption. Put together, AKY show the following:

Theorem 5.6. *[13][Thm 5.6] Under LWE, circular tensor LWE and evasive LWE, there exists a very selectively secure ABE for TM with* $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(\lambda, |M|)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

Above M is the Turing machine, $\mathbf{x}$ is the input and t is the worst case running time of M on any input. Note that while the AKY construction does not need the circular evasive LWE assumption used by HLL, they require a circular *tensor* assumption which is a new

assumption that they introduce.

Compact Functional Encryption for Pseudorandom Functionalities. Our starting point is the observation that the techniques developed in AKY are quite a bit more general and can be leveraged to compute functionalities beyond the randomized HLL advice they were developed for. Taking a step back, let us analyze what their technique enables: the encryptor provides an FHE ciphertext $\mathbf{T}$ of a message (say $\mathbf{x}$), and a BGG^+ encoding of attribute $\mathbf{T}$ with the FHE secret doubling up as the encoding randomness. Homomorphic evaluation of any function f coupled with automatic decryption allows to recover a masked version of $f(\mathbf{x})$ and evasive LWE allows to cancel the mask. Thus, this technique seems to enable computation of any function f on the input $\mathbf{x}$, while keeping it hidden! Intuitively, security follows from evasive LWE as long as the output of the functionality is pseudorandom, such as their $(\mathbf{S_r}, \mathbf{E_r})$, but more generally the output of any pseudorandom function, such as a PRF.

We show that the above intuition can be formalized to yield the first compact FE for pseudorandom functionalities, namely, functionalities where the output is (pseudo)random for any given input that is seen by the adversary in the security game. We sketch our construction for prFE below. In the following, $f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}$ has the property that the output of f is pseudorandom for every input seen by the adversary. We also use a $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to [-q/4, q/4]$. The usage of PRF is introduced for the security reasons which we will highlight later.

- The setup algorithm samples matrices $\mathbf{A}_{\mathsf{att}}$ and $(\mathbf{B}, \mathbf{B}^{-1})$ of appropriate dimensions and outputs $\mathsf{mpk} \coloneqq (\mathbf{A}_{\mathsf{att}}, \mathbf{B})$ and $\mathsf{msk} \coloneqq \mathbf{B}^{-1}$. Here, $\mathbf{B}^{-1}$ is the trapdoor for $\mathbf{B}$ which allows to compute short preimages $\mathbf{B}^{-1}(\mathbf{U})$ for any target matrix $\mathbf{U}$.

- The encryptor on input $\mathbf{x}$ first samples a GSW secret key $\mathbf{s}$ and a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$. It then computes a GSW ciphertext, $\mathbf{X} = \mathsf{hct}_{\mathbf{s}}(\mathbf{x}, \mathsf{sd})$, using public key $\mathbf{A}_{\mathsf{fhe}} = (\bar{\mathbf{A}}_{\mathsf{fhe}} \quad \bar{\mathbf{s}}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\intercal})$ and randomness $\mathbf{R}$, – followed by a BGG^+ encoding of $\mathbf{X}$ using randomness $\mathbf{s}$ as $\mathbf{c}_{\mathsf{att}}^{\intercal} \coloneqq \mathbf{s}^{\intercal}(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\intercal}$. It additionally computes $\mathbf{c}_{\mathbf{B}}^{\intercal} \coloneqq \mathbf{s}^{\intercal}\mathbf{B} + \mathbf{e}_{\mathbf{B}}^{\intercal}$ and outputs the ciphertext $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.

– The key generator on input $\mathsf{msk} = \mathbf{B}^{-1}$ and function f does the following.

(a) Samples a nonce $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ and defines function $\mathrm{F}[f, \mathbf{r}]$, with f and $\mathbf{r}$ hardwired, as

$$\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x})\, \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}).$$

It then computes the FHE evaluation circuit $\mathsf{VEval}_{\mathrm{F}}$ w.r.t. the function $\mathrm{F}[f, \mathbf{r}]$ (this can be computed using the knowledge of $\mathrm{F}[f, \mathbf{r}]$). Note that the circuit $\mathsf{VEval}_{\mathrm{F}}$ can be used to compute on a GSW ciphertext encoding an input, say $\mathbf{y}$, to recover a GSW ciphertext encoding $\mathrm{F}[f, \mathbf{r}](\mathbf{y})$.

(b) Next, it computes the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using the public matrix $\mathbf{A}_{\mathsf{att}}$. Recall that the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$ and $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$ (which can be computed given $\mathsf{VEval}_{\mathrm{F}}$, $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$) will satisfy the relation

$$(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G})\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} = \mathbf{A}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}} - \mathsf{VEval}_{\mathrm{F}}(\mathbf{X}).$$

(c) It sets $\mathbf{A}_{\mathrm{F}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$, samples $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$ and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.

– The decryption on input $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$ work as follows.

(a) It first computes the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using $\mathbf{A}_{\mathsf{att}}$ and $\mathbf{X}$.

(b) Next, it computes $\mathbf{z} \coloneqq \mathbf{c}^{\intercal}_{\mathbf{B}} \cdot \mathbf{K} - \mathbf{c}^{\intercal}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$, rounds $\mathbf{z}$ co-ordinate wise and output the most significant bits.

To see the correctness of our scheme, we note that

$$\mathbf{c}^{\intercal}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} \approx \mathbf{s}^{\intercal}\mathbf{A}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}} - \mathbf{s}^{\intercal}(\mathsf{hct}_{\mathbf{s}}(\mathrm{F}(\mathbf{x}, \mathsf{sd}))) \approx \mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{F}} - \mathrm{F}(\mathbf{x}, \mathsf{sd}),$$

where the second approximate equality follows by automatic decryption. Now to remove the masking term "$\mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{F}}$" we compute $\mathbf{c}^{\intercal}_{\mathbf{B}} \cdot \mathbf{K} \approx \mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{F}}$ and thus $\mathbf{z} \approx \mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{F}} - \mathbf{s}^{\intercal}\mathbf{A}_{\mathrm{F}} + \mathrm{F}(\mathbf{x}, \mathsf{sd}) = f(\mathbf{x})\, \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r})$. Now, rounding gives us bits of $f(\mathbf{x})$ as long as $|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})| \leq q/4$. The exact decryption error is

$$\mathbf{e}^{\intercal}_{\mathbf{B}}\mathbf{K} + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) - (\mathbf{e}^{\intercal}_{\mathsf{fhe}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}^{\intercal}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}})$$

where $\mathsf{VEval}_{\mathrm{F}}(\mathsf{bits}(\mathbf{X})) = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_{\mathrm{F}} - (\mathbf{0} \;\; \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}))^{\intercal}$.

Our construction supports functions of bounded polynomial depth $\mathsf{dep} = \mathrm{poly}(\lambda)$ and

has the following efficiency

$$|\mathsf{mpk}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

As seen above, our construction achieves compactness.

Security. Intuitively, our notion of security says that so long as the *output* of the functionality is pseudorandom, the *ciphertext* is pseudorandom, given all the additional information available to the adversary. We denote this security notion as prCT security. In more detail, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries. We say that a prFE scheme is secure if

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}}, \\ \left\{\mathsf{Enc}(\mathsf{mpk}, x_j)\right\}_{j \in [Q_{\mathsf{msg}}]},\ \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{Q_{\mathsf{key}}}} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}}, \\ \left\{\delta_j \leftarrow \mathcal{CT}\right\}_{j \in [Q_{\mathsf{msg}}]},\ \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{Q_{\mathsf{key}}}} \end{pmatrix} \tag{5.3}$$

$$\text{given} \quad \left(\mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}},\ \{f_i(x_j)\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]}\right) \approx_c \left(\mathsf{aux},\ f_1, \ldots, f_{Q_{\mathsf{key}}},\ \{\Delta_{i,j}\}_{i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]}\right) \tag{5.4}$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_i)$ for $i \in [Q_{\mathsf{key}}]$, $\mathcal{CT}$ is the ciphertext space and $\Delta_{i,j} \leftarrow \{0,1\}^\ell$ for $i \in [Q_{\mathsf{key}}]$, $j \in [Q_{\mathsf{msg}}]$.

The careful reader may have noticed that the above definition has a multi-challenge flavour, even though the construction is in the public-key setting. This peculiarity arises because single-challenge security does not generically imply multi-challenge security for our definition. To see this, recall the standard hybrid argument to prove multi-challenge security from single-challenge security: the proof follows a sequence of hybrids, where we simulate some of the ciphertexts honestly, while trying to change

a particular honest ciphertext to be random. Now, to generate honest ciphertexts, we need to know the corresponding plaintexts. However, this could ruin the precondition for invoking single-challenge security for the target ciphertext, since knowing some inputs may ruin the pseudorandomness of outputs, if the inputs are correlated to each other.

We observe that the above definition is incomparable to the standard indistinguishability (IND) security definition that is used in FE schemes [46]. Indeed, IND style security does not appear meaningful for our functionality. Recall that in the IND game, the adversary must submit two challenge messages m_0 and m_1 and request keys for functions f_i such that $f_i(m_0) = f_i(m_1)$. However since the output $f_i(m_b)$ is pseudorandom for $b \in \{0, 1\}$, the event $f_i(m_0) = f_i(m_1)$ occurs only with negligible probability[1]. However, our definition has a simulation security flavour and is very handy for applications, as we will see.

We provide some high level intuition for our proof of security for prFE. For simplicity, we focus here on the single challenge setting and refer the reader to the main body for the detailed proof in the multi-challenge setting. The proof begins by invoking evasive LWE with an appropriate sampler – this allows to reduce the reasoning to the distribution of the pre-condition, which replaces the term $\mathbf{K} = \mathbf{B}^{-1}(\mathbf{A}_F)$ with $\mathbf{c}_{\mathbf{B}}^{\intercal} \cdot \mathbf{K} = \underline{\mathbf{s}^{\intercal}\mathbf{A}_F}$. Now, as we see in Equation (5.1.4),

$$\mathbf{c}_{\mathsf{att}}^{\intercal} \cdot \mathbf{H}^{F}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} = \underline{\mathbf{s}^{\intercal}\mathbf{A}_F - \mathrm{F}(\mathbf{x}, \mathsf{sd})} = \underline{\mathbf{s}^{\intercal}\mathbf{A}_F - f(\mathbf{x})\lfloor q/2 \rceil - \mathsf{PRF}(\mathsf{sd}, \mathbf{r})}.$$

This allows to simplify these two terms to $\underline{\mathbf{s}^{\intercal}\mathbf{A}_F}$ and $\underline{f(\mathbf{x})\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r})}$, where the latter term is pseudorandom, hence simulatable and can be ignored hereafter. The terms that remain can now be handled by relying on LWE using standard techniques [169, 122, 13]. Please see Section 5.3 for the detailed proof.

[1] A generalization of IND security which requires $f_i(m_b)$ to be pseudorandom and hence computationally indistinguishable for any b might have been more suitable.

Handling Malicious Samplers. As discussed above, subsequent to the initial posting of this work on eprint, [19, 121, 85] demonstrated that the general formulation of evasive LWE is false as stated. As suggested in [19], there are multiple ways to restrain malicious samplers to prevent such attacks. The simplest one is to leverage the fact that the secret key is computed by the key generator who is an honest party (it holds the master secret key) in the real world, and can ensure that the circuit representation of any function f as well as the PRF can be made canonical by using the universal circuit or a garbled circuit representation. Nevertheless, here, we present an alternate fix to the scheme which uses modulus reduction to "throw away" the accumulated error after FHE evaluation, replacing it with rounding error which is no longer correlated with the PRF seed, even for a contrived circuit chosen by the adversary. To begin, we provide a high level outline of the attack.

Attack by [19, 121] The attacks proposed by [19, 121] on the (private-coin) Evasive LWE (Assumption 2.6) for *any* sampler breaks our initial construction. At a very high level, the attack, building upon clever ideas by [120], shows a way to create a correlation between the error term resulting from FHE evaluation (and automatic decryption) with the PRF output by using a contrived circuit to implement the PRF. In more detail,the adversary, given $\mathbf{c_B}$, $\mathbf{c}_{\mathsf{att}}$, $\mathbf{X}$, and $\mathbf{K}$, computes $\mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$. Simplifying, she obtains

$$f(\mathbf{x})\,\lfloor q/2 \rceil + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} + \mathsf{PRF}(\mathsf{sd},\mathbf{r}) - (\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}})$$

By correctness, the adversary recovers $f(\mathbf{x})$ and can therefore strip it away to obtain $\mathsf{PRF}(\mathsf{sd},\mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K} - \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$, as described in Section 5.1.4. Now, in the proof, the error term $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K}$ is replaced by i.i.d error $\mathbf{e_P}$ and is used to break any correlation between $\mathsf{PRF}(\mathsf{sd},\mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathrm{F}} - \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$. This allows us to prove the pre-condition based on just plain LWE.

However, in the real world, $\mathbf{e}_{\mathbf{B}}^{\mathsf{T}}\mathbf{K}$ cannot be used to break the dependence between the

above two terms. Then, by choosing the circuit implementing F in some contrived way, the authors set it up so that $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ and $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}}$ are correlated, and in particular cancel each other modulo 2. Note that these terms are small, and do not wraparound modulo q, so computing mod 2 is well defined. Now we are left with $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K} + \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$ – but these are linear equations with known coefficients, in the error terms of the original encodings. Using sufficiently many equations, the adversary can easily recover the error terms. On the other hand, had the term $\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K}$ been truly random, such a system of equations would not admit any solution. This leads to a distinguishing strategy.

Fixing the Scheme. We describe an approach that helps us break the problematic correlation even if the circuit implementation is chosen in a contrived manner – our idea is to use modulus reduction get rid of the problematic error terms involving $\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}}$ so that the correlation is destroyed. Informally, we fix a rounding constant $M \in \mathbb{Z}$ such that $\left\|\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}}\right\| < M$ which implies $\left\lfloor (\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}})/M \right\rceil = 0$. This gets rid of the problematic error, replacing it with rounding error which is uncorrelated with the PRF seed. For concreteness, we elaborate the changes that must be made to our prFE construction to incorporate the above fix.

1. In setup algorithm, we output M as a part of mpk. The encryption algorithm remains the same.

2. The key generation algorithm has the following changes.

   - We parse $\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x})\,\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) = M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \in [0, q/M]^{\ell}$ and $f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \in [0, M-1]^{\ell}$. Next we define functions $\mathrm{F}_{\mathsf{high}} := M \cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M \cdot f_{\mathsf{low}}$, which on input $(\mathbf{x}, \mathsf{sd})$ outputs $M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd})$ and $M \cdot f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, respectively.

   - Next, it computes circuits $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ for the functions $\mathrm{F}_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}}$, respectively and then uses these circuits to compute the matrices $\mathbf{H}^{\mathrm{F}_{\mathsf{high}}}_{\mathbf{A}_{\mathsf{att}}}$ and $\mathbf{H}^{\mathrm{F}_{\mathsf{high}}}_{\mathbf{A}_{\mathsf{att}}}$ (as described in the previous sketch).

   - It sets

$$\mathbf{A}_{\mathrm{F}} = M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}_{\mathsf{high}}}_{\mathbf{A}_{\mathsf{att}}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}_{\mathsf{low}}}_{\mathbf{A}_{\mathsf{att}}}}{M} \right\rfloor$$

and outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ where $\mathbf{K} = \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$.

3. The decryption algorithm is the same except that we compute $\mathbf{z}$ differently as

$$\mathbf{z} := \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rfloor \right)$$

We expand the correctness of the scheme to see how the above changes helps us get rid of the problematic noise terms while decryption. Observe that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}) \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}} \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
\implies \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathsf{high}} - M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}} \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor \\
&= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}} \mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}} \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rfloor - f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \\
&= \mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor - f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \quad \text{w.h.p.}
\end{aligned}$$

where we set $\left\| \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}} \mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}} \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \right\| \ll M$ such that the last equation in the above will hold with high probability. Similarly, we get $\left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rfloor = \mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor - f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$ w.h.p. The rest of correctness follows from the same argument as in the previous sketch. Note that the final error obtained now is $\mathsf{PRF}(\mathsf{sd}) + \mathbf{err}$ where (please see Equation (5.17))

$$\begin{aligned}
\mathbf{err} &= M \cdot \mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} + \mathbf{e}_{\mathbf{s},\mathsf{low}}^{\mathsf{T}} + M \cdot \mathbf{err}_{\mathsf{high}} + \mathbf{err}_{\mathsf{low}} \\
&= M \cdot \left(\mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor - \left\lfloor \mathbf{s}^{\mathsf{T}} \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor \right) + \left(\mathbf{s}^{\mathsf{T}} \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor - \left\lfloor \mathbf{s}^{\mathsf{T}} \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor \right) + M \cdot \mathbf{err}_{\mathsf{high}} + \mathbf{err}_{\mathsf{low}}
\end{aligned}$$

where $\mathbf{err}_{\mathsf{high}}, \mathbf{err}_{\mathsf{low}} \in \{0,1\}^{\ell}$ are rounding errors and matrices $\mathbf{A}_{\mathsf{high}}, \mathbf{A}_{\mathsf{low}}$ are publicly computable matrices. We conjecture flooding for appropriately defined sizes.

Relation to Circular Small-Secret Evasive LWE of HLL23. In Section 5.3.4, we show that our prFE construction can be based on a variant of the circular small-secret evasive LWE of [122] – the only difference between the assumption stated in [122] and the one

used in Section 5.3.4 is that the FHE encoding in [122] only encodes the secret $\mathbf{s}$, namely $\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s})$, whereas in our case, this must additionally include $\mathbf{x}$, i.e. $\mathbf{S} = \mathsf{hct}_{\mathbf{s}}(\mathbf{s}, \mathbf{x})$. This difference is created by the distinction between ABE and FE, since the attribute $\mathbf{x}$ can be public in the former and must be hidden in the latter. Hence, in [122], $\mathbf{x}$ can be output by the sampler directly, whereas in our case, it cannot.

Whether this fundamentally changes the assumption is a matter of opinion – in any event the [122] version of circular small-secret evasive LWE is also broken in [19] by a very similar attack as on the private coin evasive assumption – but we find the connection interesting since the assumption of [122] is considered public coin. We show in Section 5.3.4 that similar refinement of this assumption can also avoid known attacks.

Applications. While it is exciting to have a compact FE scheme for any nontrivial functionality from purely (conjectured) post-quantum assumptions, we show that our prFE is also surprisingly powerful, and yields important applications.

Removing Circularity from HLL. We demonstrate the utility of prFE by showing that it can be used to bootstrap a very weak kpABE scheme into a full fledged one. This enables us to improve the assumptions underlying prior works as discussed above.

In more detail, our weak kpABE scheme, denoted by $\mathsf{1ABE}$ is a *secret key* scheme which only supports a *single* ciphertext and *single* secret key query – this object is so simple that it can be constructed merely from one way functions. This is lifted using prFE to build a full fledged *public key* ABE scheme supporting *unbounded* ciphertexts and *unbounded* key queries. Our compiler does require $\mathsf{1ABE}$ to satisfy some structural properties:

1. Decomposability: The computation $\mathsf{1ABE.KeyGen}(C)$ can be decomposed into $\{\mathsf{1ABE.KeyGen}_i(C_i)\}_{i\in|C|}$ where C_i denotes the i-th gate of C and has fixed polynomial size. Here, the depth of $\mathsf{ABE.KeyGen}_i$ is fixed and independent of the parameters of C. Moreover, output of $\mathsf{1ABE.Enc}$ should be computable by a circuit of fixed depth, irrespective of the length of $\mathbf{x}$ input to $\mathsf{1ABE.Enc}$.

2. Blindness: The 1ABE ciphertext and secret key should be pseudorandom when decryption is not allowed.

Given the above properties, the core idea is to use prFE to generate randomized versions of bits of 1ABE secret keys and ciphertexts using randomness generated jointly by the encryptor and the key generator. This is supported by prFE of fixed depth because of property 1 and respects the constraints imposed by prFE because of property 2. Thus we obtain a public key scheme which supports unbounded ciphertexts and keys. We outline our compiler below.

- The setup algorithm generates $(\mathsf{prFE.msk}, \mathsf{prFE.mpk})$ using $\mathsf{prFE.Setup}$ and outputs these as msk and mpk respectively.

- The encryption algorithm on input mpk, attribute $\mathbf{x}$ and message μ computes a prFE ciphertext, $\mathsf{prFE.ct}$, encoding input $(\mathbf{x}, \mu, \mathsf{sd})$ where $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ is a PRF seed.

- The keygen algorithm on input $\mathsf{msk} = \mathsf{prFE.msk}$ and circuit C works as follows. It samples nonce $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and defines functions $\mathrm{F}_{\mathsf{key},i}[\mathbf{r}, C_i]$, with $\mathbf{r}$ and i-th gate of C hardwired, for $i \in [|C|]$ and $\mathrm{F}_{\mathsf{ct}}[\mathbf{r}]$ as follows

  (a) $\mathrm{F}_{\mathsf{key},i}[\mathbf{r}, C_i]$ on input $(\mathbf{x}, \mu, \mathsf{sd})$, first computes $\mathsf{1ABE.msk}$ using the randomness $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$, i.e. $\mathsf{1ABE.msk} \leftarrow \mathsf{1ABE.Setup}(1^\lambda; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$ and then outputs $\mathsf{1ABE.sk}_{C_i} \leftarrow \mathsf{1ABE.KeyGen}_i(\mathsf{1ABE.msk}, C_i)$.

  (b) $\mathrm{F}_{\mathsf{ct}}[\mathbf{r}]$ on input $(\mathbf{x}, \mu, \mathsf{sd})$, first computes $\mathsf{1ABE.msk}$ as above and then outputs an $\mathsf{1ABE.ct}$ encoding message μ w.r.t. attribute $\mathbf{x}$.

  It then computes prFE keys $\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}$ and $\mathsf{prFE.sk}_{\mathsf{ct}}$ corresponding to functions $\{\mathrm{F}_{\mathsf{key}}[\mathbf{r}, i]\}_{i\in[|C|]}$ and $\mathrm{F}_{\mathsf{ct}}[\mathbf{r}]$, respectively. It outputs $\mathsf{sk}_C = (\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}, \mathsf{prFE.sk}_{\mathsf{ct}})$.

- The decryption algorithm on input $\mathsf{sk}_C = (\{\mathsf{prFE.sk}_{\mathsf{key},i}\}_{i\in[|C|]}, \mathsf{prFE.sk}_{\mathsf{ct}})$ and $\mathsf{ct} = \mathsf{prFE.ct}$ first runs the prFE decryption, using prFE keys and ciphertext $\mathsf{prFE.ct}$, to compute

$$\mathrm{F}_{\mathsf{key},i}[\mathbf{r}, C_i](\mathbf{x}, \mu, \mathsf{sd}) = \mathsf{1ABE.sk}_{C_i}, \quad \mathrm{F}_{\mathsf{ct}}[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mathsf{1ABE.ct}.$$

  Finally it sets $\mathsf{1ABE.sk}_C = (\mathsf{1ABE.sk}_{C_1}, \ldots, \mathsf{1ABE.sk}_{C_{|C|}})$ and outputs the decryption result as $\mathsf{1ABE.Dec}(\mathsf{1ABE.sk}_C, \mathsf{1ABE.ct})$.

Correctness follows from those of the prFE and 1ABE: By the correctness of prFE, the decryptor recovers the ciphertext and secret key pair of 1ABE and by the correctness

of 1ABE, one can recover the message μ when $C(\mathbf{x}) = 1$. To prove the security, it suffices to show that ct = prFE.ct is pseudorandom. By the security of prFE, it suffices to show that the decryption results of the ciphertext using the secret keys are jointly pseudorandom. This follows from the security of 1ABE, since the decryption results are ciphertext and secret key pairs, where the decryption is not possible for each pair. Please see Section 5.B for further details.

Building 1ABE. It remains to instantiate 1ABE with the desired properties. Fortunately, these properties are relatively weak and easily satisfied. For instance, we can instantiate 1ABE simply by using blind garbled circuits [58] (Section 5.2.1) – here *blindness* is precisely the property that the garbled circuit and its labels should be pseudorandom, when the evaluation result of the garbled circuit using the given labels is random. Given a blind garbled circuit, the labels of the garbled circuit form the 1ABE ciphertext, the set of garbled gates form the 1ABE key, and decomposability follows from the structure of a garbled circuit, where we can garble each gate independently. Care needs to be taken to modify the circuit C in the secret key so that when $C(\mathbf{x}) = 0$, it outputs a random string rather than $\perp$ so as to be compatible with our prFE. This yields the following theorem.

Theorem 5.7. *Assuming LWE and Evasive LWE, there exists a very selectively secure* kpABE *scheme for circuits of unbounded depth and attribute length ℓ with* $|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda)$.

Note that HLL relied on a new assumption called circular evasive LWE, which we replace by simple evasive LWE above albeit at the cost of larger parameters – in particular the secret key is large and scales with $|C|$.

Next, we show that by using the abstraction of Attribute Based Laconic Function Evaluation (abLFE) [153], we can match the parameters by HLL. Intuitively abLFE allows to compress a circuit C into a short digest, which is then used by an encryptor to compute a ciphertext ct for some attribute, message pair $(\mathbf{x}, \mu)$. The decryptor, given C,

ct can recover μ if and only if $C(\mathbf{x}) = 1$. For our compiler, we require an abLFE scheme where the encryption algorithm can be decomposed into an offline and an online phase. The offline encryption algorithm takes as input $(\mathbf{x}, \mu)$ and outputs $\mathsf{ct}_{\mathsf{off}}$ and a private state st. The online encryption algorithm takes as input $(\mathsf{st}, \mathsf{digest})$ and outputs $\mathsf{ct}_{\mathsf{on}}$. This property is satisfied by the construction of [122].

At a high level, our kpABE uses the compression of the abLFE to shorten the secret key. The key generation computes a digest $\hat{C}$ for the circuit C, then computes a prFE key for a circuit which outputs the online part of abLFE ciphertext. The encrypt algorithm computes the offline part of the abLFE ciphertext and the state st using input $(\mathbf{x}, \mu)$. It then encrypts st using prFE encryption. Now, prFE decryption allows to recover the online part of the ciphertext, and abLFE decryption allows to recover μ if $C(\mathbf{x}) = 1$. We refer the reader to Section 5.B.3 for details. We prove the following theorem:

Theorem 5.8. *Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure* kpABE *scheme for circuits of unbounded depth and attribute length* ℓ *with*

$$|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\ell, \lambda).$$

Above, the parameters achieved match those of HLL and the assumptions are strictly weaker.

Constructing kpABE for Turing machines from Weaker Assumptions. Next, we turn our attention to kpABE for Turing machines. We show that using prFE and a kpABE for unbounded depth circuits, we can build a cpABE for unbounded depth circuits and further a kpABE for Turing machines with parameters matching AKY. The construction for cpABE is as follows.

– The setup algorithm generates $(\mathsf{prFE.msk}, \mathsf{prFE.mpk})$ using $\mathsf{prFE.Setup}$ and outputs these as $\mathsf{cpABE.msk}$ and $\mathsf{cpABE.mpk}$ respectively.

– The encryption algorithm, given circuit C and message μ, works as follows: It samples randomness R_{key} and computes $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) = \mathsf{kpABE.Setup}(1^\lambda, 1^L; R_{\mathsf{key}})$. Next, it computes a $\mathsf{prFE.ct}$ encoding $(R_{\mathsf{key}}, \mathsf{sd}, \mu)$, where sd is a PRF key, and a kpABE secret key for circuit C as $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$. It outputs $\mathsf{cpABE.ct} := (\mathsf{prFE.ct}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C)$.

– The key generation algorithm, given msk and attribute $\mathbf{x}$ outputs a prFE key, $\mathsf{prFE.sk}_{\mathrm{F}}$, for the function $\mathrm{F}[\mathbf{x}, \mathbf{r}]$ where $\mathbf{r} \leftarrow \{0,1\}^\lambda$. It outputs $\mathsf{cpABE.sk}_{\mathbf{x}} := \mathsf{prFE.sk}_{\mathrm{F}}$.
The function $\mathrm{F}[\mathbf{x}, \mathbf{r}]$ on input $(R_{\mathsf{key}}, \mathsf{sd}, \mu)$ first computes $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$ using the randomness R_{key} and then outputs a kpABE ciphertext $\mathsf{kpABE.ct}$ encoding μ w.r.t. attribute $\mathbf{x}$ using randomness $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$.

– The decryption algorithm on input secret key $\mathsf{prFE.sk}_{\mathrm{F}}$ and ciphertext $\mathsf{cpABE.ct} := (\mathsf{prFE.ct}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C)$ first runs the prFE decryption to obtain $\mathrm{F}[\mathbf{x}, \mathbf{r}](R_{\mathsf{key}}, \mathsf{sd}, \mu) = \mathsf{kpABE.ct}$ and finally performs kpABE decryption using $\mathsf{kpABE.ct}$ and $\mathsf{kpABE.sk}_C$.

Correctness follows from those of kpABE and prFE: The decryption of prFE ciphertext using the prFE secret key yields kpABE ciphertext encrypted under $\mathbf{x}$. This kpABE ciphertext can be decrypted using $\mathsf{kpABE.sk}_C$ when $C(\mathbf{x}) = 1$. To prove the security, we show $\mathsf{prFE.ct}$ is pseudorandom. By the security of prFE, it suffices to show that the decryption results of $\mathsf{prFE.ct}$ are pseudorandom. This follows from the security of kpABE, since the decryption results are kpABE ciphertexts which cannot be decrypted by $\mathsf{kpABE.sk}_C$. We finally note that while the above overall proof strategy works when the underlying kpABE has pseudorandom ciphertext, we have to consider more general case where underlying kpABE does not necessarily have pseudorandom ciphertext. Therefore, the final construction as well as the security proof are slightly different. We refer the reader to Section 5.7 for details. Thus, we obtain the following theorem.

Theorem 5.9. *Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure* cpABE *scheme for circuits of unbounded depth and attribute length* ℓ *with*

$$|\mathsf{cpABE.mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{cpABE.sk}_{\mathbf{x}}| = \mathrm{poly}(\ell, \lambda),\ |\mathsf{cpABE.ct}| = \mathrm{poly}(\lambda).$$

We note that the cpABE scheme instantiated as above replaces the reliance on circular tensor LWE and LWE assumptions used by AKY by simply circular LWE. It also achieves a shorter mpk as compared to that of AKY (which is $\mathrm{poly}(\lambda, \ell)$), other parameters being the same.

AKY provided a compiler that uses kpABE for bounded depth circuits and cpABE for unbounded depth circuits to achieve kpABE for Turing machines. Plugging our new cpABE into this compiler, we obtain:

Corollary 5.9.1. *Under the circular LWE assumption and evasive LWE assumption, there exists a very selectively secure ABE for TM with the following parameters:*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, |M|), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t).$$

In terms of parameters, the above theorem matches those of AKY. In terms of assumptions, it provides a strict improvement, as discussed above. Later, by instantiating the underlying kpABE to be one with optimal parameters (constructed below), we will obtain a cpABE scheme for unbounded depth circuits with optimal parameters – please see Section 5.7 for details.

The Quest for Optimal Parameters

Achieving optimal parameters is a central open question in the construction of ABE schemes, and one which has received much attention in the literature [127, 170]. Ideally, we wish to obtain optimal parameters for kpABE and cpABE for *unbounded* depth circuits and moreover, from the weakest assumptions possible. So far, we do not know how to step around the usage of evasive LWE either for unbounded depth kpABE or even for bounded depth cpABE, so an outstanding open question is:

Can we construct kp/cp ABE *for unbounded depth circuits with optimal parameters from* LWE *and evasive* LWE*?*

Below, we answer the above question in the affirmative. To achieve optimality, we will need a result from our companion paper [14], which we use black-box below. While reliance on the companion work is not needed to improve the assumptions and match the parameters from HLL and AKY as discussed above, we are aiming for optimality, which necessitates this dependence as discussed previously.

Wishful Thinking: Obfuscation to the Rescue? Since we do not wish to rely on any kind of circularity assumption, we return to our construction of kpABE from 1ABE (which we built using blind garbled circuits) and prFE as a starting point, and ask if we can compress the parameters. Observe that the kpABE secret key in this case contains prFE secret keys for outputting the randomized versions of the 1ABE ciphertext and secret key, i.e. $\mathsf{sk}_C = \{\mathsf{prFE.sk}_{\mathsf{key},i}, \mathsf{prFE.sk}_{\mathsf{ct},j}\}_{\{i,j\}}$. As discussed above, this approach crucially relies on the decomposability of the secret key and ciphertext of the underling 1ABE scheme, which allows it to randomize and output these "piece-wise" – gate by gate for the circuit and bit by bit for the input.

Suppose, as wishful thinking, one could have an obfuscation of a program which, given an index within the 1ABE ciphertext and secret key, could output the appropriate randomized garbled gate/label – this would allow to make the secret key and ciphertext size independent of the circuit and attribute size, and allow us to achieve optimal compression. Can such an approach be realized?

Let us examine the possibility. It is well known [29, 41] that compact FE for general circuits can be compiled into compact multi-input FE, which in turn can be compiled into iO for general circuits. Perhaps we can hope that such a compiler also be applied to our compact prFE to obtain multi-input prFE and iO for pseudorandom functionalities. Indeed this is true – in our companion work [14], we show that a similar compiler as [29, 41] can be made to work, albeit via a very different proof technique (since our security notion is itself quite different). However, even assuming the existence of iO for

pseudorandom functionalities, the problem is not solved because:

1. The FE to iO compiler suffers exponential loss in the reduction. Therefore, if we use iO, we do not get security from polynomial assumptions. Since we are finally constructing only ABE schemes, the reliance on exponential security feels too strong.

2. Even if we use iO, the problem of compression does not get solved because we require the program to "authenticate" the input and only provide the output for the legitimate input. As an example, it should not be possible to obtain a ciphertext component for $x_i = 0$ if $x_i = 1$. This would require hardwiring the input $\mathbf{x}$ (similarly C) into the obfuscation which would bring us back to square one!

We resolve these issues by observing that the second issue can, surprisingly, be used to overcome the first. This leaves us with the second issue, to resolve which, we define a new notion which we call *laconic* poly-domain obfuscation for pseudorandom functionalities, which may be of independent interest.

Let us begin with trying to resolve the first issue: observe that the second issue tells us that we need obfuscation for a very special input domain – one that corresponds to the gates of our particular C or the bits of our particular $\mathbf{x}$, and that inputs outside this domain should not be accepted. This immediately has the effect of shrinking down the domain size from exponential to polynomial! While it is as-yet unclear how to force the domain to be restricted to the special polynomial sized set we require, it at least shows that supporting a polynomial sized domain suffices. We term an obfuscation for circuit with polynomial sized domain as "Poly-Domain Obfuscation" [2]. We define iO for pseudorandom circuits which have polynomial sized domain (denoted by pPRIO) as follows.

1. $\mathsf{Obf}(1^\lambda, C) \rightarrow \mathsf{Obf}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : [N] \rightarrow [M]$ with $\mathsf{size}(C) \leq L$ for some arbitrary polynomial $L = L(\lambda)$. It outputs an obfuscation of the circuit Obf.

2. $\mathsf{Eval}(\mathsf{Obf}, x) \rightarrow y$. The evaluation algorithm takes as input an obfuscated circuit Obf and an input $x \in [N]$. It outputs $y \in [M]$.

[2]The informed reader may wonder about the connection with XIO [144] – note that in XIO, the size of the obfuscated circuit is allowed to be only slightly sublinear in the size of the truth table. In contrast, the efficiency requirement of our notion is the same as standard iO.

Security states that $\mathsf{Obf}(1^\lambda, C)$ is pseudorandom if the truth table of C is pseudorandom. Please see Section 5.2.3 for a formal definition. Restricting the input space to be of polynomial size helps us to avoid the exponential loss of the transformation as discussed above. We now invoke a theorem from our companion work:

Theorem 5.10 ([14])**.** *Assuming LWE and evasive LWE assumptions, there exists a secure* pPRIO *scheme.*

We now turn to the second issue, which we resolve by defining a new primitive as described next.

Laconic Poly-Domain Obfuscation for Pseudorandom Functionalities. In pPRIO, we obfuscate circuits with a polynomial-sized input domain of the form $1, 2, \ldots, N$. However, we must now find a way to have more control on what this polynomial set can be, so that we can implement some "authentication" of the inputs without making the size of the obfuscated circuit grow with the domain size, as discussed above. Towards this, we extend pPRIO to introduce the new notion of *laconic* pPRIO, where the input domain can be defined as $X \coloneqq X_1, \ldots, X_N$ for arbitrary strings $X_1, \ldots, X_N \in \{0,1\}^\ell$, with arbitrary length ℓ. The obfuscated circuit allows for the evaluation of inputs that are in the set $X \coloneqq \{X_1, \ldots, X_N\}$, but it does not allow evaluation for any inputs outside this set. This is exactly what we want! Looking ahead, the set $X_1, \ldots, X_N$ will be instantiated with the gates of C or the bits of $\mathbf{x}$ in the final construction.

To define laconic pPRIO, we modify the pPRIO syntax so that, in order to obfuscate a circuit with a restricted input domain X, the obfuscator only needs a short digest of X, whose size does not depend on N, rather than the entire description of it. This results in a compact obfuscation whose size is independent of N. Now, the evaluation of an input belonging to X can be performed given the description of X in the clear. We present our definition next.

1. $\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]}) \to \mathsf{dig}$. The digest algorithm takes as input the

security parameter λ and an input space X of the form $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ for some $\ell = \ell(\lambda)$ and $N \in \mathbb{N}$. We assume that X encodes the information of ℓ and N and one can retrieve them efficiently. It outputs a string dig.

2. $\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E) \to \mathsf{Lobf}$. The encode algorithm takes as input the security parameter λ, string dig and a circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ whose size is S. It outputs a ciphertext Lobf.

3. $\mathsf{LEval}(X, \mathsf{Lobf}) \to Y$. The decode algorithm takes as input $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ and Lobf. It outputs $Y = \{Y_i \in \{0,1\}^L\}_{i\in[N]}$.

For efficiency, we need that the size of $\mathsf{dig} = \mathsf{LDigest}(1^\lambda, X)$ should be bounded by $\mathrm{poly}(\lambda, S)$. In particular, the size of dig should be independent of N. For security, we require (roughly) that:

$$\text{If } (X, E(X_1), \ldots, E(X_N)) \approx_c \left(X, \Delta_1 \leftarrow \{0,1\}^L, \ldots, \Delta_N \leftarrow \{0,1\}^L\right)$$
$$\text{then } \left(X, \mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)\right) \approx_c (X, \delta \leftarrow O)$$

where $\mathsf{dig} \leftarrow \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]})$ and O denotes the output space of $\mathsf{LObfuscate}$ algorithm.

Constructing Laconic Poly-Domain Obfuscation. It remains to build laconic pPRIO. To do so, we use pPRIO in addition to blind garbled circuits (bGC) and blind batch encryption (BBE). Since pPRIO and bGC have been introduced earlier, we recall the notion of BBE [58] here. Intuitively, BBE allows to hash an input $\mathbf{x}$ so that encryption is performed against the hash of $\mathbf{x}$ instead of $\mathbf{x}$ together with some index $i \in [|\mathbf{x}|]$, and decryption allows to recover one of two messages depending on the i^{th} bit of $\mathbf{x}$[3].

In more detail, a BBE scheme $\mathsf{BBE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{SingleEnc}, \mathsf{SingleDec})$ is defined as follows: The setup algorithm on input security parameter λ and key length N outputs a common reference string crs. The Gen algorithm on input crs and a string $\mathbf{x} \in \{0,1\}^N$ (to be used as secret key) outputs a hash value h to be used as a public key. The

[3]The informed reader may notice similarities with laconic oblivious transfer [73].

SingleEnc algorithm on input $(\mathsf{crs}, h, i \in [N], (m_0, m_1) \in \mathcal{M}^2)$ outputs a (single) ciphertext ct. The SingleDec algorithm on input $(\mathsf{crs}, \mathbf{x}, i \in [N], \mathsf{ct})$ outputs a message $m \in \mathcal{M}$. For correctness, it is required that m_b is recovered if $x_i = b$ and h is the hash of $\mathbf{x}$. Security requires the other message to remain hidden. The blindness of the scheme states that if the encrypted message is random then the ciphertext is indistinguishable from random. We give a construction of a BBE scheme from LWE satisfying these properties in Section 5.A.

The basic intuition behind the construction of laconic pPRIO (denoted by LprIO) is as follows. To restrict the circuit evaluation to a special input domain X, we use the BBE algorithm Gen to generate the hash h of X and output this as part of the digest. The obfuscate algorithm, given as input a circuit E now provides a pPRIO obfuscation of another circuit which garbles E and generates BBE encryptions of its labels using the hash. The BBE ciphertext can be decrypted to recover the appropriate label, using which, the garbled circuit can be executed. In more detail:

1. The setup algorithm compresses the input domain $X = \{X_i \in \{0,1\}^\ell\}_{i \in [N]}$ by hashing it using BBE.Gen into a short hash value h. It outputs $\mathsf{dig} = h$.

2. The obfuscate algorithm $\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)$ outputs a pPRIO obfuscation, pPRIO.Obf, of $C[\mathsf{dig}, E]$.
   The circuit $C[\mathsf{dig}, E]$ on input $i \in [N]$ first computes bGC garbled circuit $\tilde{E}_i$ and input labels $\mathsf{lab}_{i,j,b}$ for $j \in [\ell], b \in \{0,1\}$ – next, it encrypts the bGC labels $(\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1})$ using BBE encryption and finally outputs $\tilde{E}_i$ and BBE ciphertexts $\mathsf{BBE.ct}_{i,j}$. The BBE ciphertexts are encrypted so that one can retrieve the labels for X_i for $\tilde{E}_i$ if a decryptor knows X.

3. The $\mathsf{LEval}(X, \mathsf{Lobf})$ algorithm first runs pPRIO evaluation on pPRIO.Obf and $i \in [N]$ to obtain bGC garbled circuit $\tilde{E}_i$ and BBE ciphertexts $\mathsf{BBE.ct}_{i,j}$ for $j \in [\ell]$. Next it runs BBE decryption using X, index $(i-1)\ell + j$ (this corresponds to j'th bit of X_i) and $\mathsf{BBE.ct}_{i,j}$ to get labels $\{\mathsf{lab}_{i,j}\}_{j \in [\ell]}$ corresponding to X_i. Finally, it runs bGC evaluation using $\tilde{E}_i$ and $\{\mathsf{lab}_{i,j}\}_{j \in [\ell]}$ to obtain $E(X_i)$ for all $i \in [N]$.

Correctness is immediate given the decryption outline above. We then discuss security. Our goal is to show that pPRIO.Obf is pseudorandom assuming $E(X_1), \ldots, E(X_N)$ are pseudorandom. By the security guarantee of pPRIO, it suffices to prove that the truth

table $\{C[\mathsf{dig}, E](i) = (\tilde{E}_i, \{\mathsf{BBE.ct}_{i,j}\}_j)\}_{i\in[N]}$ of the circuit $C[\mathsf{dig}, E]$ is pseudorandom. To show this, we first replace the labels that are *not* corresponding to X_i encrypted inside $\mathsf{BBE.ct}_{i,j}$ with random ones using the security of BBE. Then, by the security of bGC, we replace the garbled circuit $\tilde{E}_i$ and labels corresponding to X_i with the simulated one, computed from $E(X_i)$. We then replace $\{E(X_i)\}_i$ with random strings, which can be done by invoking the assumption of the security definition. Then, by the blindness of bGC, $\tilde{E}_i$ and labels can be replaced with random strings. Finally, by the blindness of BBE, we can replace $\{\mathsf{BBE.ct}_{i,j}\}_j$ with random strings, since they encrypt random strings.

Unfortunately, the above proof strategy does *not* work as described. This is because we assumed in our description above that the output of the pPRIO is pseudorandom, which is required for the security of LprIO to hold. However, unfortunately, this is not the case. Rather, what we have is that the obfuscation of pPRIO can be divided into online and offline parts and that the online part of it is pseudorandom, where the offline part is not dependent on the circuit being obfuscated. Furthermore, the offline part can be reused across invocations. We can show similar property for LprIO, which will be sufficient for the upcoming application of LprIO to PHprFE. We refer the reader to Section 5.4 for details. We will get back to the issue when we discuss the security proof for our PHprFE.

Optimal Parameters at Last.

Using the techniques developed above, we construct a partially hiding compact FE for pseudorandom functionalities, denoted by PHprFE. The syntax of a $\mathsf{PHprFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ scheme is similar to that of a prFE scheme where the input to the encryption scheme now consists of a public part and a secret part. In more detail, the encryption algorithm takes as input $(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ and outputs a ciphertext ct. The ciphertext ct can be decrypted using sk_f and $\mathbf{x}_{\mathsf{pub}}$ to recover $f(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$. The security of the scheme, at a high level, states that if $f(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom then

the ciphertext encoding $(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is indistinguishable from random. We note that a PHprFE scheme implies a prFE scheme if we set $\mathbf{x}_{\mathsf{pub}} = \perp$. We provide a PHprFE construction that achieves optimal parameters and supports circuits of unbounded depth. This in turn will be used to construct optimal kpABE and cpABE for unbounded depth circuits.

Partially Hiding Pseudorandom FE. As discussed above, to get parameters independent of sizes of C and $\mathbf{x}_{\mathsf{pub}}$, we compress them using LprIO scheme. Now, instead of letting the prFE keys directly output labels corresponding to the attribute $\mathbf{x}$ or the garbled circuit corresponding to C, we have it output an obfuscation of circuits that compute these components, as described above. In more detail, we use LprIO, bGC and prFE scheme to construct a PHprFE scheme as follows.

1. The PHprFE setup algorithm generates $(\mathsf{prFE.msk}, \mathsf{prFE.mpk})$ using prFE.Setup and outputs these as msk and mpk respectively.

2. The PHprFE keygen algorithm, given as input a circuit C, does the following.

   - First it uses LprIO scheme to compress C, i.e. compute $\mathsf{dig}_C \leftarrow \mathsf{LDigest}(\{i, C_i\})$ where C_i represents the i^{th} gate of C.

   - Next it samples $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and gives prFE.sk for the circuit $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$. The circuit $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$ on input $(\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}})$ outputs the following :

     - bGC garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$.

     - Obfuscation, $\mathsf{LprIO.Obf}_1$, of a circuit that outputs bGC labels corresponding to $\mathbf{x}_{\mathsf{pub}}$ index by index. We only require short digest of $\mathbf{x}_{\mathsf{pub}}$, $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}$, to compute $\mathsf{LprIO.Obf}_1$.

     - Obfuscation, $\mathsf{LprIO.Obf}_2$, of a circuit that outputs garbled gates of C, gate by gate. We only require short digest of C, dig_C, to compute $\mathsf{LprIO.Obf}_2$.

   - Outputs $\mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk})$.

3. The PHprFE encryptor on input mpk and $\mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ does the following:

- Run $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]})$, where $\mathbf{x}_{\mathsf{pub},i} \in \{0,1\}$ is the i-th bit of $\mathbf{x}_{\mathsf{pub}}$.

- Run $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}))$.

  We note that here the input size of prFE is independent of $|\mathbf{x}_{\mathsf{pub}}|$.

4. The PHprFE decryptor does the following:

   - It first runs the prFE decryption to obtain garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$ and obfuscations $\mathsf{LprIO.Obf}_1$ and $\mathsf{LprIO.Obf}_2$.

   - Next it performs LprIO evaluations using $(\{i, \mathbf{x}_{\mathsf{pub},i}\}_i, \mathsf{LprIO.Obf}_1)$ and $(\{i, C_i\}_i, \mathsf{LprIO.Obf}_2)$ to get the garbled labels corresponding to $\mathbf{x}_{\mathsf{pub}}$ and garbled circuit of C, respectively. Here $\mathbf{x}_{\mathsf{pub},i}$ denotes the i-th bit of $\mathbf{x}_{\mathsf{pub}}$ and C_i denotes the i-th gate of C.

   - Finally, it uses evaluation of the garbling scheme to evaluate the garbled circuit of C on the labels of $\mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$.

Correctness of the above scheme follows from the correctness of the underlying ingredients.

For security we want to show that the $\mathsf{ct} = \mathsf{prFE.ct}(\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom. The adversary additionally sees $\mathsf{mpk}, \mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk})$ and we have the guarantee that $C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom. We prove security via the following three steps.

- First, we invoke prFE security for function $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C]$. By security guarantee of a prFE scheme, it suffices to show pseudorandomness of $\mathrm{F}[\mathbf{r}, \mathsf{dig}_C](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}) = (\mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{LprIO.Obf}_1, \mathsf{LprIO.Obf}_2)$.

- Next, we invoke the security of LprIO scheme using which we wish to show the pseudorandomness of $\mathsf{LprIO.Obf}_1$ and $\mathsf{LprIO.Obf}_2$. Recall that the security of LprIO scheme states that it suffices to show pseudorandomness of circuit's output on the compressed input domain. Thus, it suffices to show the pseudorandomness of $\mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}$ and $\tilde{C}$.
  Now, we are left with showing the pseudorandomness of $\mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}$ and $\tilde{C}$.

- Next, we use bGC simulator to output the garbled values. That is $(\tilde{C}, \mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}) \leftarrow \mathsf{bGC.Sim}(C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. Now using the fact that $C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ is pseudorandom, we invoke the *blindness* of bGC scheme to substitute the simulated value with a random string. Hence the security.

While the above overview for the security proof conveys the main idea, it does not work

as it is, since we do not have the ideal LprIO whose obfuscated circuit is pseudorandom. Rather, we have an imperfect one, where only the online part of the obfuscated circuit is pseudorandom. This leads to a mismatch with the security guarantee of prFE, which requires the decryption result to be pseudorandom. To resolve the above issue, we change the construction so that the offline part of the LprIO obfuscation, which is not necessarily pseudorandom, is generated during the encryption and included into the ciphertext. This change is possible since the offline part of the obfuscation is not dependent on the circuit being obfuscated, which is unknown to the encryptor. Furthermore, the (pseudorandom) online part of the obfuscated circuit is generated during the decryption, which is aligned with the security guarantee of prFE. A subtlety that arises here is that this change requires strong security guarantee for LprIO, where pseudorandomness of the online part should hold even if its offline part is reused across many invocations. Luckily, we can prove such security for LprIO and thus the entire proof works. We refer the reader to Section 5.5 for details.

We reason about efficiency next. First we note that it suffices to use a bounded depth prFE scheme supporting circuits of depth $\mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. Next, we note that the size of prFE input and output is also bounded by $\mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. Thus we get a PHprFE scheme with $|\mathsf{mpk}| = |\mathsf{ct}| = |\mathsf{sk}| = \mathrm{poly}(\lambda, |\mathbf{x}_{\mathsf{priv}}|)$. The sizes of the master public key, ciphertexts, and the secret keys of our construction above are all independent from the length of $\mathbf{x}_{\mathsf{pub}}$ and C. However, they still depend on the length of $\mathbf{x}_{\mathsf{priv}}$. In Section 5.5.4 we show how to remove this dependency from the master public key and the secret keys using a SKE scheme. We get the following theorem.

Theorem 5.11. *Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for circuit class* $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$*, that satisfies reusable security (as per Definition 5.20) whose sizes of the master public key and the secret key are fixed polynomial* $\mathrm{poly}(\lambda)$*. Furthermore, the size of the ciphertext is* $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Optimal kpABE **via** PHprFE**.** Finally we are ready to show how to obtain an optimal kpABE scheme using an optimal PHprFE scheme. The construction is straightforward: we set the attribute as the public input of PHprFE scheme and the message as private input. Additionally we hardwire a nonce in the circuit C during keygen so that it outputs a pseudorandom value when $C(\mathbf{x}) = 0$, using some private input sd. In more detail,

1. The setup generates $(\mathsf{PHprFE.msk}, \mathsf{PHprFE.mpk})$ using PHprFE.Setup and outputs these as msk and mpk respectively.

2. The key generator on input msk and a circuit C does the following

   - Defines circuit $C[\mathbf{r}]$, for $\mathbf{r} \leftarrow \{0,1\}^\lambda$, which on input $(\mathbf{x}, \mu, \mathsf{sd})$ outputs μ if $C(\mathbf{x}) = 1$ and $\mathsf{PRF}(\mathsf{sd}, \mathbf{r})$ otherwise.

   - Computes a PHprFE.sk for circuit $C[\mathbf{r}]$.

   - Outputs $\mathsf{sk}_C \coloneqq (\mathbf{r}, \mathsf{PHprFE.sk})$.

3. The encryptor on input mpk, $\mathbf{x}$, and μ, first samples a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, sets $\mathbf{x}_{\mathsf{pub}} = \mathbf{x}, \mathbf{x}_{\mathsf{priv}} = (\mu, \mathsf{sd})$ and outputs a PHprFE ciphertext $\mathsf{PHprFE.ct} \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.mpk}, \mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$.

4. The decryptor on input the secret key and the ciphertext simply runs the PHprFE decryption.

The correctness of the scheme follows in a straightforward manner from the correctness of the PHprFE scheme. Note that the PHprFE decryption will output $C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mu$ for $C(\mathbf{x}) = 1$. Security of the scheme is implied by the security of the underlying PHprFE scheme and the PRF. To prove security, we need to show the pseudorandomness of $\mathsf{ct} = \mathsf{PHprFE.ct}$ when $C(\mathbf{x}) = 0$ – this follows by our usage of a PRF to generate the output, as discussed above. Finally, from the efficiency of a PHprFE scheme and noting that $|\mathbf{x}_{\mathsf{priv}}| = 1 + \lambda$, we get the following theorem.

Theorem 5.12 (Optimal KP-ABE)**.** *Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-ABE scheme supporting circuits* $\{C : \{0,1\}^\ell \to \{0,1\}\}$ *with unbounded depth and single bit message space with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda).$$

We also obtain a *predicate* encryption scheme for unbounded depth circuits with optimal parameters, where predicate encryption allows to hide the entire input of the ciphertext against restricted adversaries. The optimal kpABE implies an optimal cpABE for unbounded depth circuits as well as optimal kpABE for Turing machines as discussed above. For more details, please see Section 5.6.

Organization of the Chapter We define the preliminaries used in this work in Section 5.2. In Section 5.3 we define and construct the notion of (bounded-depth) compact functional encryption scheme for pseudorandom functionalities (prFE). In Section 5.4 we define the notion of laconic pseudorandom poly-domain obfuscation scheme (LprIO) and construct it using a blind garbling scheme, a blind batch encryption scheme (constructed in Section 5.A) and a pPRIO scheme (constructed in our companion paper [14]). In Section 5.5 we define the notion of partially-hiding prFE (denoted as PHprFE) and construct unbounded-depth PHprFE using a bounded depth prFE and a LprIO. In Section 5.6, we use unbounded-depth PHprFE to construct unbounded-depth kpABE and unbounded-depth PE with optimal parameters. In Section 5.7 we construct unbounded-depth cpABE with optimal parameters using our optimal kpABE and unbounded-depth prFE (as a special case of unbounded-depth PHprFE).

In Section 5.B we show an alternate pathway to construct an unbounded-depth kpABE scheme with sub-optimal parameters *without* using any building block from our companion paper [14] but still improving the state of the art.

5.2 PRELIMINARIES

In this section we introduce some additional preliminaries for this chapter. We refer to Chapter 2 for preliminaries on lattices, Section 4.2.7 for properties of GSW Homomorphic encryption and evaluation and Section 4.2.8 for properties of BGG^+ homomorphic evaluation procedures.

Definition 5.1 (Symmetric Key Encryption with Pseudorandom Ciphertext)**.** A symmetric key encryption scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in[\mathbb{N}]}$ and key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in[\mathbb{N}]}$ and ciphertext space $\mathcal{CT}_{\mathsf{SKE}} = \{\mathcal{CT}_{\mathsf{SKE},\lambda}\}_{\lambda\in[\mathbb{N}]}$ has the following syntax:

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ and outputs a secret key sk.

$\mathsf{Enc}(\mathsf{sk}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the secret key sk and a message $m \in \mathcal{M}_\lambda$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m'$. The decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs a message $m' \in \mathcal{M}_\lambda$.

Correctness: A SKE scheme is said to be correct if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\Pr\left[\begin{array}{rl} & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ m' = m : & \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m); \\ & m' = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}). \end{array}\right] \geq 1 - \mathsf{negl}(\lambda), \tag{5.5}$$

Security: A SKE scheme is said to have pseudorandom ciphertext if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for every message $m \in \mathcal{M}_\lambda$, we have

$$\left|\Pr\left[\begin{array}{rl} & \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda); \\ \beta' = \beta : & \\ & \beta' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk},\cdot),\mathsf{Enc}^\beta(\mathsf{sk},\cdot)}. \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda), \tag{5.6}$$

where the $\mathsf{Enc}(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns $\mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{Enc}^\beta(\mathsf{sk}, \cdot)$ oracle, on input a message m, returns ct_β, where $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.

5.2.1 Blind Garbled Circuit

Here we provide the definition of a garbling scheme for circuit class $C = \{C : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\}$. A garbling scheme for circuit class C consists of three

algorithms $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ with the following syntax.

$\mathsf{Garble}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C) \to (\mathsf{lab}, \tilde{C})$. The garbling algorithm takes as input the security parameter λ, the input length ℓ_{in} and output length ℓ_{out} for circuit C, the description of the circuit C, and a random value $\mathsf{st} \in \{0,1\}^\lambda$ and outputs the labels for input wire of the garbled circuit $\mathsf{lab} = \{\mathsf{lab}_{j,b}\}_{j\in[\ell_{\mathsf{in}}],b\in\{0,1\}}$ where each $\mathsf{lab}_{j,b} \in \{0,1\}^\lambda$ and the garbled circuit $\tilde{C}$.

$\mathsf{Eval}(1^\lambda, \tilde{C}, \mathsf{lab}_{\mathbf{x}}) \to \mathbf{y}$. The evaluation algorithm takes as input the garbled circuit $\tilde{C}$ and labels corresponding to an input $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$, $\mathsf{lab}_{\mathbf{x}} = \{\mathsf{lab}_{i,x_i}\}_{i\in[\ell_{\mathsf{in}}]}$ where x_i denotes the i-th bit of $\mathbf{x}$, and it outputs $\mathbf{y} \in \{0,1\}^{\ell_{\mathsf{out}}}$.

$\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, \mathbf{y})$ is a PPT algorithm that takes as input the security parameter, the description length of C, an input length ℓ_{in} and a string $\mathbf{y} \in \{0,1\}^{\ell_{\mathsf{out}}}$, and outputs a simulated garbled circuit $\bar{C}$ and labels $\bar{\mathsf{lab}}$.

A garbling scheme satisfies the following properties.

Definition 5.2 (Correctness)**.** A garbling scheme is said to be correct if for any circuit $C \in \mathcal{C}$ and any input $x \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

$$\Pr\left[\, \mathbf{y} = C(\mathbf{x}) : (\mathsf{lab}, \tilde{C}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C); \mathbf{y} \leftarrow \mathsf{Eval}(\tilde{C}, \mathsf{lab}_{\mathbf{x}}) \right] = 1.$$

Definition 5.3 (Simulation Security)**.** A garbling scheme is said to satisfy simulation security if for any circuit $C \in \mathcal{C}$ and any input $\mathbf{x} \in \{0,1\}^{\ell_{\mathsf{in}}}$, the following holds

$$\{(\tilde{C}, \mathsf{lab}_{\mathbf{x}}) \mid (\mathsf{lab}, \tilde{C}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C)\} \approx_c \{(\bar{C}, \bar{\mathsf{lab}}) \mid (\bar{C}, \bar{\mathsf{lab}}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, C(\mathbf{x}))\}$$

where $\mathsf{lab} = \{\mathsf{lab}_{j,b}\}_{j\in[\ell_{\mathsf{in}}],b\in\{0,1\}}$ and $\mathsf{lab}_{\mathbf{x}} = \{\mathsf{lab}_{i,x_i}\}_{i\in[\ell_{\mathsf{in}}]}$.

Definition 5.4 (Blindness)**.** [58] A garbling scheme $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{Sim})$ is called blind if the distribution $\mathsf{Sim}(1^\lambda, 1^{|C|}, 1^{\ell_{\mathsf{in}}}, \mathbf{y})$ for $\mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, representing the output of the simulator on a completely uniform output, is indistinguishable from a completely uniform bit string. (Note that the distinguisher must not know the random output value that was

used for the simulation.)

Definition 5.5 (Decomposability)**.** We note that the $\mathsf{Garble}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, C)$ algorithm can be decomposed, using shared randomness st, as follows : (i) $\mathsf{Garble}_i(1^\lambda, C_i; \mathsf{st})$ for $i \in [|C|]$, where $\mathsf{Garble}_i(1^\lambda, C_i)$ outputs the garbling of i-th gate of the circuit C (denoted by C_i) and (ii) $\mathsf{Garble}_{\mathsf{inp}}(1^\lambda, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}; \mathsf{st}) = \mathsf{lab}$ which outputs $2 \cdot \ell_{\mathsf{in}}$ labels. Note that information of a single gate C_i of C can be represented by a binary string of length at most 4λ for example, since it suffices to encode its index, indices of its two incoming wires, and the truth table of the gate.

Theorem 5.13. *[58] Assume that one-way function exists. Then, there exists a blind garbled circuits scheme.*

5.2.2 Blind Batch Encryption

Here we provide the definition of a batch encryption scheme largely adapted from [58]. A batch encryption scheme with the message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ consists of the following algorithms.

$\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ and key length N, and outputs a common reference string crs.

$\mathsf{Gen}(\mathsf{crs}, \mathbf{x}) \to h$. The generation algorithm takes as input the common reference string crs and a secret key $\mathbf{x} \in \{0,1\}^N$. It outputs a public key h.

$\mathsf{SingleEnc}(\mathsf{crs}, h, i, (m_0, m_1)) \to \mathsf{ct}$. The encryption algorithm takes as input a common reference string crs, the public key h, an index $i \in [N]$, and a message $(m_0, m_1) \in \mathcal{M}^2$ and outputs a (single) ciphertext ct.

$\mathsf{SingleDec}(\mathsf{crs}, \mathbf{x}, i, \mathsf{ct}) \to m$. The encryption algorithm takes as input a common reference string crs, the secret key $\mathbf{x}$, an index $i \in [N]$, and a (single) ciphertext ct and outputs a message $m \in \mathcal{M}$.

In [58], they define additional algorithms Enc and Dec, which can be defined using $\mathsf{SingleEnc}$ and $\mathsf{SingleDec}$ above. We omit the definition of these algorithms since they are not used in our paper.

Definition 5.6 (Correctness.)**.** A batch encryption scheme is said to be correct if for any $\lambda, N \in \mathbb{N}$, secret key $\mathbf{x} \in \{0,1\}^N$, $i \in [N]$, $(m_0, m_1) \in \mathcal{M}^2$, $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, and $h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x})$ it holds that

$$\mathsf{Pr}[m = m_{x_i} \mid m = \mathsf{SingleDec}\,(\mathsf{crs}, \mathbf{x}, i, \mathsf{SingleEnc}(\mathsf{crs}, h, i, (m_0, m_1)))] = 1,$$

where x_i is the i-th bit of $\mathbf{x}$.

Definition 5.7 (Succinctness)**.** A batch encryption scheme is α-succinct if letting $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, $h = \mathsf{Gen}(\mathsf{crs}, \mathbf{x})$ for some $\mathbf{x} \in \{0,1\}^N$, it holds that $|h| \leq \alpha N$. It is said fully succinct if $|h| \leq p(\lambda)$ for some fixed polynomial $p(\lambda)$.

Definition 5.8 (Security)**.** A batch encryption scheme is said to be secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds

$$\mathsf{Pr}\left[\beta' = \beta : \begin{array}{l} (1^N, \mathbf{x} \in \{0,1\}^N, i \in [N]) \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N); \\ \left(\mathbf{m}^{(0)} = (m_0^{(0)}, m_1^{(0)}), \mathbf{m}^{(1)} = (m_0^{(1)}, m_1^{(1)})\right) \leftarrow \mathcal{A}(\mathsf{crs}); \\ h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x}), \beta \leftarrow \{0,1\}, \mathsf{ct}_\beta \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}^{(\beta)}); \\ \beta' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where we require that $\mathbf{m}^{(0)}, \mathbf{m}^{(1)} \in \mathcal{M}^2$ and $m_{x_i}^{(0)} = m_{x_i}^{(1)}$.

We require somewhat stronger blindness condition than that defined in [58].

Definition 5.9 (Strong Blindness)**.** A batch encryption scheme is said to satisfy blindness if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the

following holds

$$\mathsf{Pr}\left[\beta' = \beta : \begin{array}{l} (1^N, \mathbf{x} \in \{0,1\}^N, i \in [N]) \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N), h \leftarrow \mathsf{Gen}(\mathsf{crs}, \mathbf{x}); \\ \mathbf{m} \leftarrow \mathcal{M}^2, \beta \leftarrow \{0,1\}; \\ \mathsf{ct}_0 \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h, i, \mathbf{m}), \mathsf{ct}_1 \leftarrow \mathcal{CT}; \\ \beta' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{ct}_\beta) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{CT}$ is the ciphertext space of the scheme.

Remark 17. Here, we compare strong blindness defined above with the blindness defined in [58]. In [58], they divide a ciphertext ct into an "offline part" subct_1 and "online part" subct_2, where subct_1 only depends on the encryption randomness and the CRS, whereas subct_2 may depend on h, i, and $\mathbf{m}$ additionally. They then define blindness as the security notion that essentially requires that subct_2 is pseudorandom, whereas subct_1 may not be. The above security notion is stronger than theirs in that we require the entire ciphertext to be pseudorandom rather than part of it. In other words, we can see our definition as more stringent version of their blindness notion where we require subct_1 to be an empty string.

5.2.3 Poly-Input Obfuscation for Pseudorandom Functionalities

In this section we give the definition of poly-input indistinguishability obfuscation for pseudorandom functionalities (pPRIO), adapted from [14].

Syntax A pPRIO scheme consists of the following algorithms.

$\mathsf{Obf}(1^\lambda, C) \to \mathsf{Obf}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : [N] \to [M]$ with $\mathsf{size}(C) \leq L$ for some arbitrary polynomial $L = L(\lambda)$. It outputs an obfuscation of the circuit Obf.

We consider a definition where the Obf algorithm can be decomposed into the following two phases.

$\mathsf{ObfOff}(1^\lambda, 1^L) \to (\mathsf{obf}_{\mathsf{off}}, \mathsf{st})$. The offline obfuscation algorithm takes as input the security parameter λ and the circuit size bound L. It outputs $\mathsf{obf}_{\mathsf{off}}$ and st.

$\mathsf{ObfOn}(\mathsf{st}, C) \to \mathsf{obf}_{\mathsf{on}}$. The online obfuscation algorithm takes as input the security parameter st and the circuit C and outputs $\mathsf{obf}_{\mathsf{on}}$.

The final output of Obf is $\mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}})$.

$\mathsf{Eval}(\mathsf{Obf}, x) \to y$. The evaluation algorithm takes as input an obfuscated circuit Obf and an input $x \in [N]$. It outputs $y \in [M]$.

Next, we define the properties of a pPRIO scheme.

Definition 5.10 (Correctness)**.** For all security parameters $\lambda \in \mathbb{N}$, for any $C : [N] \to [M]$, $L = L(\lambda)$ such that $\mathsf{size}(C) \leq L$ and every input $x \in [N]$, we have that:

$$\mathsf{Pr}\left[\mathsf{Eval}(\mathsf{Obf}, x) = C(x) \mid \mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}),\ (\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L), \mathsf{obf}_{\mathsf{on}} \leftarrow \mathsf{ObfOn}(\mathsf{st}, C)\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator Obf.

Definition 5.11 (Security)**.** Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(1^{N_1+N_2+\cdots N_Q}, 1^L, \mathsf{aux}, C^1, \ldots, C^Q\right), \quad \text{where} \quad C^i : [N_i] \to [M_i], \ \ \mathsf{size}(C^i) \leq L$$

where we enforce Samp to output $1^{N_1+N_2+\cdots N_Q}$ to make sure that all N_i are bounded by $\mathrm{poly}(\lambda)$. We say that a pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\text{If } \left(\mathsf{aux}, \{C^1(i)\}_{i\in[N_1]}, \ldots, \{C^Q(i)\}_{i\in[N_Q]}\right) \approx_c \left(\mathsf{aux}, \{\Delta_i^1\}_{i\in[N_1]}, \ldots, \{\Delta_i^Q\}_{i\in[N_Q]}\right) \tag{5.7}$$

$$\text{then} \quad \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}^1, \ldots, \mathsf{obf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \delta^1 \leftarrow \mathcal{CT}^1 \ldots, \delta^Q \leftarrow \mathcal{CT}^Q\right), \tag{5.8}$$

where $\Delta_i^j \leftarrow [M_j]$ for $j \in [Q], i \in [N_j]$, $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L)$, $\mathsf{obf}_{\mathsf{on}}^j \leftarrow \mathsf{ObfOn}(\mathsf{st}, C^j)$ for $j \in [Q]$, and $\mathcal{CT}^j$ denotes the set of binary strings of the same length as the output of $\mathsf{obf}_{\mathsf{on}}(\mathsf{st}, C^j)$ algorithm.

Remark 18. It is shown in [19, 65] that there is no pPRIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of pPRIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was pPRIO that is secure for all the samplers.

Theorem 5.14 ([14])**.** *Assuming LWE and evasive LWE assumptions, there exists a secure* pPRIO *scheme satisfying* $|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda)$, $|\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda)$ *where* $(\mathsf{Obf}_{\mathsf{off}}, \mathsf{Obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ *for circuit* $C : [N] \to [M]$ *whose size is bounded by* $L = L(\lambda)$.

5.3 FUNCTIONAL ENCRYPTION FOR PSEUDORANDOM FUNCTIONALITIES

5.3.1 Definition

In this section we give the definitions for functional encryption for pseudorandom functionalities. Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$ for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input a string $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x) \in \mathcal{Y}_{\mathsf{prm}}$.

Syntax. A functional encryption scheme prFE for pseudorandom functionalities $\mathcal{F}_{\mathsf{prm}}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk[4].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master

[4]We assume w.l.o.g that msk includes mpk.

secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk and an input $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{CT}$, where $\mathcal{CT}$ is the ciphertext space.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}) \to y$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and a ciphertext ct and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 5.12 (Correctness)**.** A prFE scheme is said to satisfy *perfect* correctness if for all prm, any input $x \in \mathcal{X}_{\mathsf{prm}}$ and function $f \in \mathcal{F}_{\mathsf{prm}}$, we have

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}) \ , \ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{Dec}\big(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{Enc}(\mathsf{mpk}, x)\big) = f(x)\end{array}\right] = 1.$$

We define our security notion next. At a high level, our notion says that so long as the output of the functionality is pseudorandom, the ciphertext is pseudorandom. For notational brevity, we denote this by prCT security.

Definition 5.13 (prCT Security)**.** For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries, and $f_i \in \mathcal{F}_{\mathsf{prm}}, x_j \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.

We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} &\mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{f_i,\ f_i(x_j)\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\right) = 1] \\ &- \mathsf{Pr}[\mathcal{A}_0(\ \mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1]\end{aligned} \tag{5.9}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1]$$
$$- \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}], j\in[Q_{\mathsf{msg}}]}) = 1] \tag{5.10}$$

where $(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathrm{Samp}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ satisfies prCT security with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ there exists a polynomial $Q(\cdot)$ such that for every PPT adversary $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{5.11}$$

and $\mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda)$.

Remark 19. It is shown in [19] that there is no prFE that satisfies prCT security for all general samplers. Therefore, when we use the security of prFE, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was prFE that is secure for all the samplers.

Remark 20. We note that the above security definition is in the multi-challenge flavor. One may wonder whether single-challenge version of the definition where $Q_{\mathsf{msg}} = 1$ implies the multi-challenge version or not. However, unlike the standard security notions for public key primitives (e.g., indistinguishability security for functional encryption [91]), this does not seem to be the case. This is because the standard hybrid argument to prove the multi-challenge security from the single challenge security where we replace the ciphertext to be random one-by-one fails. To see why, recall that in these hybrids, we simulate some of the ciphertexts honestly, while we try to change a particular honest ciphertext to be a random one. To generate honest ciphertexts, we may have to know the corresponding plaintexts. However, this may ruin the precondition for invoking the single challenge security for the target ciphertext, since knowing some of the inputs (say, x_1) may make the output (say, $f_1(x_2)$) not pseudorandom any more when the inputs are

correlated with each other.

Definition 5.14 (Compactness)**.** A prFE scheme is said to be compact if for any input message $x \in \mathcal{X}$, the running time of the encryption algorithm is polynomial in the security parameter and the size of x. In particular, it does not depend on the circuit description size or the output length of any function f supported by the scheme.

5.3.2 Construction

In this section, we provide our construction of a functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{\mathrm{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}\}$, where the depth of a function $f \in \mathcal{F}$ is at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$. We denote the information of the parameters representing the supported class of the circuits by $\mathsf{prm} = (1^{\mathrm{L}(\lambda)}, 1^{\ell(\lambda)}, 1^{\mathsf{dep}(\lambda)})$.

Ingredients. Our construction needs a pseudorandom function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to [-q/4 + B, q/4 - B]^{1\times\ell}$ that can be evaluated by a circuit of depth at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$. Here B is chosen to be exponentially smaller than $q/4$. We note that for our choice of B the statistical distance between the uniform distribution over $[-q/4, q/4]$ and $[-q/4 + B, q/4 - B]$ is negligible.

$\mathsf{Setup}(1^{\lambda}, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm parses $\mathsf{prm} = (1^{\mathrm{L}(\lambda)}, 1^{\ell(\lambda)}, 1^{\mathsf{dep}(\lambda)})$ and does the following.

- Sample appropriate parameters $q, n, m, \tau, \sigma, \sigma_{\mathbf{B}}, B$ and M such that M divides q, as in Equation (5.12)[5].
- Samples appropriate parameters as in Equation (5.12).
- Set $\mathrm{L}_{\mathrm{X}} = m(\lambda + \mathrm{L})(n+1)\lceil \log q \rceil$, sample $\mathbf{A}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{X}}+1)m}$ and $(\mathbf{B}, \mathbf{B}_{\tau}^{-1}) \leftarrow \mathsf{TrapGen}(1^{n+1}, 1^{mw}, q)$, where $w \in O(\log q)$.
- Output $\mathsf{mpk} := (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$ and $\mathsf{msk} := \mathbf{B}_{\tau}^{-1}$.

[5] We assume these parameters to be part of the mpk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm parses $\mathsf{msk} = \mathbf{B}_\tau^{-1}$ and does the following.

- Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and define function $\mathrm{F} = \mathrm{F}[f, \mathbf{r}]$ with $f, \mathbf{r}$ hardwired as follows[6]:
  On input $(\mathbf{x}, \mathsf{sd})$, compute and output $f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) \in \mathbb{Z}_q^{1\times\ell}$.

- Parse $\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, where $f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) \in [0, q/M]^\ell$ and $f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \in [0, M-1]^\ell$.Using the fact that the PRF and $f(\mathbf{x})$ can be computed by a circuit of depth at most $\mathsf{dep}(\lambda) = \mathrm{poly}(\lambda)$, the function $\mathrm{F}[f, \mathbf{r}]$ can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.

- Define functions $\mathrm{F}_{\mathsf{high}} := M \cdot f_{\mathsf{high}}$ and $\mathrm{F}_{\mathsf{low}} := M \cdot f_{\mathsf{low}}$, which on input $(\mathbf{x}, \mathsf{sd})$ outputs $M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd})$ and $M \cdot f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd})$, respectively. We note that these functions can be computed by a circuit of depth at most $d = \mathrm{poly}(\mathsf{dep})$.

- Define $\mathsf{VEval}_{\mathsf{high}} = \mathrm{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{high}})$ and $\mathsf{VEval}_{\mathsf{low}} = \mathrm{MakeVEvalCkt}(n, m, q, \mathrm{F}_{\mathsf{low}})$. From Theorem 4.5, the depth of $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ is bounded by $(dO(\log m \log\log q) + O(\log^2 \log q))$.

- Compute $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{high}})$, $\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}} = \mathsf{MEvalC}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{low}}) \in \mathbb{Z}_q^{(\mathrm{L_X}+1)m\times\ell}$.

- Compute $\mathbf{A}_{\mathsf{high}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}}$ and $\mathbf{A}_{\mathsf{low}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{low}}}$.

- Compute
$$\mathbf{A}_{\mathrm{F}} = M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rceil$$
and sample $\mathbf{K} \leftarrow \mathbf{B}_\tau^{-1}(\mathbf{A}_{\mathrm{F}})$.

- Output $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}) \to \mathsf{ct}$. The encryption parse $\mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$ algorithm does the following.

- Sample $\bar{\mathbf{s}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{s}}}^n$ and set $\mathbf{s} = (\bar{\mathbf{s}}^\intercal, -1)^\intercal$.

- Sample $\mathbf{e}_{\mathbf{B}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_{\mathbf{B}}}^{mw}$ and compute $\mathbf{c}_{\mathbf{B}}^\intercal := \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal$.

[6]The circuit representation of the function F is the universal circuit with F hardwired.

– Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, $\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, $\mathbf{R} \leftarrow \{0,1\}^{m\times m(\lambda+\mathrm{L})}$ and compute a GSW encryption as follows.

$$\mathbf{A}_{\mathsf{fhe}} \coloneqq \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix}, \quad \mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{sd}) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}.$$

Let $\mathrm{L}_X = m(\lambda+\mathrm{L})(n+1)\lceil \log q \rceil$ be the bit length of $\mathbf{X}$.

– Compute a BGG^+ encoding as follows.

$$\mathbf{e}_{\mathsf{att}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(\mathrm{L}_X+1)m}, \qquad \mathbf{c}_{\mathsf{att}}^\intercal \coloneqq \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^\intercal.$$

– Output $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}) \to \mathbf{y}$. The decryption algorithm does the following.

– Parse $\mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M)$, $\mathsf{sk}_f = (\mathbf{K}, \mathbf{r})$ and $\mathsf{ct} = (\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X})$.

– Compute $\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{high}}, \mathbf{X})$ and $\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}} = \mathsf{MEvalCX}(\mathbf{A}_{\mathsf{att}}, \mathsf{VEval}_{\mathsf{low}}, \mathbf{X})$ for circuits $\mathsf{VEval}_{\mathsf{high}}$ and $\mathsf{VEval}_{\mathsf{low}}$ as defined in KeyGen algorithm.

– Compute

$$\mathbf{z} \coloneqq \mathbf{c}_{\mathbf{B}}^\intercal \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^\intercal \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{low}}}}{M} \right\rceil \right).$$

– For $i \in [\ell]$, set $y_i = 0$ if $z_i \in [-q/4, q/4)$ and $y_i = 1$ otherwise, where z_i is the i-th coordinate of $\mathbf{z}$.

– Output $\mathbf{y} = (y_1, \ldots, y_\ell)$.

Parameters. We set our parameters as follows.

$$\beta = 2^{O(\mathsf{dep}\cdot\log\lambda)},\ q = 2^{12\lambda}\beta,\ M = 2^{4\lambda}\beta,\ n = \mathrm{poly}(\lambda, \mathsf{dep}),\ m = O(n\log q),\ \ B = 2^{10\lambda}\beta,$$
$$\tau = O\left(\sqrt{(n+1)\log q}\right)\ \ \sigma_{\mathbf{s}} = \sigma = 2^{2\lambda},\ \ \sigma_{\mathbf{B}} = 2^{9\lambda}\beta,\ \ \sigma_1 = 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda). \tag{5.12}$$

Efficiency. Using the above set parameters, we have

$$|\mathsf{mpk}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = \mathrm{L} \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

Correctness We analyze the correctness of our scheme below.

– First, we note that

$$\begin{aligned}
\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} &= (\mathbf{s}^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}})\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{att}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}}}^{\mathrm{F}_{\mathsf{high}}} - \mathbf{s}^{\mathsf{T}}\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
&= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - \mathrm{F}_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}} \\
\implies \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} - M \cdot f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil \\
&= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \quad (5.13)
\end{aligned}$$

where $\mathsf{VEval}_{\mathsf{high}}(\mathsf{bits}(\mathbf{X})) = \mathbf{A}_{\mathsf{fhe}}\mathbf{R}_{\mathsf{high}} - \begin{pmatrix} \mathbf{0}_{n\times\ell} \\ \mathrm{F}_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) \end{pmatrix}$. Using Theorem 4.5, we have

$$\begin{aligned}
\left\|\mathbf{R}_{\mathsf{high}}^{\mathsf{T}}\right\| &\le (m+2)^d \lceil \log q \rceil \cdot m = (m+2)^d \lceil \log q \rceil \cdot 3(n+1)\lceil \log q \rceil \\
&\le (m+2)^d O(\log q) \le \beta.
\end{aligned}$$

and using the depth bound from Section 4.2.8,

$$\left\|\left(\mathrm{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right)^{\mathsf{T}}\right\| \le (m+2)^{d_{\mathsf{VEval}_{\mathsf{high}}}} \lceil \log q \rceil \le 2^{d \cdot O(\log \lambda)} \le \beta$$

where $d_{\mathsf{VEval}_{\mathsf{high}}}$ denotes the depth of the circuit $\mathsf{VEval}_{\mathsf{high}}$. So we have $\left\|\mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \le 2^{2\lambda+1}\sqrt{\lambda}\beta \le 2^{3\lambda}\beta < M$. Using this in Equation (5.13),

$$\begin{aligned}
\left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}}{M} \right\rceil &= \left\lfloor \frac{\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}}}{M} \right\rceil - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{err}_{\mathsf{high}} \\
&= \mathbf{s}^{\mathsf{T}}\left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rceil + \mathbf{e}_{\mathbf{s},\mathsf{high}}^{\mathsf{T}} - f_{\mathsf{high}}(\mathbf{x},\mathsf{sd}) + \mathbf{err}_{\mathsf{high}}^{\mathsf{T}} \qquad (5.14)
\end{aligned}$$

where $\mathbf{err}_{\mathsf{high}}^{\mathsf{T}} \in \{0,1\}^{\ell}$, is the rounding error which is 1 if $\left\|(\mathbf{s}^{\mathsf{T}}\mathbf{A}_{\mathsf{high}})^{\mathsf{T}} + \mathbf{e}_{\mathsf{fhe}}^{\mathsf{T}}\mathbf{R}_{\mathsf{high}} + \mathbf{e}_{\mathsf{att}}^{\mathsf{T}}\mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathrm{F}_{\mathsf{high}}}\right\| \ge M$ and 0 otherwise, and $\left\|\mathbf{e}_{\mathbf{s},\mathsf{high}}\right\| \le (n+1) \cdot \|\mathbf{s}\|$. To see the latter, we use the fact that

$\lfloor \mathbf{s}^\intercal X \rfloor - \mathbf{s}^\intercal \lfloor X \rfloor = \lfloor \mathbf{s}^\intercal X - \mathbf{s}^\intercal \lfloor X \rfloor \rfloor = \lfloor \mathbf{s}^\intercal (X - \lfloor X \rfloor) \rfloor$, where $X - \lfloor X \rfloor < 1$. So $\mathbf{e}^\intercal_{\mathbf{s},\mathsf{high}} = \left\lfloor \mathbf{s}^\intercal \left(\frac{\mathbf{A}_{\mathsf{high}}}{M} - \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor \right) \right\rfloor$ and $\left\| \mathbf{e}_{\mathbf{s},\mathsf{high}} \right\| \leq \|\mathbf{s}\| \left\| \left(\frac{\mathbf{A}_{\mathsf{high}}}{M} - \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor \right)^\intercal \right\| < (n+1) \|\mathbf{s}\|$.

Using a similar analysis as to obtain Equation (5.14), we get

$$\left\lfloor \frac{\mathbf{c}^\intercal_{\mathsf{att}} \cdot \mathbf{H}^{\mathsf{F}_{\mathsf{low}}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}}{M} \right\rfloor = \mathbf{s}^\intercal \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor + \mathbf{e}^\intercal_{\mathbf{s},\mathsf{low}} - f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) + \mathbf{err}^\intercal_{\mathsf{low}} \tag{5.15}$$

where $\mathbf{err}^\intercal_{\mathsf{low}} \in \{0,1\}^\ell$ and $\left\| \mathbf{e}_{\mathbf{s},\mathsf{low}} \right\| \leq (n+1) \cdot \|\mathbf{s}\|$. Using Equation (5.14) and Equation (5.15), we get

$$\begin{aligned}
& M \cdot \left\lfloor \frac{\mathbf{c}^\intercal_{\mathsf{att}} \cdot \mathbf{H}^{\mathsf{F}_{\mathsf{high}}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}^\intercal_{\mathsf{att}} \cdot \mathbf{H}^{\mathsf{F}_{\mathsf{low}}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}}{M} \right\rfloor \\
&= \mathbf{s}^\intercal \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor \right) - \left(M \cdot f_{\mathsf{high}}(\mathbf{x}, \mathsf{sd}) + f_{\mathsf{low}}(\mathbf{x}, \mathsf{sd}) \right) + \mathbf{err} \\
&= \mathbf{s}^\intercal \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor \right) - \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) + \mathbf{err}
\end{aligned} \tag{5.16}$$

where

$$\begin{aligned}
\mathbf{err} &= M \cdot \mathbf{e}^\intercal_{\mathbf{s},\mathsf{high}} + \mathbf{e}^\intercal_{\mathbf{s},\mathsf{low}} + M \cdot \mathbf{err}_{\mathsf{high}} + \mathbf{err}_{\mathsf{low}} \\
&= M \cdot \left(\mathbf{s}^\intercal \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor - \left\lfloor \mathbf{s}^\intercal \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor \right) + \left(\mathbf{s}^\intercal \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor - \left\lfloor \mathbf{s}^\intercal \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor \right) + M \cdot \mathbf{err}_{\mathsf{high}} + \mathbf{err}_{\mathsf{low}}
\end{aligned} \tag{5.17}$$

where $\mathbf{err}_{\mathsf{high}}, \mathbf{err}_{\mathsf{low}} \in \{0,1\}^\ell$ are rounding errors and matrices $\mathbf{A}_{\mathsf{high}}, \mathbf{A}_{\mathsf{low}}$ are publicly computable matrices and

$$\begin{aligned}
\|\mathbf{err}\| \leq & M \cdot ((n+1) \cdot \|\mathbf{s}\| + 1) + (n+1) \cdot \|\mathbf{s}\| + 1 \leq 2M \cdot ((n+1) \cdot \|\mathbf{s}\| + 1) \\
= & 2^{4\lambda+1} \beta \left((n+1) \cdot 2^{2\lambda+1} \sqrt{\lambda} \right) \leq 2^{7\lambda} \beta
\end{aligned}$$

– Next, we note that

$$\mathbf{c}^\intercal_{\mathbf{B}} \cdot \mathbf{K} = \mathbf{s}^\intercal \left(M \cdot \left\lfloor \frac{\mathbf{A}_{\mathsf{high}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{A}_{\mathsf{low}}}{M} \right\rfloor \right) + \mathbf{c}^\intercal_{\mathbf{B}} \cdot \mathbf{K}. \tag{5.18}$$

where $\left\| (\mathbf{c}^\intercal_{\mathbf{B}} \cdot \mathbf{K})^\intercal \right\| \leq 2^{9\lambda} \beta \sqrt{\lambda} \cdot \tau$ from our parameter setting.

– Using Equations (5.16) and (5.18), we get

$$\begin{aligned}\mathbf{z} &= \mathbf{c}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \left(M \cdot \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathsf{F}_{\mathsf{high}}}}{M} \right\rfloor + \left\lfloor \frac{\mathbf{c}_{\mathsf{att}}^{\mathsf{T}} \cdot \mathbf{H}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}^{\mathsf{F}_{\mathsf{low}}}}{M} \right\rfloor \right) \\ &= \mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{err} \\ &= f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{err}\end{aligned}$$

where

$$\begin{aligned}\left\| \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) + \mathbf{e}_{\mathbf{B}}^{\mathsf{T}} \cdot \mathbf{K} - \mathbf{err} \right\| &\leq \|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})\| + 2^{9\lambda} \beta \sqrt{\lambda} \cdot \tau + 2^{7\lambda} \beta \\ &\leq \|\mathsf{PRF}(\mathsf{sd}, \mathbf{r})\| + 2^{9\lambda+1} \beta \sqrt{\lambda} \cdot \tau < q/4 - B + B < q/4\end{aligned}$$

Hence the last step of decryption outputs $\mathbf{y}$ correctly with probability 1.

5.3.3 Security Proof for Pseudorandom Functionalities

Theorem 5.15. *Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for* prFE*. Assuming LWE (Assumption 2.5) and private coin Evasive LWE (Assumption 2.6) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{evs}}(1^\lambda)$ induced by $\mathrm{Samp}_{\mathsf{prFE}} \in \mathcal{SC}_{\mathsf{prFE}}$ as defined in Figure 5.1, our* prFE *scheme satisfies* prCT *security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 5.13.*

Proof. Consider a sampler $\mathrm{Samp}_{\mathsf{prFE}}$ that generates the following:

1. **Key Queries.** It issues Q_{key} number of functions $f_1, \ldots, f_{Q_{\mathsf{key}}}$ for key queries.
2. **Ciphertext Queries.** It issues Q_{msg} ciphertext queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the prCT security as per Definition 5.13, we show

$$\left(\begin{array}{r} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} = \mathbf{SB} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^{\mathsf{T}} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^{\mathsf{T}}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}]} \end{array}\right) \approx_c \left(\begin{array}{r} \mathsf{mpk} = (\mathbf{A}_{\mathsf{att}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\ \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}} \times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathbf{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}]} \end{array}\right) \tag{5.19}$$

where

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1^\mathsf{T} \\ \vdots \\ \mathbf{s}_{Q_{\mathsf{msg}}}^\mathsf{T} \end{pmatrix}, \ \mathbf{E}_\mathbf{B} = \begin{pmatrix} \mathbf{e}_{\mathbf{B},1}^\mathsf{T} \\ \vdots \\ \mathbf{e}_{\mathbf{B},Q_{\mathsf{msg}}}^\mathsf{T} \end{pmatrix},$$

$$(\mathsf{aux}_\mathcal{A}, \{f_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}_j\}_{j\in[Q_{\mathsf{msg}}]}) \leftarrow \mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$$

and for $j \in [Q_{\mathsf{msg}}], \mathbf{s}_j, \mathbf{e}_{\mathbf{B},j}, \mathbf{A}_{\mathsf{fhe},j}, \mathbf{R}_j, \mathsf{sd}_j, \mathbf{e}_{\mathsf{att},j}$ are sampled as in the construction, for $k \in [Q_{\mathsf{key}}]$, we have $\mathbf{r}_k \leftarrow \{0,1\}^\lambda$, $\mathrm{F}_k = \mathrm{F}[f_k, \mathbf{r}_k]$ and $\mathbf{A}_{\mathrm{F}_k}$ is as defined in the construction, and $\mathbf{K}_k = \mathbf{B}_\tau^{-1}(\mathbf{A}_{\mathrm{F}_k})$

assuming we have

$$(1^\lambda,\ \mathsf{aux}_\mathcal{A},\ \{f_k,\ f_k(x_j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_\mathcal{A},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}). \tag{5.20}$$

We invoke evasive LWE assumption for a matrix $\mathbf{B}$ with the private coin sampler $\mathsf{Samp}_{\mathsf{evs}}$ that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2))$ with private coin $\mathsf{coins}_{\mathsf{priv}}^{\mathsf{Samp}_{\mathsf{evs}}} = \{\mathsf{sd}_j, \mathbf{R}_j, \mathbf{e}_{\mathsf{att},j}, \mathbf{A}_{\mathsf{fhe},j}\}_{j\in[Q_{\mathsf{msg}}]}$, defined as follows.

By Theorem 2.7, to prove Equation (5.19) assuming evasive LWE, it suffices to show

$$\begin{pmatrix} \mathsf{aux}_2,\ \ \mathbf{B},\ \ \mathbf{C}_\mathbf{B} = \mathbf{S}\mathbf{B} + \mathbf{E}_\mathbf{B}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j}^\mathsf{T} = \mathbf{s}_j^\mathsf{T}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{att},j}^\mathsf{T}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_\mathbf{P} = \mathbf{S}\mathbf{P} + \mathbf{E}_\mathbf{P} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_2,\ \ \mathbf{B},\ \ \mathbf{C}_\mathbf{B} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathrm{X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_\mathbf{P} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}} \end{pmatrix} \tag{5.21}$$

$\mathsf{Samp}_{\mathsf{evs}}(1^\lambda)$

The sampler does the following.

- Runs the prFE sampler $\mathsf{Samp}_{\mathsf{prFE}}$ to obtain $(\{f_k\}_{k\in Q_{\mathsf{key}}}, \{\mathbf{x}_j\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux}_{\mathcal{A}})$ where $f_k : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}$, $\mathbf{x}_j \in \{0,1\}^{\mathrm{L}}$ and $\mathsf{aux}_{\mathcal{A}} \in \{0,1\}^\star$.
- Set appropriate parameters as in Equation (5.12)[a].
- Samples $\mathsf{sd}_j \leftarrow \{0,1\}^\lambda$, $\bar{\mathbf{A}}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_{\mathsf{fhe},j} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\mathbf{R}_j \leftarrow \{0,1\}^{m\times m(\lambda+\mathrm{L})}$ and computes $\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}$ for $j\in[Q_{\mathsf{msg}}]$ where $\mathbf{A}_{\mathsf{fhe},j} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe},j} \\ \bar{\mathbf{s}}^\intercal\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^\intercal_{\mathsf{fhe},j}\end{pmatrix}$ $\forall\ j\in[Q_{\mathsf{msg}}]$.
- Samples $\bar{\mathbf{s}}_j \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_{\mathsf{s}}}, \mathbf{e}_{\mathsf{att},j} \leftarrow \mathcal{D}^{(\mathrm{L}_{\mathrm{X}}+1)m}_{\mathbb{Z},\sigma}$, sets $\mathbf{s}_j = (\bar{\mathbf{s}}_j^\intercal, -1)^\intercal$ and computes $\mathbf{c}^\intercal_{\mathsf{att},j} = \mathbf{s}_j^\intercal(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},j}$ $\forall\ j\in[Q_{\mathsf{msg}}]$.
- Samples $\mathbf{r}_k \leftarrow \{0,1\}^\lambda$, defines $\mathrm{F}[f_k,\mathbf{r}_k]$ and computes $\mathbf{A}_{\mathrm{F}_k}$, for $k\in[Q_{\mathsf{key}}]$, as in the key generation algorithm.
- It outputs

$$\mathbf{S} = \begin{pmatrix}\mathbf{s}_1^\intercal \\ \vdots \\ \mathbf{s}^\intercal_{Q_{\mathsf{msg}}}\end{pmatrix}, \quad \mathbf{P} = [\mathbf{A}_{\mathrm{F}_1}||\ldots||\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}}]$$

$$\mathsf{aux}_1 = \left(\{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j,\mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \{\mathbf{c}^\intercal_{\mathsf{att},j} = \mathbf{s}_j^\intercal(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}\right),$$

$$\mathsf{aux}_2 = (f_1,\ldots,f_{Q_{\mathsf{key}}}, \mathsf{aux}_{\mathcal{A}}, \mathbf{r}_1,\ldots,\mathbf{r}_{Q_{\mathsf{key}}}, \mathbf{A}_{\mathsf{att}}, M).$$

[a] We assume the parameters to be output as a part of aux_2, even though we do not explicitly write so.

Figure 5.1: Description of the Sampler for Evasive LWE

where $\mathbf{E}_{\mathbf{P}} \leftarrow \mathcal{D}^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}}_{\mathbb{Z},\sigma_1}$. Using the representation

$$\mathbf{C}_{\mathbf{B}} = \begin{pmatrix} \mathbf{c}^\intercal_{\mathbf{B},1} = \mathbf{s}_1^\intercal\mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B},1} \\ \vdots \\ \mathbf{c}^\intercal_{\mathbf{B},Q_{\mathsf{msg}}} = \mathbf{s}^\intercal_{Q_{\mathsf{msg}}}\mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B},Q_{\mathsf{msg}}}\end{pmatrix} = \{\mathbf{c}^\intercal_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]},$$

$$\mathbf{C}_{\mathbf{P}} = \begin{pmatrix} \mathbf{c}^\intercal_{\mathbf{P},1} = \mathbf{s}_1^\intercal\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^\intercal_{\mathbf{P},1,1}||\ldots||\mathbf{s}_1^\intercal\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^\intercal_{\mathbf{P},1,Q_{\mathsf{key}}} \\ \vdots \\ \mathbf{c}^\intercal_{\mathbf{P},Q_{\mathsf{msg}}} = \mathbf{s}^\intercal_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_1} + \mathbf{e}^\intercal_{\mathbf{P},Q_{\mathsf{msg}},1}||\ldots||\mathbf{s}^\intercal_{Q_{\mathsf{msg}}}\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}} + \mathbf{e}^\intercal_{\mathbf{P},Q_{\mathsf{msg}},Q_{\mathsf{key}}}\end{pmatrix} = \{\mathbf{c}^\intercal_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]},$$

we rewrite Equation (5.21) as follows.

$$\left(\begin{array}{r} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}^{\mathsf{T}}_{\mathbf{B},j} = \mathbf{s}_j^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j)\otimes\mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1,\mathbf{X}_j)\otimes\mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k} = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{array}\right) \approx_c \left(\begin{array}{r} \mathsf{aux}_2,\ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{array}\right) \tag{5.22}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma_1}$. Now, to prove Equation (5.19) it suffices to show Equation (5.22). We prove Equation (5.22) via the following sequence of hybrids.

Hyb_0**.** This is L.H.S distribution of Equation (5.22).

Hyb_1**.** This hybrid is same as Hyb_0, except we compute $\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k}$ as

$$\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k} = M\cdot\left\lfloor\frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j}\cdot\mathbf{H}^{\mathrm{F}_{\mathsf{high},k}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j}\cdot\mathbf{H}^{\mathrm{F}_{\mathsf{low},k}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}}{M}\right\rfloor + f_k(\mathbf{x}_j)\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma_1}$. We claim that Hyb_0 and Hyb_1 are statistically indistinguishable. To see this, we observe the following.

- From Equation (5.16) we note that

$$\begin{aligned} M\cdot\left\lfloor\frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j}\cdot\mathbf{H}^{\mathrm{F}_{\mathsf{high},k}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{att},j}\cdot\mathbf{H}^{\mathrm{F}_{\mathsf{low},k}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}_j}}{M}\right\rfloor + f_k(\mathbf{x}_j)\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j,\mathbf{r}_k) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k} \\ = \mathbf{s}_j^{\mathsf{T}}\left(M\cdot\left\lfloor\frac{\mathbf{A}_{\mathsf{high},k}}{M}\right\rfloor + \left\lfloor\frac{\mathbf{A}_{\mathsf{low},k}}{M}\right\rfloor\right) + \mathbf{err}_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k} \\ = \mathbf{s}_j^{\mathsf{T}}\mathbf{A}_{\mathrm{F}_k} + \mathbf{err}_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k} \end{aligned}$$

  where $\left\|\mathbf{err}_{j,k}\right\| \le 2^{7\lambda}\beta$.

- Next, we note that $\left\|\mathbf{err}_{j,k}\right\| \le 2^{8\lambda+O(1)}\beta/\mathrm{poly}(\lambda) = \chi_1 = \left\|\mathbf{e}_{\mathbf{P},j,k}\right\|$. Thus by noise flooding (Theorem 2.3) we have $\mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k} \approx_s \mathbf{err}_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$ with a statistical distance of $\mathrm{poly}(\lambda)2^{-\lambda}$.

From the above, we have

$$\Delta(\mathsf{Hyb}_0, \mathsf{Hyb}_1) = \frac{Q_{\mathsf{key}} \cdot Q_{\mathsf{msg}} \cdot \mathrm{poly}(\lambda)}{2^\lambda}.$$

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B}, \quad \{\mathbf{c}^\intercal_{\mathbf{B},j} = \mathbf{s}^\intercal_j \mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]} \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\intercal_{\mathsf{att},j} = \mathbf{s}^\intercal_j (\mathbf{A}_{\mathsf{att}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}_j, \mathbf{r}_k) + \mathbf{e}^\intercal_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array}\right)$$

Hyb_2. This hybrid is same as Hyb_1 except that for all $j \in [Q_{\mathsf{msg}}]$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$, where $\mathbf{A}_{\mathsf{fhe},j}$ is the fhe public key used to compute $\mathbf{X}_j$. We have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ using LWE.

To prove this we consider sub-hybrids $\mathsf{Hyb}_{1.i}$ for $i \in [Q_{\mathsf{msg}}]$, where in $\mathsf{Hyb}_{1.i}$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L_X}+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$ for $1 \le j \le i$. We set $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.0}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.Q_{\mathsf{msg}}}$. Next, we prove that for all $i \in [Q_{\mathsf{msg}}]$, $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$ via the following claim.

Claim 5.16. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$, for $i \in [Q_{\mathsf{msg}}]$, assuming the security of LWE.

Proof. We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible advantage, then there is a reduction $\mathcal{B}$ that breaks LWE security with non-negligible advantage. The reduction is as follows.

1. The adversary $\mathcal{A}$ sends the function queries $f_1, \ldots, f_{Q_{\mathsf{key}}}$, message queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$ and auxiliary input $\mathsf{aux}_{\mathcal{A}}$ to the reduction.

2. $\mathcal{B}$ initiates the LWE security game with the LWE challenger. The challenger sends $\mathbf{A}_{\mathsf{lwe}} \in \mathbb{Z}_q^{n\times mw+m+(\mathrm{L_X}+1)m}$ and $\mathbf{b} \in \mathbb{Z}_q^{mw+m+(\mathrm{L_X}+1)m}$ to $\mathcal{B}$.

3. $\mathcal{B}$ parses $\mathbf{A}_{\mathsf{lwe}} = (\mathbf{B}', \hat{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{A}'_{\mathsf{att}})$, where $\mathbf{B}' \in \mathbb{Z}_q^{n\times mw}$, $\hat{\mathbf{A}}_{\mathsf{fhe}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{A}'_{\mathsf{att}} \in \mathbb{Z}_q^{n\times(\mathrm{L_X}+1)m}$ and $\mathbf{b}^\intercal = (\mathbf{b}^\intercal_{\mathbf{B}}, \mathbf{b}^\intercal_{\mathsf{fhe}}, \mathbf{b}^\intercal_{\mathsf{att}})$. For $j \in [Q_{\mathsf{msg}}]$, it computes

$\mathbf{c}_{\mathbf{B},j}$, $\mathbf{c}_{\mathsf{att},j}$ and $\mathbf{A}_{\mathsf{fhe},j}$ as follows.

- For $1 \le j < i$: $\mathcal{B}$ samples $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_X+1)m}$ and $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.
- For $j = i$: $\mathcal{B}$ does the following.
  - Samples $\underline{\mathbf{b}} \leftarrow \mathbb{Z}_q^{mw}$ and sets $\mathbf{B} = \begin{pmatrix} \mathbf{B}' \\ \underline{\mathbf{b}}^\intercal \end{pmatrix}$ and $\mathbf{c}_{\mathbf{B},i}^\intercal \coloneqq \mathbf{b}_{\mathbf{B}}^\intercal - \underline{\mathbf{b}}^\intercal$.
  - Sets $\mathbf{A}_{\mathsf{fhe},i} \coloneqq \begin{pmatrix} \hat{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{b}_{\mathsf{fhe}}^\intercal \end{pmatrix}$ and computes $\mathbf{X}_i = \mathbf{A}_{\mathsf{fhe},i}\mathbf{R}_i - (\mathbf{x}_i, \mathsf{sd}_i) \otimes \mathbf{G}$ as in the construction.
  - Sets $\bar{\mathbf{A}}_{\mathsf{att}} = \mathbf{A}'_{\mathsf{att}} + \mathsf{bits}(1, \mathbf{X}_i) \otimes \bar{\mathbf{G}}$, $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^\intercal \end{pmatrix}$, where $\underline{\mathbf{a}}_{\mathsf{att}} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_X+1)m}$, and $\mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{b}_{\mathsf{att}}^\intercal - \left(\underline{\mathbf{a}}_{\mathsf{att}}^\intercal - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}}\right)$, where $\bar{\mathbf{G}}$ and $\underline{\mathbf{G}}$ denotes the first n rows and $n+1$-th row of the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m}$, respectively.
- For $j > i$: $\mathcal{B}$ computes $\mathbf{c}_{\mathbf{B},j}^\intercal, \mathbf{X}_j$ and $\mathbf{c}_{\mathsf{att},j}^\intercal$ as in the construction, where $\mathbf{c}_{\mathsf{att},j}^\intercal$ is computed using $\mathbf{A}_{\mathsf{att}} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^\intercal \end{pmatrix}$.

4. $\mathcal{B}$ sets $\mathsf{aux}_2 = (f_1, \ldots, f_{Q_{\mathsf{key}}}, \mathsf{aux}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_{Q_{\mathsf{key}}}, \mathbf{A}_{\mathsf{att}}, M)$ where $\mathbf{r}_k \leftarrow \{0,1\}^\lambda$ and computes $\tilde{\mathrm{F}}_{j,k}$ as in Hyb_1. It sends $(\mathsf{aux}_2, \{\mathbf{c}_{\mathbf{B},j}^\intercal, \mathbf{X}_j, \mathbf{c}_{\mathsf{att},j}^\intercal, \tilde{\mathrm{F}}_{j,k}\})$ to the adversary.
5. $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the LWE challenger.

We note that if the LWE challenger sent $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A}_{\mathsf{lwe}} + \mathbf{e}_{\mathsf{lwe}}$, then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i-1}$ with $\mathcal{A}$ else if LWE challenger sent random $\mathbf{b} \leftarrow \mathbb{Z}_q^{mw+m+(\mathrm{L}_X+1)m}$ then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i}$ with $\mathcal{A}$.

To see the former case, we note that if $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A}_{\mathsf{lwe}} + \mathbf{e}_{\mathsf{lwe}}^\intercal = \bar{\mathbf{s}}(\mathbf{B}',\ \hat{\mathbf{A}}_{\mathsf{fhe}},\ \mathbf{A}'_{\mathsf{att}}) + (\mathbf{e}_{\mathbf{B}}^\intercal,\ \mathbf{e}_{\mathsf{fhe}}^\intercal,\ \mathbf{e}_{\mathsf{att}}^\intercal)$, then $\mathbf{b}_{\mathbf{B}} = \bar{\mathbf{s}}\mathbf{B}' + \mathbf{e}_{\mathbf{B}}^\intercal$, $\mathbf{b}_{\mathsf{fhe}} \coloneqq \bar{\mathbf{s}}\hat{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal$, and $\mathbf{b}_{\mathsf{att}} \coloneqq \bar{\mathbf{s}}\mathbf{A}'_{\mathsf{att}} + \mathbf{e}_{\mathsf{att}}^\intercal$. Thus we have

$$\mathbf{c}_{\mathbf{B},i}^\intercal = (\bar{\mathbf{s}}, -1)\begin{pmatrix} \mathbf{B}' \\ \underline{\mathbf{b}}^\intercal \end{pmatrix} + \mathbf{e}_{\mathbf{B}}^\intercal, \quad \mathbf{A}_{\mathsf{fhe},i} = \begin{pmatrix} \hat{\mathbf{A}}_{\mathsf{fhe}} \\ \bar{\mathbf{s}}\hat{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix}, \quad \mathbf{c}_{\mathsf{att},i}^\intercal = (\bar{\mathbf{s}}, -1)\left(\begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{att}} \\ \underline{\mathbf{a}}_{\mathsf{att}}^\intercal \end{pmatrix} - \mathsf{bits}(1, \mathbf{X}_i) \otimes \mathbf{G}\right) + \mathbf{e}_{\mathsf{att}}^\intercal$$

To see the latter case, we note that if $\mathbf{b} \leftarrow \mathbb{Z}_q^{mw+m+(\mathrm{L}_\mathsf{X}+1)m}$ then it implies $\mathbf{b}_\mathbf{B} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{b}_\mathsf{fhe} \leftarrow \mathbb{Z}_q^{m}$, $\mathbf{b}_\mathsf{att} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathsf{X}+1)m}$. This implies the following.

- Randomness of $\mathbf{b}_\mathbf{B}$ implies the randomness of $\mathbf{c}_{\mathbf{B},i}^\intercal := \mathbf{b}_\mathbf{B}^\intercal - \underline{\mathbf{b}}^\intercal$.
- Randomness of $\mathbf{b}_\mathsf{fhe}$ implies $\mathbf{A}_{\mathsf{fhe},i} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.
- Randomness of $\mathbf{b}_\mathsf{att}$ implies randomness of $\mathbf{c}_{\mathsf{att},i}^\intercal = \mathbf{b}_\mathsf{att}^\intercal - \left(\underline{\mathbf{a}}_\mathsf{att}^\intercal - \mathsf{bits}(1, \mathbf{X}_i) \otimes \underline{\mathbf{G}}\right)$.

■

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B},\ \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_\mathsf{msg}]},\ \ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}\}_{j\in[Q_\mathsf{msg}]}, \\ \{\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathsf{X}+1)m}\}_{j\in[Q_\mathsf{msg}]},\ \ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j, \mathbf{r}_k) + \mathbf{e}_{\mathbf{P},j,k}^\intercal\}_{j\in[Q_\mathsf{msg}],k\in[Q_\mathsf{key}]}. \end{array}\right)$$

where $\mathbf{A}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{(n+1)\times m}$.

Hyb_3**.** This hybrid is same as Hyb_2 except that for $j \in [Q_\mathsf{msg}]$ we sample $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$. We have $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$ using leftover hash lemma. By leftover hash lemma (Theorem 2.4) we have that the statistical distance between $\mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j$ and a uniform matrix $U \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is $m(\lambda+\mathrm{L})/2^n$. This implies that the statistical distance between $\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{sd}_j) \otimes \mathbf{G}$ and $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}$ is $m(\lambda+\mathrm{L})/2^n$ and we have

$$\Delta(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \le \frac{Q_\mathsf{msg} \cdot m(\lambda+\mathrm{L})}{2^n} \le \frac{Q_\mathsf{msg} \cdot \mathrm{poly}(\lambda)}{2^\lambda}.$$

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B},\ \ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathsf{X}+1)m}\}_{j\in[Q_\mathsf{msg}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2\rceil + \mathsf{PRF}(\mathsf{sd}_j, \mathbf{r}_k) + \mathbf{e}_{\mathbf{P},j,k}^\intercal\}_{j\in[Q_\mathsf{msg}],k\in[Q_\mathsf{key}]}. \end{array}\right)$$

Hyb_4. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}(\mathsf{sd}_j, \cdot)$ with the real random function $\mathsf{R}^j(\cdot)$ for each $j \in [q_{\mathsf{msg}}]$. Since sd_j is not used anywhere else, we can use the security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one.

Hyb_5. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}_k\}_{k\in[Q_{\mathsf{key}}]}$, in aux_2, contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q_{\mathsf{key}}]$ such that $k \neq k'$ and $\mathbf{r}_k = \mathbf{r}_{k'}$. The probability of this happening can be bounded by $Q_{\mathsf{key}}^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q_{\mathsf{key}}^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_6. In this hybrid we compute $\tilde{\mathrm{F}}_{j,k}$ as

$$\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$$

for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. Namely, we use fresh randomness $R_{j,k} \leftarrow [-q/4 + B, q/4 - B]^{1\times\ell}$ instead of deriving the randomness by $\mathsf{R}^j(\mathbf{r}_k)$. We claim that this change is only conceptual. To see this, we observe that unless the failure condition introduced in Hyb_5 is satisfied, every invocation of the function R^j is with respect to a fresh input and thus the output can be replaced with a fresh randomness.

Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\mathbf{B},\ \begin{array}{c}\{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]},\\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2 \rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}.\end{array}\right)$$

Hyb_7. This hybrid is same as the previous one except we sample $R_{j,k} \leftarrow [-q/4, q/4]^{1\times\ell}$.

We note that $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$. To see this note that the statistical distance between the uniform distributions $U_1 = [-q/4 + B, q/4 - B]$ and $U_2 = [-q/4, q/4]$ is

$$\Delta(U_1, U_2) = \frac{1}{2}\left|\frac{2}{q-4B} - \frac{2}{q}\right| \leq \frac{4B}{q} \leq \frac{\mathrm{poly}(\lambda)}{2^\lambda}$$

by our parameter setting. Therefore,

$$\Delta(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \leq \frac{Q_{\mathsf{key}} \cdot Q_{\mathsf{msg}} \cdot \mathrm{poly}(\lambda)}{2^\lambda}.$$

Hyb_8. This hybrid is same as the previous one except we sample $\tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell$. This follows from the pseudorandomness of $\{f_k(x_j)\}_{j,k}$. To see this note that we have

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ f_k(x_j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})$$

which implies

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} = f_k(x_j)\,\lfloor q/2\rceil + R_{j,k} + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \tag{5.23}$$

$$\approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \tag{5.24}$$

where $R_{j,k} \leftarrow [-q/4, q/4]^{1\times\ell}$ and $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^\ell_{\mathbb{Z},\sigma_1}$.

Thus, using Equation (5.23) and noting that adding random strings does not make the task of distinguishing the two distributions any easier, we achieve the following distribution

$$\left(\mathsf{aux}_{\mathcal{A}},\quad \mathbf{B},\quad \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\lambda+\mathrm{L})}, \mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]},\right.$$
$$\left.\{\tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}.\right)$$

which is the R.H.S distribution of Equation (5.22), hence the proof.

■

5.3.4 Basing Security on Variant of Circular Evasive LWE ([122])

In this section, we provide our construction of a functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{\mathrm{L}(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^{\mathrm{L}} \rightarrow$

$\{0,1\}^{\ell}\}$ basing the security on a variant of the circular evasive assumption introduced by [122], which is considered public-coin.

Assumptions

Here, we state the assumptions used in this section.

Assumption 5.17 (Circular Small Secret LWE)**.** [122] Let n, m, m', q, χ, χ' be functions of λ and

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m},\ \bar{\mathbf{A}}' \leftarrow \mathbb{Z}_q^{n\times m'},\ \mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{n},\ \mathbf{s} \leftarrow (\mathbf{r}^{\intercal}, -1)^{\intercal},\ \ \mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m},\ \ \mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z},\chi'}^{m'},$$
$$\mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m},\ \ \delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{m},\ \ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \ \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$$

The circular small-secret LWE assumption $\mathsf{csLWE}_{n,m,m',q,\chi,\chi'}$ states that

$$\left\{\left(1^{\lambda}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\intercal}\end{pmatrix}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \mathbf{r}^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^{\intercal}\end{pmatrix}\mathbf{R} - \mathsf{bits}(\mathbf{s})\otimes\mathbf{G}, \bar{\mathbf{A}}', \mathbf{r}^{\intercal}\bar{\mathbf{A}}' + (\mathbf{e}')^{\intercal}\right)\right\}_{\lambda\in\mathbb{N}}$$
$$\approx \left\{\left(1^{\lambda}, \begin{pmatrix}\bar{\mathbf{A}}_{\mathsf{fhe}}\\ \delta_{\mathsf{fhe}}^{\intercal}\end{pmatrix}, \quad \Delta, \quad \bar{\mathbf{A}}', \quad (\delta_i)^{\intercal}\right)\right\}_{\lambda\in\mathbb{N}}$$

Next, we define a variant of evcsLWE assumption introduced by [122] further refined to avoid the attacks as discusses in [19].

Assumption 5.18 (evcsLWE)**.** Let $\mathcal{S}(1^{\lambda}; \mathsf{aux})$ be an algorithm that, given randomness aux, outputs

$$\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \bar{\mathbf{A}}' \in \mathbb{Z}_q^{n\times m'},\ \mathbf{P} \in \mathbb{Z}_q^{n\times J},\ \sigma,\ \sigma',\ \sigma_{-1},\ \sigma_{\mathsf{post}},\ \sigma_{\mathsf{pre}}$$

where $m \geq m_0(n,q)$ and $\sigma_{-1} \geq \sigma_0(n,m)$ and $\sigma_{\mathsf{post}} \geq \sigma_{\mathsf{pre}}$. Suppose

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}, \quad (\mathbf{B},\tau) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \quad \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}),$$

$$\mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{m},\ \mathbf{e}_{\mathsf{circ}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}^{m'},\ \mathbf{e}_{\mathbf{B}} \in \mathbb{Z}^{m},\ \mathbf{e}_{\mathbf{P}} \in \mathbb{Z}^{J},$$

$$\delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{m},\ \delta_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \delta_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{m},\ \delta_{\mathbf{P}} \leftarrow \mathbb{Z}_q^{J},$$

$$\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{n}, \quad \mathbf{s} \leftarrow (\mathbf{r}^{\intercal}, -1)^{\intercal}, \quad \mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}, \quad \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m},$$

$$\mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - (\mathbf{x}, \mathsf{bits}(\mathbf{s})) \otimes \mathbf{G} \quad \text{for} \quad \mathbf{x} \in \{0,1\}^{\mathrm{L}}$$

In the precondition, the entries of $\mathbf{e_B}, \mathbf{e_P}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ states that

$$\left\{ \begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal, \mathbf{S}, \\ \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal \\ \mathbf{r}^\intercal \bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \mathbf{r}^\intercal \mathbf{P} + \mathbf{e}_{\mathbf{P}}^\intercal \end{pmatrix} \right\}_{\lambda \in \mathbb{N}} \approx \left\{ \begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta_{\mathsf{fhe}}^\intercal, \Delta, \\ \delta_{\mathsf{circ}}^\intercal, \\ (\delta')^\intercal, \delta_{\mathbf{B}}^\intercal, \delta_{\mathbf{P}}^\intercal \end{pmatrix} \right\}_{\lambda \in \mathbb{N}}.$$

In the postcondition, the entries of $\mathbf{e_B}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ states that

$$\left\{ \begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal, \mathbf{S}, \\ \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal \\ \mathbf{r}^\intercal \bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B}}^\intercal, \mathbf{K} \end{pmatrix} \right\}_{\lambda \in \mathbb{N}} \approx \left\{ \begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta_{\mathsf{fhe}}^\intercal, \Delta, \\ \delta_{\mathsf{circ}}^\intercal, \\ (\delta')^\intercal, \delta_{\mathbf{B}}^\intercal, \mathbf{K} \end{pmatrix} \right\}_{\lambda \in \mathbb{N}}.$$

The *evasive circular small-secret LWE assumption* with respect to the sampler class $\mathcal{SC}$ states that $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ implies $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ for all efficient samplers $\mathcal{S}$.

We conjecture that for reasonable class of samplers, the evasive LWE assumption holds. In particular, we conjecture that our sampler $\mathrm{Samp}_{\mathsf{prFE}}(1^\lambda)$ used for the security proof of our prFE for natural class of functions should be in the secure class of samplers $\mathcal{SC}$ for which the evasive LWE holds.

Remark 21. In our assumption, the FHE encoding encode the attribute $\mathbf{x}$, whereas the one in [122] it only encode the FHE secret key. This is created by the difference between ABE and FE, since the attribute $\mathbf{x}$ can be public in the former and must be hidden in the latter.

Construction

The construction is same as in Section 5.3.2 except the following changes.

1. We use a concrete pseudorandom function PRF, $\mathsf{PRF} : \mathbb{Z}^{(n+1)\times m'} \times \mathbb{Z}^{n+1} \to [-q/4 + B, q/4 - B]^{1\times \ell}$ defined as $\mathsf{PRF}(\mathbf{A}, \mathbf{s}) = [\mathbf{G}_q \mathbf{G}_p^{-1}(\lfloor (\mathbf{s}^\intercal \mathbf{A})^\intercal \rfloor_p)]_B$. Here

$m' = \lceil \ell(\lceil \log q \rceil / \lceil \log p \rceil) \rceil$, $\mathbf{G}_q = \mathbf{I}_\ell \otimes (1, 2, 2^2, \ldots, 2^{\lceil \log_2 q \rceil - 1})$, $\mathbf{G}_p^{-1}(\mathbf{x})$ for a vector $\mathbf{x} \in \mathbb{Z}^{m'}$ is $(\mathsf{bits}(\mathbf{x}[1]), \ldots, \mathsf{bits}(\mathbf{x}[m']))^\intercal$, where $\mathsf{bits}(\mathbf{x}[i]) \in \{0,1\}^{\log p}$ and $[\mathbf{x}]_B$ for any $\mathbf{x} \in \mathbb{Z}_q^\ell$ represents truncating the range to $[-q/2 + B, q/2 - B]^\ell$ by mapping the out-of-range values to 0. Here B is chosen to be exponentially smaller than $q/4$. We note that for our choice of B the statistical distance between the uniform distribution over $[-q/4, q/4]$ and $[-q/4 + B, q/4 - B]$ is negligible. We show how to set p later.

2. In the setup algorithm, we set $\mathrm{L}_\mathrm{X} = m(\mathrm{L} + (n+1)\lceil \log q \rceil)(n+1)\lceil \log q \rceil$ and replace the notation $\mathbf{A}_{\mathsf{att}}$ with $\mathbf{A}_{\mathsf{circ}}$ to match the notation of the underlying assumption.

3. The $\mathsf{KeyGen}(\mathsf{msk}, f)$ algorithm has the following changes.

   - It samples $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n+1)\times m'}$ where $m' = \lceil \ell(\lceil \log q \rceil / \lceil \log p \rceil) \rceil$ instead of $\mathbf{r} \leftarrow \{0,1\}^\lambda$.

   - It defines function $\mathrm{F}[f, \mathbf{A}']$ (instead of $\mathrm{F} = \mathrm{F}[f, \mathbf{r}]$), with $f, \mathbf{A}'$ hardwired, as follows:
     On input $(\mathbf{x} \in \{0,1\}^\mathrm{L}, \mathsf{bits}(\mathbf{s}) \in \{0,1\}^{(n+1)\lceil \log q \rceil})$, first recover $\mathbf{s} \in \mathbb{Z}^{n+1}$ from $\mathsf{bits}(\mathbf{s})$ and then compute and output $f(\mathbf{x})\lfloor q/2 \rceil + \mathsf{PRF}(\mathbf{A}', \mathbf{s}) \in \mathbb{Z}_q^{1\times \ell}$.

   The rest of the algorithm remains as it is. It outputs $\mathsf{sk}_f = (\mathbf{K}, \mathbf{A}')$.

4. The encryption algorithm has the following changes

   - It samples $\mathbf{R}$ differently as $\mathbf{R} \leftarrow \{0,1\}^{m \times m(\mathrm{L} + (n+1)\lceil \log q \rceil)}$ and computes $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{bits}(\mathbf{s})) \otimes \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times m(\mathrm{L} + (n+1)\lceil \log q \rceil)}$.

   - We also change the notation of $\mathbf{c}_{\mathsf{att}}$ and denote this as $\mathbf{c}_{\mathsf{circ}}^\intercal = \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ}}^\intercal$ for $\mathbf{e}_{\mathsf{circ}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{(\mathrm{L}_\mathrm{X}+1)m}$.

   The rest of the algorithm remains as it is.

The above modified prFE scheme satisfies correctness. This can be argued using same steps as for the construction in Section 5.3.2.

Security

Theorem 5.19. *Let $\mathcal{SC}_{\mathsf{prFE}}$ be a sampler class for prFE. Assuming circular small-secret LWE (Assumption 5.17) and evasive circular small-secret LWE (Assumption 5.18) with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{evcs}}(1^\lambda)$ induced by $\mathrm{Samp}_{\mathsf{prFE}} \in \mathcal{SC}_{\mathsf{prFE}}$ as defined in Figure 5.2, our prFE scheme satisfies prCT security with respect to $\mathcal{SC}_{\mathsf{prFE}}$ as defined in Definition 5.13.*

Proof. For a prFE sampler $\mathrm{Samp}_{\mathsf{prFE}}$ as defined in the proof for Theorem 5.15, we show

$$\left(\begin{array}{r} \mathsf{mpk} = (\mathbf{A}_{\mathsf{circ}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\quad \mathbf{C}_{\mathbf{B}} = \mathbf{SB} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} = \mathbf{s}_j^{\mathsf{T}}(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]} \end{array}\right)$$

$$\approx_c \left(\begin{array}{r} \mathsf{mpk} = (\mathbf{A}_{\mathsf{circ}}, \mathbf{B}, M),\ \mathsf{aux}_{\mathcal{A}},\quad \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{K}_k, \mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]} \end{array}\right) \tag{5.25}$$

where

$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{s}_{Q_{\mathsf{msg}}}^{\mathsf{T}} \end{pmatrix},\ \mathbf{E}_{\mathbf{B}} = \begin{pmatrix} \mathbf{e}_{\mathbf{B},1}^{\mathsf{T}} \\ \vdots \\ \mathbf{e}_{\mathbf{B},Q_{\mathsf{msg}}}^{\mathsf{T}} \end{pmatrix},$$

$$(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}_j\}_{j\in[Q_{\mathsf{msg}}]}) \leftarrow \mathrm{Samp}_{\mathsf{prFE}}(1^\lambda)$$

and for $j \in [Q_{\mathsf{msg}}]$, $\mathbf{s}_j, \mathbf{e}_{\mathbf{B},j}, \mathbf{A}_{\mathsf{fhe},j}, \mathbf{R}_j, \mathsf{sd}_j, \mathbf{e}_{\mathsf{circ},j}$ are sampled as in the construction, for $k \in [Q_{\mathsf{key}}]$, we have $\mathbf{A}' \leftarrow \mathbb{Z}_q^{(n+1)\times m'}$, $\mathrm{F}_k = \mathrm{F}[f_k, \mathbf{A}'_k]$ and $\mathbf{A}_{\mathrm{F}_k}$ is as defined in the construction, and $\mathbf{K}_k = \mathbf{B}_\tau^{-1}(\mathbf{A}_{\mathrm{F}_k})$

assuming we have

$$(1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ f_k(x_j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux}_{\mathcal{A}},\ \{f_k,\ \Delta_{j,k} \leftarrow \{0,1\}^\ell\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}). \tag{5.26}$$

Consider the evcsLWE sampler $\mathrm{Samp}_{\mathsf{evcs}}(1^\lambda, \mathsf{aux})$ as defined in Figure 5.2. To prove Equation (5.25), assuming evcsLWE assumption (Assumption 5.18) with sampler

$\mathsf{Samp}_{\mathsf{evcs}}(1^\lambda, \mathsf{aux})$

The sampler does the following.

- Parse $\mathsf{aux} = (\mathsf{aux}_{\mathcal{A}}, \{f_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{A}'_k \leftarrow \mathbb{Z}_q^{(n+1)\times m'}\}_{k\in[Q_{\mathsf{key}}]}, \mathsf{aux}_{\mathrm{Samp}})$.
- Sample parameters as in Equation (5.12). Additionally set $p = q/(2^\lambda\sigma\sqrt{\lambda})$[a].
- Samples $\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(\mathrm{L}_{\mathrm{X}}+1)m}$ using the randomness $\mathsf{aux}_{\mathrm{Samp}}$.
- Defines $\mathrm{F}_k[f_k, \mathbf{A}'_k]$ and computes $\mathbf{A}_{\mathrm{F}_k}$ as in the construction. $\mathbf{A}_{\mathrm{F}_k}$ is publicly computable given f_k, $\mathbf{A}'_k$, M and $\mathbf{A}_{\mathsf{circ}}$.
- It outputs

$$\mathbf{A}_{\mathsf{circ}}, \quad \mathbf{P} = [\mathbf{A}_{\mathrm{F}_1}||\ldots||\mathbf{A}_{\mathrm{F}_{Q_{\mathsf{key}}}}], \ \mathsf{aux} = (\mathsf{aux}_{\mathrm{Samp}}, \mathsf{aux}_{\mathsf{prFE}}, \{\mathbf{A}'_k\}_{k\in[Q_{\mathsf{key}}]})$$

[a] We assume the parameters to be part of the output of $\mathsf{Samp}_{\mathsf{evcs}}$, even though we do not explicitly write so.

Figure 5.2: Description of the Sampler for circular small-secret evasive LWE

$\mathsf{Samp}_{\mathsf{evcs}}$ it suffices to show

$$\left(\begin{array}{r} \mathsf{aux}_1, \ \mathbf{B}, \ \mathbf{C}_{\mathbf{B}} = \mathbf{S}\mathbf{B} + \mathbf{E}_{\mathbf{B}}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\intercal_{\mathsf{circ},j} = \mathbf{s}_j^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} = \mathbf{S}\mathbf{P} + \mathbf{E}_{\mathbf{P}} \end{array}\right) \approx_c \left(\begin{array}{r} \mathsf{aux}_1, \ \mathbf{B}, \ \mathbf{C}_{\mathbf{B}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times mw}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \mathbf{C}_{\mathbf{P}} \leftarrow \mathbb{Z}_q^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}} \end{array}\right) \tag{5.27}$$

where $\mathsf{aux}_1 = (\mathsf{aux}, M)$ and $\mathbf{E}_{\mathbf{P}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma_1}^{Q_{\mathsf{msg}}\times\ell\cdot Q_{\mathsf{key}}}$. Using the representation in Section 5.3.3, we rewrite the Equation (5.27) as

$$\left(\begin{array}{r} \mathsf{aux}_1, \ \mathbf{B}, \{\mathbf{c}^\intercal_{\mathbf{B},j} = \mathbf{s}_j^\intercal\mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\intercal_{\mathsf{circ},j} = \mathbf{s}_j^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^\intercal_{\mathbf{P},j,k} = \mathbf{s}_j^\intercal\mathbf{A}_{\mathrm{F}_k} + \mathbf{e}^\intercal_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{array}\right) \approx_c \left(\begin{array}{r} \mathsf{aux}_1, \ \mathbf{B}, \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathbf{P},j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]} \end{array}\right) \tag{5.28}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma_1}$.

We prove Equation (5.28) using the following sequence of hybrids.

Hyb_0. This is L.H.S distribution of Equation (5.28).

Hyb_1. This hybrid is same as Hyb_0, except we compute $\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k}$ as

$$\mathbf{c}^{\mathsf{T}}_{\mathbf{P},j,k} = M \cdot \left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} \cdot \mathbf{H}^{\mathsf{F}_{\mathsf{high},k}}_{\mathbf{A}_{\mathsf{circ}},\mathbf{X}_j}}{M} \right\rceil + \left\lfloor \frac{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} \cdot \mathbf{H}^{\mathsf{F}_{\mathsf{low},k}}_{\mathbf{A}_{\mathsf{circ}},\mathbf{X}_j}}{M} \right\rceil + f_k(\mathbf{x}_j)\,\lfloor q/2 \rceil + \mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}$$

where $\mathbf{e}_{\mathbf{P},j,k} \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma_1}$. The proof of $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$ is exactly the same as the proof of $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$ in Theorem 5.15, hence we omit it. So it suffices to show pseudorandomness of the following distribution given aux_1

$$\left(\begin{array}{c} \mathbf{B},\ \ \{\mathbf{c}^{\mathsf{T}}_{\mathbf{B},j} = \mathbf{s}^{\mathsf{T}}_j \mathbf{B} + \mathbf{e}^{\mathsf{T}}_{\mathbf{B},j}\}_{j\in[Q_{\mathsf{msg}}]} \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}^{\mathsf{T}}_{\mathsf{circ},j} = \mathbf{s}^{\mathsf{T}}_j (\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}_{\mathsf{circ},j}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\,\lfloor q/2 \rceil + [\mathbf{G}_q \mathbf{G}_p^{-1}(\lfloor (\mathbf{s}^{\mathsf{T}}_j \mathbf{A}'_k)^{\mathsf{T}} \rfloor_p)]_B + \mathbf{e}^{\mathsf{T}}_{\mathbf{P},j,k}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array}\right)$$

where we write $\mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) = \left[\mathbf{G}_q \mathbf{G}_p^{-1}\left(\left\lfloor (\mathbf{s}^{\mathsf{T}}_j \mathbf{A}'_k)^{\mathsf{T}} \right\rfloor_p\right)\right]_B$.

Hyb_2: This hybrid is same as Hyb_1 except that for all $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$, we compute the PRF differently. We compute $\mathsf{PRF}(\mathbf{A}'_k, \mathbf{s}_j) = \left[\mathbf{G}_q \mathbf{G}_p^{-1}\left(\left\lfloor (\bar{\mathbf{s}}^{\mathsf{T}}_j \bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\mathsf{T}} + \mathbf{e}^{\mathsf{T}}_k)^{\mathsf{T}} \right\rfloor_p\right)\right]_B$ for $\mathbf{e}_k \leftarrow \mathcal{D}^{\ell}_{\mathbb{Z},\sigma}$, where $\bar{\mathbf{A}}'_k$ denotes the first n rows and $\underline{\mathbf{a}}'_k$ denotes the last row of the matrix $\mathbf{A}'_k$. We claim that $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$ with all but negligible probability. To see this, we note the following.

- The statistical distance between $(\bar{\mathbf{s}}^{\mathsf{T}}_j \bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\mathsf{T}} + \mathbf{e}^{\mathsf{T}}_k)^{\mathsf{T}}$ and $(\bar{\mathbf{s}}^{\mathsf{T}}_j \bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^{\mathsf{T}})^{\mathsf{T}}$ is bounded by $\|\mathbf{e}_k\| \le \sigma\sqrt{\lambda}$.

– So,
$\mathsf{Pr}\left[\left\lfloor(\bar{\mathbf{s}}_j^\intercal \bar{\mathbf{A}}'_k - \underline{\mathbf{a}}'_k)^\intercal)^\intercal\right\rfloor_p \neq \left\lfloor(\bar{\mathbf{s}}_j^\intercal \bar{\mathbf{A}}'_k - \underline{\mathbf{a}}'_k)^\intercal + \mathbf{e}_k^\intercal)^\intercal\right\rfloor_p\right] \leq \sigma\sqrt{\lambda}p/q = 1/2^\lambda$
due to our parameter setting.

– Taking the probability over all $k \in [Q_{\mathsf{key}}]$, we get $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$ with all but $Q_{\mathsf{key}}/2^\lambda$ probability.

So it suffices to show pseudorandomness of the following distribution given aux_1

$$\left(\begin{array}{c} \mathbf{B},\ \{\mathbf{c}_{\mathbf{B},j}^\intercal = \mathbf{s}_j^\intercal \mathbf{B} + \mathbf{e}_{\mathbf{B},j}^\intercal\}_{j\in[Q_{\mathsf{msg}}]} \\ \{\mathbf{X}_j = \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}_j - (\mathbf{x}_j, \mathsf{bits}(\mathbf{s}_j)) \otimes \mathbf{G}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j}^\intercal = \mathbf{s}_j^\intercal(\mathbf{A}_{\mathsf{circ}} - \mathsf{bits}(1,\mathbf{X}_j) \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{circ},j}^\intercal\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2\rceil + [\mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor(\bar{\mathbf{s}}_j^\intercal \bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^\intercal + \mathbf{e}_k^\intercal)^\intercal\rfloor_p)]_B + \mathbf{e}_{\mathbf{P},j,k}^\intercal\}_{j\in[Q_{\mathsf{msg}}]}, k \in [Q_{\mathsf{key}}]. \end{array}\right)$$

Hyb_3**:** This hybrid is same as Hyb_2 except that for all $j \in [Q_{\mathsf{msg}}]$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathrm{X}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^\ell$, where $R_{j,k} = \mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor(\bar{\mathbf{s}}_j^\intercal \bar{\mathbf{A}}'_k - (\underline{\mathbf{a}}'_k)^\intercal + \mathbf{e}_k^\intercal)^\intercal\rfloor_p)$ in Hyb_2 . We have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ assuming the hardness of circular small secret LWE.

To prove this we consider sub-hybrids $\mathsf{Hyb}_{1.i}$ for $i \in [Q_{\mathsf{msg}}]$, where in $\mathsf{Hyb}_{1.i}$ we sample $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{att},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathrm{X}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^\ell$ for $1 \leq j \leq i$. We set $\mathsf{Hyb}_1 = \mathsf{Hyb}_{1.0}$ and $\mathsf{Hyb}_2 = \mathsf{Hyb}_{1.Q_{\mathsf{msg}}}$. We prove that for all $i \in [Q_{\mathsf{msg}}]$, $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$ in Claim 5.20. Thus, it suffices to show pseudorandomness of the following distribution given aux_2

$$\left(\begin{array}{c} \mathbf{B},\ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}\}_{j\in[Q_{\mathsf{msg}}]},\ \{\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}\}_{j\in[Q_{\mathsf{msg}}]}, \\ \{\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_\mathrm{X}+1)m}\}_{j\in[Q_{\mathsf{msg}}]},\ \{\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2\rceil + [R_{j,k}]_B + \mathbf{e}_{\mathbf{P},j,k}^\intercal\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}. \end{array}\right).$$

Hyb_4**.** This hybrid is same as the previous one except we sample $\tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^\ell$. This follows from the pseudorandomness of $\{f_k(x_j)\}_{j,k}$. Thus we achieve the following

distribution

$$\left(\mathsf{aux}_{\mathcal{A}},\ \mathbf{B},\ \{\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}, \mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil \log q\rceil)}, \mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}\}_{j\in[Q_{\mathsf{msg}}]}, \right.$$
$$\left. \{\tilde{\mathrm{F}}_{j,k} \leftarrow \mathbb{Z}_q^{\ell}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}.\right)$$

which is the R.H.S distribution of Equation (5.22), hence the proof.

■

Claim 5.20. $\mathsf{Hyb}_{1.i-1} \approx_c \mathsf{Hyb}_{1.i}$.

Proof. We show that if there exists an adversary $\mathcal{A}$ who can distinguish between the two hybrids with non-negligible advantage, then there is a reduction $\mathcal{B}$ that breaks csLWE security with non-negligible advantage. The reduction is as follows.

1. The adversary $\mathcal{A}$ sends the function queries $f_1, \ldots, f_{Q_{\mathsf{key}}}$, message queries $\mathbf{x}_1, \ldots, \mathbf{x}_{Q_{\mathsf{msg}}}$ and auxiliary input $\mathsf{aux}_{\mathcal{A}}$ to the reduction.

2. $\mathcal{B}$ initiates the csLWE security game with the csLWE challenger. The csLWE challenger samples a bit $\beta \leftarrow \{0,1\}$ and does the following

   - It samples $\bar{\mathbf{A}}_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^{n\times m}$, $\bar{\mathbf{A}}'_j \leftarrow \mathbb{Z}_q^{n\times(mw+(\mathrm{L}_{\mathrm{X}}+1)m+m'\cdot Q_{\mathsf{key}})}$, $\bar{\mathbf{s}}_j \leftarrow \mathcal{D}^n_{\mathbb{Z},\sigma_{\mathbf{s}}}$, $\mathbf{e}_{\mathsf{fhe},j} \leftarrow \mathcal{D}^m_{\mathbb{Z},\sigma}$, $\mathbf{e}'_j \leftarrow \mathcal{D}^{mw+(\mathrm{L}_{\mathrm{X}}+1)m+m'\cdot Q_{\mathsf{key}}}_{\mathbb{Z},\sigma}$, $\mathbf{R}_j \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}$, $\delta_{\mathsf{fhe},j} \leftarrow \mathbb{Z}_q^m$, $\delta'_j \leftarrow \mathbb{Z}_q^{mw+(\mathrm{L}_{\mathrm{X}}+1)m+m'\cdot Q_{\mathsf{key}}}$, $\Delta_j \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m}$. It sets $\mathbf{s}_j = (\bar{\mathbf{s}}_j^{\intercal}, -1)^{\intercal}$.

   - If $\beta = 0$, it sets $\mathbf{b}^{\intercal}_{\mathsf{fhe},j} \coloneqq \bar{\mathbf{s}}_j^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^{\intercal}_{\mathsf{fhe},j}$, $\mathbf{S}_j \coloneqq \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe},j} \\ \bar{\mathbf{s}}_j^{\intercal}\bar{\mathbf{A}}_{\mathsf{fhe},j} + \mathbf{e}^{\intercal}_{\mathsf{fhe},j}\end{pmatrix}\mathbf{R}_j - \mathsf{bits}(\mathbf{s}_j)\otimes\mathbf{G}$, and $(\mathbf{b}'_j)^{\intercal} = \bar{\mathbf{s}}_j^{\intercal}\bar{\mathbf{A}}'_j + (\mathbf{e}'_j)^{\intercal}$.

   - If $\beta = 1$, it sets $\mathbf{b}_{\mathsf{fhe},j} \coloneqq \delta_{\mathsf{fhe},j}$, $\mathbf{S}_j \coloneqq \Delta_j$, and $\mathbf{b}'_j \coloneqq \delta'_j$.

   - It returns $\left(\mathbf{A}_{\mathsf{fhe},j} = (\bar{\mathbf{A}}_{\mathsf{fhe},j}\ \mathbf{b}^{\intercal}_{\mathsf{fhe},j})^{\intercal},\ \mathbf{S}_j,\ \bar{\mathbf{A}}'_j,\ (\mathbf{b}'_j)^{\intercal}\right)$ to the reduction.

3. $\mathcal{B}$ does the following.

   - For $1 \le j < i$: $\mathcal{B}$ samples $\mathbf{c}_{\mathbf{B},j} \leftarrow \mathbb{Z}_q^{mw}$, $\mathbf{c}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}$, $\mathbf{X}_j \leftarrow \mathbb{Z}_q^{(n+1)\times m(\mathrm{L}+(n+1)\lceil\log q\rceil)}$ and $R_{j,k} \leftarrow \mathbb{Z}_q^{\ell}$.

   - For $j = i$, it does as follows.

a) It samples $\mathbf{R}'_j \leftarrow \{0,1\}^{m\times m\mathrm{L}}$, computes $\mathbf{C}_j \coloneqq \mathbf{A}_{\mathsf{fhe},j}\mathbf{R}'_j - \mathbf{x}_j \otimes \mathbf{G}$, sets $\mathbf{X}_j = \mathbf{S}_j + \mathbf{C}_j$.

b) It parses $\bar{\mathbf{A}}'_j = [\bar{\mathbf{B}}, \bar{\mathbf{A}}'_{\mathsf{circ},j}, \bar{\mathbf{A}}'_{j,1}, \ldots \bar{\mathbf{A}}'_{j,Q_{\mathsf{key}}}]$, where $\bar{\mathbf{B}} \in \mathbb{Z}_q^{n\times mw}$, $\bar{\mathbf{A}}'_{\mathsf{circ},j} \in \mathbb{Z}_q^{n\times(\mathrm{L}_{\mathrm{X}}+1)m}$, $\bar{\mathbf{A}}'_{j,k} \in \mathbb{Z}_q^{n\times m'}$, and similarly $(\mathbf{b}'_j)^\intercal = \left((\mathbf{b}_j)^\intercal, (\mathbf{b}'_{\mathsf{circ},j})^\intercal, (\mathbf{b}'_{j,1})^\intercal, \ldots, (\mathbf{b}'_{j,Q_{\mathsf{key}}})^\intercal\right)$, where $\mathbf{b}_j \in \mathbb{Z}_q^{mw}, \mathbf{b}'_{\mathsf{circ},j} \in \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}$ and $\mathbf{b}'_{j,k} \in \mathbb{Z}_q^{m'}$ for $k \in [Q_{\mathsf{key}}]$.

c) It samples $\underline{\mathbf{b}} \leftarrow \mathbb{Z}_q^{mw}, \underline{\mathbf{a}}_{\mathsf{circ},j} \leftarrow \mathbb{Z}_q^{(\mathrm{L}_{\mathrm{X}}+1)m}$ and sets $\mathbf{c}_{\mathbf{B},j} \coloneqq (\mathbf{b}_j)^\intercal - \underline{\mathbf{b}}^\intercal$ and $\mathbf{c}^\intercal_{\mathsf{circ},j} \coloneqq (\mathbf{b}'_{\mathsf{circ},j})^\intercal - (\underline{\mathbf{a}}^\intercal_{\mathsf{circ},j} - \mathsf{bits}(1, \mathbf{X}_j) \otimes \mathbf{G})$.

d) For $k \in [Q_{\mathsf{key}}]$, it samples $\underline{\mathbf{a}}'_k \leftarrow \mathbb{Z}_q^{m'}$ and sets $R_{j,k} = (\mathbf{b}'_{j,k})^\intercal - (\underline{\mathbf{a}}'_k)^\intercal$.

– For $j > i$, it computes $\mathbf{c}_{\mathbf{B},j}$, $\mathbf{c}_{\mathsf{circ},j}$, $\mathbf{X}_j$ and $R_{j,k}$ as in Hyb_2.

4. It sets $\mathsf{aux}_1 = (\mathsf{aux}, M)$, $\mathbf{B} = (\bar{\mathbf{B}} \quad \underline{\mathbf{b}}^\intercal)^\intercal$ and $\tilde{\mathrm{F}}_{j,k} = f_k(\mathbf{x}_j)\lfloor q/2 \rceil + [\mathbf{G}_q\mathbf{G}_p^{-1}(\lfloor (R_{j,k})^\intercal \rfloor_p)]_B + \mathbf{e}^\intercal_{\mathbf{P},j,k}$ and sends $(\mathsf{aux}_1, \mathbf{B}, \mathbf{c}_{\mathbf{B},j}, \mathbf{X}_j, \mathbf{c}_{\mathsf{circ},j}, \tilde{\mathrm{F}}_{j,k})$ for all $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ forwards the bit β' to the csLWE challenger.

We note that if the csLWE challenger choose $\beta = 0$ then $\mathcal{B}$ simulated $\mathsf{Hyb}_{1,i-1}$ with $\mathcal{A}$ else it simulated $\mathsf{Hyb}_{1,i}$ with $\mathcal{A}$. ■

5.4 LACONIC PSEUDORANDOM POLY-DOMAIN OBFUSCATION

In this section, we introduce the notion of laconic pPRIO and construct it from several ingredients, which are all implied by the evasive LWE and LWE. The construction of laconic pPRIO will be used in Section 5.5. Since the intuition for this notion was discussed in Section 5.1, we proceed directly to the formal definition.

5.4.1 Definition

Syntax. A laconic pPRIO scheme supporting any circuit consists of the following algorithms.

$\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]}) \to \mathsf{dig}$. The digest algorithm takes as input the security parameter λ and an input space X of the form $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ for some

$\ell = \ell(\lambda)$ and $N \in \mathbb{N}$. We assume that X encodes the information of ℓ and N and one can retrieve them efficiently. It outputs a string dig.

$\mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E) \to \mathsf{Lobf}$. The obfuscate algorithm takes as input the security parameter λ, string dig and a circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ whose size is S. It outputs a ciphertext Lobf.

We decompose this algorithm into two phases.

$\mathsf{LObfOff}(1^\lambda, 1^S) \to (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st})$. The offline obfuscate algorithm takes as input the security parameter λ and the circuit size S. It outputs $\mathsf{Lobf}_{\mathsf{off}}$ and a state st.

$\mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E) \to \mathsf{Lobf}_{\mathsf{on}}$. The online obfuscate algorithm takes as input the state st, string dig and circuit E. It outputs $\mathsf{Lobf}_{\mathsf{on}}$.

With the above decomposition, the obfuscate algorithm outputs $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}})$.

$\mathsf{LEval}(X, \mathsf{Lobf}) \to Y$. The evaluation algorithm takes as input $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$ and Lobf. It outputs $Y = \{Y_i \in \{0,1\}^L\}_{i\in[N]}$.

Definition 5.15 (Compactness)**.** For all $\ell, N \in \mathbb{N}$ and $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$, we need that the size of $\mathsf{dig} = \mathsf{LDigest}(1^\lambda, X)$ should be bounded by $\mathrm{poly}(\lambda)$. In particular, the size of dig should be independent of N.

Definition 5.16 (Correctness)**.** For all $\ell, N \in \mathbb{N}$, $X = \{X_i \in \{0,1\}^\ell\}_{i\in[N]}$, and circuit $E : \{0,1\}^\ell \to \{0,1\}^L$ such that $|E| \le S$ for an arbitrary polynomial $S = S(\lambda)$, we have

$$\Pr\left[\mathsf{LEval}\left(X, (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}})\right) = \{E(X_i)\}_{i\in[N]} : \begin{array}{r} \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]}) \to \mathsf{dig} \\ \mathsf{LObfOff}(1^\lambda, 1^S) \to (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \\ \mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E) \to \mathsf{Lobf}_{\mathsf{on}} \end{array}\right] = 1.$$

Remark 22. We note that the above correctness requirement implies that for the correctness to hold, $\mathsf{LObfOff}$ only has to know the upper bound S on the size of the

circuit E that is going to be input to LObfOn and does not have to know anything else.

Definition 5.17 (Security)**.** Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(\mathsf{aux}, 1^S, X^1 = \{X_i^1\}_{i\in[N^1]}, \ldots, X^Q = \{X_i^Q\}_{i\in[N^Q]}, E^1, \ldots, E^Q\right)$$

where for all for $k \in [Q], i \in [N^k]$, $X_i^k \in \{0,1\}^{\ell^k}$, $E^k : \{0,1\}^{\ell^k} \to \{0,1\}^{L^k}$ and $|E^k| \leq S$. We say that a laconic pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\begin{aligned}\text{If } &\left(\mathsf{aux}, 1^S, X^1, \ldots, X^Q, \{E^1(X_i^1)\}_{i\in[N^1]}, \ldots, \{E^Q(X_i^Q)\}_{i\in[N^Q]}, \right)\\ &\approx_c \left(\mathsf{aux}, 1^S, X^1, \ldots, X^Q, \{\Delta_i^1\}_{i\in[N^1]}, \ldots, \{\Delta_i^Q\}_{i\in[N^Q]}\right)\end{aligned}$$

$$\begin{aligned}\text{then } &\left(\mathsf{aux}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \ldots, \mathsf{Lobf}_{\mathsf{on}}^Q\right)\\ &\approx_c \left(\mathsf{aux}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right),\end{aligned}$$

where

$\Delta_i^k \leftarrow \{0,1\}^{L^k}$ for $k \in [Q], i \in [N^k]$,

$(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$, $\mathsf{dig}^k \leftarrow \mathsf{LDigest}(1^\lambda, X^k)$, $\mathsf{Lobf}_{\mathsf{on}}^k \leftarrow \mathsf{LObfOn}(\mathsf{dig}^k, E^k)$,

$\delta^k \leftarrow \mathcal{O}_{\mathsf{on}}$ for $k \in [Q]$, where $\mathcal{O}_{\mathsf{on}}$ is the co-domain of LObfOn algorithm.

Remark 23. It is shown in [19, 65] that there is no laconic pPRIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of laconic pPRIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was laconic pPRIO that is secure for all the samplers.

5.4.2 Construction

In this section we construct a laconic pPRIO scheme for any circuit.

Building Blocks. Below, we list the ingredients for our construction.

1. A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ with key space, input space and output space as $\{0,1\}^\lambda$. The input to the PRF will be in the form of $(i\|j\|b)$ where $i \in [N]$, $j \in [\ell]$, and $b \in \{0,1\}$. Here, N and ℓ are some polynomial functions in λ. Since we have $2^\lambda > \mathrm{poly}(\lambda)$, the input space of the PRF is large enough to accommodate such inputs with an appropriate encoding. It is known that PRF can be constructed from one-way functions.

2. A blind garbling scheme $\mathsf{bGC} = (\mathsf{Garble}, \mathsf{bGC.Eval}, \mathsf{bGC.Sim})$ (defined in Section 5.2.1) for any circuit. Without loss of generality, we assume the labels are in $\{0,1\}^\lambda$. We also assume the randomness space of Garble algorithm to be $\{0,1\}^\lambda$. If longer randomness is required by the algorithm, we can expand it by using a PRF. We require the scheme to be secure as per Definition 5.3 and Definition 5.4. We can construct bGC with the required properties assuming one-way functions (See Theorem 5.13).

3. A pPRIO scheme $\mathsf{pPRIO}.(\mathsf{ObfOff}, \mathsf{ObfOn}, \mathsf{Eval})$. We require the scheme to be secure as per Definition 5.11. We can construct pPRIO with the required properties assuming evasive LWE and LWE (See Theorem 5.14).

4. A blind batch encryption scheme $\mathsf{BBE} = \mathsf{BBE}.(\mathsf{Setup}, \mathsf{Gen}, \mathsf{SingleEnc}, \mathsf{SingleDec})$ with message space $\{0,1\}^\lambda$. We assume the randomness space of $\mathsf{SingleEnc}$ algorithm to be $\{0,1\}^\lambda$ and denote the ciphertext space by $\mathcal{CT}_{\mathsf{BBE}} = \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{BBE}}}$. We require the scheme to secure as per Definition 5.8 and Definition 5.9. We also require that the CRS is a uniformly random string. As for the efficiency requirement, we need the size of the CRS to be $|\mathsf{crs}| = \mathrm{poly}(\lambda, \log N, \log \ell)$ and the size of the single ciphertext to be $|\mathsf{BBE.ct}| = \mathrm{poly}(\lambda, \log N, \log \ell)$. We can construct BBE with the required properties assuming the LWE assumption (See Theorem 5.39).

Now, we describe our construction.

$\mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]})$. The digest algorithm does the following.

- Retrieve N and ℓ from the input and run $\mathsf{crs} \leftarrow \mathsf{BBE.Setup}(1^\lambda, 1^{\ell N})$.
- Compute $h \coloneqq \mathsf{BBE.Gen}(\mathsf{crs}, X_1\| \cdots \|X_N)$, where $X_1\| \cdots \|X_N \in \{0,1\}^{\ell N}$ is the concatenation of the bit strings $X_1, \ldots, X_N \in \{0,1\}^\ell$.
- Output $\mathsf{dig} \coloneqq (\mathsf{crs}, h, N, \ell)$.

$\mathsf{LObfOff}(1^\lambda, 1^S)$. The offline obfuscate algorithm does the following.

- Generate $(\mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{pPRIO.st}) \leftarrow \mathsf{pPRIO.ObfOff}(1^\lambda, 1^{\mathsf{size}})$ where $\mathsf{size} = \mathrm{poly}(S, \lambda)$ is the maximum size of the circuit $C[\mathsf{dig}, E, \mathsf{sd}]$ defined in Figure 5.3 when the size of E is bounded by S.

– Output $\mathsf{Lobf}_{\mathsf{off}} := \mathsf{pPRIO.obf}_{\mathsf{off}}$ and $\mathsf{st} := \mathsf{pPRIO.st}$.

$\mathsf{LObfOn}(\mathsf{st}, \mathsf{dig}, E)$. The online obfuscate algorithm does the following.

– Parse $\mathsf{dig} = (\mathsf{crs}, h, N, \ell)$ and $\mathsf{st} = \mathsf{pPRIO.st}$.

– Sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.

– Construct a circuit $C[\mathsf{dig}, E, \mathsf{sd}]$ as in Figure 5.3 and compute $\mathsf{pPRIO.obf}_{\mathsf{on}} \leftarrow \mathsf{pPRIO.ObfOn}(\mathsf{pPRIO.st}, C[\mathsf{dig}, E, \mathsf{sd}])$.

– Output $\mathsf{Lobf}_{\mathsf{on}} := (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}})$.

$\mathsf{LEval}(X, \mathsf{Lobf})$. The evaluation algorithm does the following.

– Parse $X = \{X_i\}_{i\in[N]}$ and $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}} = (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}}))$.

– Set $\mathsf{pPRIO.Obf} := (\mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{pPRIO.obf}_{\mathsf{on}})$ and run $\mathsf{pPRIO.Eval}(\mathsf{pPRIO.Obf}, i) \to y_i$ for $i \in [N]$.

– Parse $y_i = (\{\mathsf{BBE.ct}_{i,j}\}_{j\in[\ell]}, \tilde{E}_i)$ for each $i \in [N]$.

– Compute $\mathsf{lab}_{i,j} \leftarrow \mathsf{BBE.SingleDec}(\mathsf{crs}, X, \ell(i-1)+j, \mathsf{BBE.ct}_{i,j})$ for each $i \in [N]$ and $j \in [\ell]$ and set $\mathsf{lab}_i := \{\mathsf{lab}_{i,j}\}$.

– Compute $z_i = \mathsf{bGC.Eval}(\mathsf{lab}_i, \tilde{E}_i)$ for $i \in [N]$.

– Output $\{z_i\}_{i\in[N]}$.

Correctness. For $X = \{X_i\}_{i\in[N]}$ and $\mathsf{Lobf} = (\mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}} = (\mathsf{crs}, \mathsf{pPRIO.obf}_{\mathsf{on}}))$, we have $\mathsf{pPRIO.obf}_{\mathsf{on}} \leftarrow \mathsf{pPRIO.ObfOn}(\mathsf{pPRIO.st}, C[\mathsf{dig}, E, \mathsf{sd}])$. From the correctness of pPRIO and the definition of $C[\mathsf{dig}, E, \mathsf{sd}]$, for each $i \in [N]$ we get

$$\mathsf{pPRIO.Eval}(\mathsf{pPRIO.Obf}, i) = y_i = (\{\mathsf{BBE.ct}_{i,j}\}_{j\in[\ell]}, \tilde{E}_i)$$

where $\mathsf{BBE.ct}_{i,j} \leftarrow \mathsf{BBE.SingleEnc}(\mathsf{crs}, h, (i-1)\ell + j, (\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1}); S_{i,j})$ for $(\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}, \tilde{E}_i) \leftarrow \mathsf{Garble}(1^\lambda, E_i; R_i)$.

Circuit $C[\mathsf{dig}, E, \mathsf{sd}](i)$**:**

Hardwired constant: $\mathsf{dig} = (\mathsf{crs}, h, N, \ell)$, circuit E, and PRF seed sd.
Given input $i \in [N]$, do the following:

1. Compute $R_i \coloneqq \mathsf{PRF}(\mathsf{sd}, i\|1\|0)$ and $S_{i,j} \coloneqq \mathsf{PRF}(\mathsf{sd}, i\|j\|1)$ for $j \in [\ell]$.
2. $(\{\mathsf{lab}_{i,j,b}\}_{j\in[\ell],b\in\{0,1\}}, \tilde{E}_i) \leftarrow \mathsf{Garble}(1^\lambda, E; R_i)$.
3. Compute $\mathsf{BBE.ct}_{i,j} \leftarrow \mathsf{BBE.SingleEnc}(\mathsf{crs}, h, (i-1)\ell + j, (\mathsf{lab}_{i,j,0}, \mathsf{lab}_{i,j,1}); S_{i,j})$ for $j \in [\ell]$.
4. Output $(\{\mathsf{BBE.ct}_{i,j}\}_{j\in[\ell]}, \tilde{E}_i)$.

Figure 5.3: The Circuit $C[\mathsf{dig}, E, \mathsf{sd}](i)$.

Next, from the perfect correctness of the BBE scheme we get

$$\mathsf{BBE.SingleDec}(\mathsf{crs}, X, \ell(i-1)+j, \mathsf{BBE.ct}_{i,j}) = \mathsf{lab}_{i,j} \text{ for } i \in [N],\ j \in [\ell]$$

Next, we set $\mathsf{lab}_i \coloneqq \{\mathsf{lab}_{i,j}\}$, which are the labels corresponding to input X_i for $i \in [N]$.

Now, from the correctness of the bGC scheme and definition of E_i, for each $i \in [N]$ we get

$$\mathsf{bGC.Eval}(\mathsf{lab}_i, \tilde{E}_i) = E(X_i)$$

and hence the correctness.

Efficiency. First, we note the following.

1. Instantiating the BBE scheme as in Theorem 5.39, we have $|\mathsf{crs}| = \mathrm{poly}(\lambda, \log N, \log \ell)$ and $|\mathsf{BBE.ct}| = \mathrm{poly}(\lambda, \log N, \log \ell)$, which are bounded by a fixed polynomial $\mathrm{poly}(\lambda)$ for any polynomially bounded N and ℓ.
2. Instantiating the pPRIO scheme as in Theorem 5.14, we have $|\mathsf{pPRIO.Obf}_{\mathsf{off}}| = \mathrm{poly}(S, \lambda)$, $|\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(S, \lambda)$.

From the above observations, our laconic pPRIO scheme satisfies

$$|\mathsf{dig}| = O(\lambda),\ |\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(S, \lambda),\ |\mathsf{Lobf}_{\mathsf{on}}| = \mathrm{poly}(S, \lambda).$$

We formalise the instantiation using the following theorem. The security of our laconic pPRIO will be proven in Theorem 5.22.

Theorem 5.21. *Assuming LWE and evasive LWE assumptions, there exists a secure (Definition 5.17) laconic* pPRIO *scheme satisfying*

$$|\mathsf{dig}| = O(\lambda),\ |\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(S,\lambda),\ |\mathsf{Lobf}_{\mathsf{on}}| = \mathrm{poly}(S,\lambda).$$

where $\mathsf{dig} \leftarrow \mathsf{LDigest}(1^\lambda, X = \{X_i\}_{i\in[N]})$, $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}) \leftarrow \mathsf{LObfuscate}(1^\lambda, \mathsf{dig}, E)$ *for circuit* $E : \{0,1\}^\ell \to \{0,1\}^L$ *whose size is bounded by* $S = S(\lambda)$.

5.4.3 Security Proof

Theorem 5.22. *Let* $\mathcal{SC}_{l\text{-}\mathsf{pPRIO}}$ *be a sampler class for laconic* pPRIO. *Assume* pPRIO *is secure (Definition 5.11) with respect to the sampler class that contains all* $\mathsf{Samp}_{\mathsf{pPRIO}}(1^\lambda)$, *induced by* $\mathrm{Samp}_{l\text{-}\mathsf{pPRIO}} \in \mathcal{SC}_{l\text{-}\mathsf{pPRIO}}$, *as defined in Section 5.4.3,* BBE *satisfies security (Definition 5.8) and strong blindness (Definition 5.9),* bGC *satisfies security (Definition 5.3) and blindness (Definition 5.4) and* PRF *is secure. Then the above construction of laconic* pPRIO *satisfies security as defined in Definition 5.17.*

Proof. Let us consider a sampler $\mathrm{Samp}_{l\text{-}\mathsf{pPRIO}}$ that outputs

$$\left(\mathsf{aux}, \{X^k = \{X_i^k \in \{0,1\}^{\ell^k}\}_{i\in[N^k]}\}_{k\in[Q]}, \{E^k\}_{k\in[Q]}\right).$$

To prove the theorem, we show that

$$\begin{aligned}&\left(\mathsf{aux}, \mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \{X^k, \mathsf{Lobf}_{\mathsf{on}}^k = (\mathsf{crs}^k, \mathsf{pPRIO.obf}_{\mathsf{on}}^k)\}_{k\in[Q]}\right)\\ &\approx_c \left(\mathsf{aux}, \mathsf{Lobf}_{\mathsf{off}} = \mathsf{pPRIO.obf}_{\mathsf{off}}, \{X^k, \delta^k \leftarrow O_{\mathsf{on}}\}_{k\in[Q]}\right) \qquad (5.29)\end{aligned}$$

holds assuming

$$\left(\mathsf{aux}, \{X^k, \{E^k(X_i^k)\}_{i\in[N^k]}\}_{k\in[Q]}\right) \approx_c \left(\mathsf{aux}, \{X^k, \{\Delta_i^k \leftarrow \{0,1\}^{L^k}\}_{i\in[N^k]}\}_{k\in[Q]}\right) \tag{5.30}$$

where O_{on} is the co-domain of LObfOn algorithm. Recalling that each crs^k is a random strting, it suffices to prove that $\{\mathsf{pPRIO.obf}_{\mathsf{on}}^k\}_k$ is pseudorandom.

We invoke the security of pPRIO scheme with respect to a sampler $\mathsf{Samp}_{\mathsf{pPRIO}}(1^\lambda)$ that outputs

$$\left(1^{N^1+\cdots+N^Q},\ \mathsf{aux}_{\mathsf{pPRIO}} = (\mathsf{aux}, \{\mathsf{crs}^k\}_{k\in[Q]}, \{X^k\}_{k\in[Q]}),\ \left\{C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k]\right\}_{k\in[Q]}\right)$$

where $\mathsf{crs}^k \leftarrow \mathsf{BBE.Setup}(1^\lambda, 1^{\ell^k N^k})$, $h^k = \mathsf{BBE.Gen}(\mathsf{crs}^k, X_1^k \| \cdots \| X_{N^k}^k)$, $\mathsf{dig}^k = (\mathsf{crs}^k, h^k, N^k, \ell^k)$, and $\mathsf{sd}^k \leftarrow \{0,1\}^\lambda$. From the security of pPRIO, we can see that it suffices to prove

$$\begin{aligned}\left(\mathsf{aux}, \{\mathsf{crs}^k\}_{k\in[Q]}, \{X^k\}_{k\in[Q]}, \left\{C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k](i)\right\}_{k\in[Q], i\in[N^k]}\right)\\ \approx_c \left(\mathsf{aux}, \{\mathsf{crs}^k\}_{k\in[Q]}, \{X^k\}_{k\in[Q]}, \left\{\gamma_i^k\right\}_{k\in[Q], i\in[N^k]}\right)\end{aligned} \tag{5.31}$$

where $C^k[\mathsf{dig}^k, E^k, \mathsf{sd}^k](i) = (\{\mathsf{BBE.ct}_{i,j}^k\}_{j\in[\ell^k]}, \tilde{E}_i^k)$ for $i \in [N^k], k \in [Q]$ and $\gamma_i^k \leftarrow \mathcal{CT}_{\mathsf{BBE}}^{\ell^k} \times \{0,1\}^{\ell_{\mathsf{bGC}}^k}$. Here, ℓ_{bGC}^k is the length of the binary string $\tilde{E}_i^k$. To prove Equation (5.31), we consider the following sequence of games.

Hyb_0. This is the LHS distribution of Equation (5.31). Rearranging the terms and recalling the definition of the circuit, we can rewrite the distribution as

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}_{i,j}^k\}_{j\in[\ell^k]}, \tilde{E}_i^k\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right)$$

where $(\{\mathsf{lab}_{i,j,b}^k\}_{j\in[\ell^k], b\in\{0,1\}}, \tilde{E}_i^k) \leftarrow \mathsf{Garble}(1^\lambda, E^k; R_i^k)$ and $\mathsf{BBE.ct}_{i,j}^k \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\mathsf{lab}_{i,j,0}^k, \mathsf{lab}_{i,j,1}^k); S_{i,j}^k)$ for $R_i^k := \mathsf{PRF}(\mathsf{sd}^k, i\|1\|0)$ and $S_{i,j}^k := \mathsf{PRF}(\mathsf{sd}^k, i\|j\|1)$.

Hyb_1. This hybrid is same as the previous one except that we compute each R_i^k and $S_{i,j}^k$ differently. Concretely, we sample $R_i^k, S_{i,j}^k \leftarrow \{0,1\}^\lambda$ for all $j \in [\ell^k], i \in [N^k]$ and $k \in [Q]$. This hybrid is computationally indistinguishable from the previous one due to the security of PRF.

Hyb_2. In this hybrid, we change how we compute $\mathsf{BBE.ct}_{i,j}^k$. Namely, we set

$$\mathsf{BBE.ct}_{i,j}^k \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\overline{\mathsf{lab}}_{i,j,0}^k, \overline{\mathsf{lab}}_{i,j,1}^k))$$

$$\text{where} \quad \overline{\mathsf{lab}}_{i,j,b}^k \begin{cases} = \mathsf{lab}_{i,j,b}^k & \text{if} \quad b = X_{i,j}^k \\ \leftarrow \{0,1\}^\lambda & \text{otherwise} \end{cases}.$$

In the above, $X_{i,j}^k$ is the j-th bit of X_i^k. By the security of BBE, this hybrid is computationally indistinguishable from the previous one. To see this, observe that the decryption outputs the labels corresponding to the j-th bit of X_i^k and we only substitute $\overline{\mathsf{lab}}_{i,j,b}^k \leftarrow \{0,1\}^\lambda$ when $b \neq X_{i,j}^k$.

Hyb_3. In this hybrid, we change how we compute $\tilde{E}_i^k$ and $\overline{\mathsf{lab}}_{i,j,b}^k$. Namely, we set

$$(\{\mathsf{lab}_{i,j}^k\}_{j\in[\ell^k]}, \tilde{E}_i^k) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{|E^k|}, 1^{\ell^k}, E^k(X_i^k)) \quad \text{and} \quad \overline{\mathsf{lab}}_{i,j,b}^k \begin{cases} = \mathsf{lab}_{i,j}^k & \text{if} \quad b = X_{i,j}^k \\ \leftarrow \{0,1\}^\lambda & \text{otherwise} \end{cases}. \tag{5.32}$$

By the simulation security of bGC (Definition 5.3), this hybrid is computationally indistinguishable from the previous one. To see this, it suffices to observe that only the information of $\{\mathsf{lab}_{i,j,X_{i,j}}^k\}_{i,j,k}$ is necessary for simulating Hyb_2 and the labels $\{\mathsf{lab}_{i,j,1-X_{i,j}}^k\}_{i,j,k}$ are not necessary.

Hyb_4. In this hybrid, we change how we compute $\tilde{E}_i^k$ and $\mathsf{lab}_{i,j}^k$. Namely, we set

$$(\{\mathsf{lab}_{i,j}^k\}_{j\in[\ell]}, \tilde{E}_i^k) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{|E^k|}, 1^\ell, \Delta_i^k),$$

where $\Delta_i^k \leftarrow \{0,1\}^{L^k}$ is chosen uniformly at random. This hybrid is computationally indistinguishable from the previous one by Equation (5.30). To see this note that Equation (5.30) implies $\{E^k(X_i^k)\}_{i\in[N^k]} \approx_c \{\Delta_i^k \leftarrow \{0,1\}^{L^k}\}_{i\in[N^k]}$, for $k \in [Q]$, given $\mathsf{aux}, \{X^k\}_{k\in[Q]}$.

Hyb_5. In this hybrid, we sample $\mathsf{lab}_{i,j}^k$ and $\tilde{E}_i^k$ as random strings. In particular, we sample $\mathsf{lab}_{i,j}^k \leftarrow \{0,1\}^\lambda$ and $\tilde{E}_i^k \leftarrow \{0,1\}^{\ell_{\mathsf{bGC}}^k}$. By the blindness of bGC scheme (Definition 5.4), this hybrid is computationally indistinguishable from the previous one. To see this, note that in the previous hybrid the simulator $\mathsf{bGC.Sim}$ takes as input Δ_i^k which is a uniformly random string and thus by the blindness property of bGC we can replace the output of $\mathsf{bGC.Sim}$ by a completely random string.

The view of the adversary in this hybrid is as follows.

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}_{i,j}^k\}_{j\in[\ell]}, \tilde{E}_i^k\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right)$$

where $\mathsf{BBE.ct}_{i,j}^k \leftarrow \mathsf{SingleEnc}(\mathsf{crs}, h^k, (i-1)\ell^k + j, (\overline{\mathsf{lab}}_{i,j,0}^k, \overline{\mathsf{lab}}_{i,j,1}^k))$ for $\overline{\mathsf{lab}}_{i,j,0}^k \leftarrow \{0,1\}^\lambda, \overline{\mathsf{lab}}_{i,j,1}^k \leftarrow \{0,1\}^\lambda$ and $\tilde{E}_i^k \leftarrow \mathsf{SIM}_{\mathsf{circ}}$.

Hyb_6. In this hybrid, we replace $\mathsf{BBE.ct}_{i,j}^k$ with a random string. Namely, we sample $\mathsf{BBE.ct}_{i,j}^k \leftarrow \mathcal{CT}_{\mathsf{BBE}}$ for all i, j, k. This hybrid is indistinguishable from previous one by strong blindness (Definition 5.9) property of the BBE scheme. To see this, note that $\mathsf{BBE.ct}_{i,j}^k$ encrypts random strings $\overline{\mathsf{lab}}_{i,j,0}^k, \overline{\mathsf{lab}}_{i,j,1}^k \leftarrow \{0,1\}^\lambda$ in the previous hybrid and thus the blindness property allows us to replace each $\mathsf{BBE.ct}_{i,j}^k$ with a random string.

The view of the adversary in this hybrid is as follows.

$$\left(\mathsf{aux}, \left\{\mathsf{crs}^k, X^k, \left\{\{\mathsf{BBE.ct}_{i,j}^k \leftarrow \mathcal{CT}_{\mathsf{BBE}}\}_{j\in[\ell]}, \tilde{E}_i^k \leftarrow \{0,1\}^{\ell_{\mathsf{bGC}}^k}\right\}_{i\in[N^k]}\right\}_{k\in[Q]}\right).$$

Rearranging the terms, we can observe that the distribution in Hyb_6 corresponds to the RHS distribution of Equation (5.31). This concludes the proof of Theorem 5.22. ■

5.5 PARTIAL-HIDING prFE FOR UNBOUNDED DEPTH WITH OPTIMAL PARAMETERS

In this section, we extend the notion of prFE to introduce partially-hiding prFE, where the part of the input to the circuit can be public. We then construct partially hiding prFE with short parameter size in this section from several ingredients, which are all implied by evasive LWE and LWE. Our construction of partially hiding prFE will be used in Section 5.6 and 5.7.

5.5.1 Definition

In this section we give the definitions for partial-hiding functional encryption for pseudorandom functionalities. Consider a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ where $C \in \mathcal{C}_{\mathsf{prm}}$ takes as input a string $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ and outputs $C(x) \in \mathcal{Y}$.

Syntax. A partial-hiding functional encryption for parameterized circuits $\mathcal{C}_{\mathsf{prm}}$ by prm consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a parameter prm and outputs a master public key mpk and a master secret key msk. We assume w.l.o.g that msk includes mpk. We also assume that prm is implicitly input to all the algorithms below.

$\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm takes as input the master secret key msk and a circuit $C \in \mathcal{C}_{\mathsf{prm}}$ and it outputs a functional secret key sk_C.

$\mathsf{Enc}(\mathsf{mpk}, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk and an input $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{CT}$, where $\mathcal{CT}$ is the ciphertext space.

For the purpose of some applications, we consider a variant where the Enc algorithm can be decomposed into the following two phases.

$\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$. The offline encryption algorithm takes as input the security parameter λ and outputs offline part of the ciphertext $\mathsf{ct}_{\mathsf{off}}$ and the state st.

$\mathsf{EncOn}(\mathsf{st}, x) \to \mathsf{ct}_{\mathsf{on}}$. The online encryption algorithm takes as input the state st and the input x and outputs the online part of the ciphertext $\mathsf{ct}_{\mathsf{on}}$.

The final output of ct is $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{ct}) \to y$. The decryption algorithm takes as input the master public key mpk, the public input $\mathbf{x}_{\mathsf{pub}}$, secret key sk_C, circuit C and a ciphertext ct and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 5.18 (Correctness)**.** A PHprFE scheme is said to satisfy *perfect* correctness if for all prm, any input $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ and circuit $C \in \mathcal{C}_{\mathsf{prm}}$, we have

$$\mathsf{Pr}\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}) \ , \ \mathsf{sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C), \\ \mathsf{Dec}\big(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{Enc}(\mathsf{mpk}, x)\big) = C(x)\end{array}\right] = 1.$$

Definition 5.19 (Security)**.** For a PHprFE scheme for circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ parameterized by $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(C^1, \ldots, C^Q, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*)$$

where Q is the number of key queries, $C^k \in \mathcal{C}_{\mathsf{prm}}$ for $k \in [Q]$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$.

We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} &\mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ C^k(x)\}_{k\in[Q]}\right) = 1] \\ &- \mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \Delta^k \leftarrow \mathcal{Y}\}_{k\in[Q]}\right) = 1]\end{aligned} \tag{5.33}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \mathsf{Enc}(\mathsf{mpk}, x),\ \mathsf{sk}_{C^k}\}_{k\in[Q]}) = 1] \\ - \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{C^k,\ x_{\mathsf{pub}},\ \delta \leftarrow \mathsf{Sim}(1^\lambda),\ \mathsf{sk}_{C^k}\}_{k\in[Q]}) = 1] \tag{5.34}$$

where $(C^1, \ldots, C^Q, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathrm{Samp}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a PHprFE scheme for circuit class $\mathcal{C}_{\mathsf{prm}}$ is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler $\mathrm{Samp} \in \mathcal{SC}$, $\mathcal{A}_1$ and Sim, there exists another PPT $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{5.35}$$

and $\mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda)$.

Remark 24 (prFE as a special PHprFE)**.** We remark that prFE is a special case of PHprFE with $\mathbf{x}_{\mathsf{pub}} = \perp$.

Next, we define the security notion that we require for the PHprFE variant where we decompose the Enc algorithm as $\mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn})$ and reuse the same state output by EncOff multiple times for generating the online part of the ciphertexts. We require the online part of the ciphertexts to be pseudorandom, whereas the offline part may not be.

Definition 5.20 (Reusable Security)**.** For a PHprFE scheme for circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ parameterized by $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(C^1, \ldots, C^{Q_{\mathsf{key}}}, x^1 = (x^1_{\mathsf{pub}}, x^1_{\mathsf{priv}}), \ldots, x^{Q_{\mathsf{msg}}} = (x^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, x^{Q_{\mathsf{msg}}}_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*\right)$$

where Q_{key} and Q_{msg} are the number of key queries and messages respectively, $x^j = (x^j_{\mathsf{pub}}, x^j_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ for $j \in [Q_{\mathsf{msg}}]$, $C^k \in \mathcal{C}_{\mathsf{prm}}$ for $k \in [Q]$.

We define the following advantage functions:

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{C^k,\ x^j_{\mathsf{pub}},\ C^k(x^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right) = 1] \\ - \mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{C^k,\ x^j_{\mathsf{pub}},\ \Delta^{j,k} \leftarrow \mathcal{Y}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right) = 1] \tag{5.36}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \mathsf{ct}_{\mathsf{off}},\ \{x^j_{\mathsf{pub}},\ \mathsf{ct}^j_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathsf{sk}_{C^k}\}_{k\in[Q_{\mathsf{key}}]}) = 1] - \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \mathsf{ct}_{\mathsf{off}},\ \{x^j_{\mathsf{pub}},\ \delta^j\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathsf{sk}_{C^k}\}_{k\in[Q_{\mathsf{key}}]}) = 1] \tag{5.37}$$

where $(C^1,\ldots,C^{Q_{\mathsf{key}}}, x^1 = (x^1_{\mathsf{pub}}, x^1_{\mathsf{priv}}),\ldots,x^{Q_{\mathsf{msg}}} = (x^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, x^{Q_{\mathsf{msg}}}_{\mathsf{priv}}), \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathrm{Samp}(1^\lambda)$, $(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda,\mathsf{prm})$, $(\mathsf{ct}_{\mathsf{off}},\mathsf{st}) \leftarrow \mathsf{EncOff}(1^\lambda)$, $\mathsf{ct}^j_{\mathsf{on}} \leftarrow \mathsf{EncOn}(\mathsf{st}, x^j), \delta^j \leftarrow \mathcal{CT}_{\mathsf{on}}$ for $j \in [Q_{\mathsf{msg}}]$ and $\mathcal{CT}_{\mathsf{on}}$ is the online part of the ciphertext space. We say that a PHprFE scheme for circuit class $\mathcal{C}_{\mathsf{prm}}$ is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{5.38}$$

and $\mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1)\cdot Q(\lambda)$.

Remark 25. Similar to Remark 19, when we use the security of PHprFE, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was PHprFE that is secure for all the samplers.

Remark 26. We remark that Definition 5.20 is stronger security notion than Definition 5.19, since the former collapses to the latter if we restrict the adversary so that $Q_{\mathsf{msg}} = 1$ and set the simulator so that it runs $\mathsf{EncOff}(\mathsf{mpk}) \rightarrow (\mathsf{ct}_{\mathsf{off}},\mathsf{st})$, samples $\delta \leftarrow \mathcal{CT}_{\mathsf{on}}$, and outputs $(\mathsf{ct}_{\mathsf{off}},\delta)$.

Remark 27. We remark that Definition 5.20 is in the single-challenge flavor in that only single offline part of the challenge ciphertext is given to the adversary (though multiple online part of the ciphertext is given to it). While it does not seem to imply multi-challenge flavor of the security definition (See Remark 20), we only define this simpler version of the definition since it suffices for our applications and leave the extension to the multi-challenge flavor for the future work.

5.5.2 Construction

In this section we provide our construction of a PHprFE scheme for circuit family $C_{L_{\mathsf{pub}},L_{\mathsf{priv}}} = \{C : \{0,1\}^{L_{\mathsf{pub}}(\lambda)} \times \{0,1\}^{L_{\mathsf{priv}}(\lambda)} \to \{0,1\}\}$. Namely, the construction supports a class of circuits whose public and private input lengths are fixed and output is binary, but its size and depth are unbounded.

Building Blocks. Below, we list the ingredients for our construction.

1. A blind garbling scheme $\mathsf{bGC} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{bGC.Sim})$ (defined in Section 5.2.1) with decomposability (defined in Definition 5.5). Without loss of generality, we assume the labels are in $\{0,1\}^\lambda$ and the random coins used by the algorithm $\{\mathsf{Garble}_i\}_i$ is in $\{0,1\}^\lambda$. The latter is for the sake of notational convenience and can be achieved by using a PRF to derive longer (pseudo-)random coins if needed. We also use $\mathcal{CT}^i_{\mathsf{bGC}}$ to denote the co-domain of $\{\mathsf{Garble}_i\}_i$ algorithm. We can construct bGC with the required properties assuming one-way functions (See Theorem 5.13).

2. A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ with key space, input space and output space as $\{0,1\}^\lambda$. It is known that PRF can be constructed from one-way functions.

3. A laconic pPRIO scheme $\mathsf{LprIO} = (\mathsf{LDigest}, \mathsf{LObfOff}, \mathsf{LObfOn}, \mathsf{LEval})$. Without loss of generality, the random coins used by LObfOn is in $\{0,1\}^\lambda$ and the state output by $\mathsf{LObfOff}$ is in $\{0,1\}^\lambda$. The former can be satisfied by using a PRF. The latter can be achieved in two steps. We first let the state Lst to be the randomness used by $\mathsf{LObfOff}$ and then replace it with a string in $\{0,1\}^\lambda$, which can be done by using a PRF. The length of the digest is of fixed polynomial in the security parameter and we denote it by $L_{\mathsf{LDigest}}(\lambda)$. As we show in Theorem 5.21, such a laconic pPRIO can be constructed by assuming evasive LWE and LWE.

4. A prFE scheme $\mathsf{prFE} = \mathsf{prFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $C_{\mathsf{inp}(\lambda),\mathsf{dep}(\lambda),\mathsf{out}(\lambda)}$ consisting of circuits with input length $\mathsf{inp}(\lambda) = L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda$, maximum depth $\mathsf{dep}(\lambda)$ and output length $\mathsf{out}(\lambda)$. We set the maximum depth and output length so that the circuit class supports the circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ defined as Figure 5.4, 5.5, and 5.6, respectively. We denote the information $(1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{dep}(\lambda)}, 1^{\mathsf{out}(\lambda)})$ specifying the circuit class by prm and the ciphertext space of the prFE scheme by $\mathcal{CT}_{\mathsf{prFE}} = \{0,1\}^{\ell^{\mathsf{ct}}_{\mathsf{prFE}}}$. Assuming LWE and evasive LWE, we can construct prFE with the required parameters (See Theorem 5.15).

For our construction, we set the following parameters

$\mathsf{Setup}(1^\lambda, 1^{L_{\mathsf{pub}}}, 1^{L_{\mathsf{priv}}})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.mpk}, \mathsf{prFE.msk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, \mathsf{prm})$.
- Output $\mathsf{mpk} \coloneqq \mathsf{prFE.mpk}$ and $\mathsf{msk} \coloneqq \mathsf{prFE.msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, C)$. The key generation algorithm does the following.

- Parse $\mathsf{msk} = \mathsf{prFE.msk}$.
- Sample a string $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
- Run $\mathsf{dig}_C \leftarrow \mathsf{LDigest}(\{(i, C_i)\}_{i \in [|C|]})$, where C_i is the description of the i-th gate of C and $|C|$ is the number of gates in C. The description of C_i can be encoded into a string of length at most 4λ since it suffices to encode its index, two indices of the incoming wires, and the type of the gate.
- Construct circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ as in Figure 5.4, 5.5, and 5.6 respectively.
- Run $\mathsf{prFE.sk}_0 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}])$, $\mathsf{prFE.sk}_1 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_1[\mathbf{r}])$, and $\mathsf{prFE.sk}_2 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_2[\mathbf{r}, \mathsf{dig}_C])$.
- Output $\mathsf{sk}_C \coloneqq (\mathbf{r}, \mathsf{prFE.sk}_0, \mathsf{prFE.sk}_1, \mathsf{prFE.sk}_2)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. The encryption algorithm does the following. We divide the algorithm into the following two steps.

$\mathsf{EncOff}(\mathsf{mpk})$. It takes as input $\mathsf{mpk} = \mathsf{prFE.mpk}$ and does the following.

- Run $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$, where S is the maximum size of the circuits $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$ defined in Figure 5.5 and Figure 5.6, respectively.
- Output $\mathsf{st} \coloneqq (\mathsf{Lst}, \mathsf{prFE.mpk})$ and $\mathsf{ct}_{\mathsf{off}} \coloneqq \mathsf{Lobf}_{\mathsf{off}}$.

$\mathsf{EncOn}(\mathsf{st}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. It does the following.

- Parse the input as $\mathsf{st} \rightarrow (\mathsf{Lst}, \mathsf{prFE.mpk})$.
- Sample $\mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2 \leftarrow \{0,1\}^\lambda$.

- Run $\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]})$, where $\mathbf{x}_{\mathsf{pub},i} \in \{0,1\}$ is the i-th bit of $\mathbf{x}_{\mathsf{pub}}$.
- Run $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2))$.
- Output $\mathsf{ct}_{\mathsf{on}} \coloneqq \mathsf{prFE.ct}$.

The final output of $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ is $\mathsf{ct} \coloneqq (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{prFE.ct})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_C, C, \mathsf{ct})$. It parses the input as $\mathsf{mpk} = \mathsf{prFE.mpk}$, $\mathsf{ct} \coloneqq (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{prFE.ct})$, $\mathsf{sk}_C = (\mathbf{r}, \mathsf{prFE.sk}_0, \mathsf{prFE.sk}_1, \mathsf{prFE.sk}_2)$ and does the following.

- Compute $\mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_0, F_0[\mathbf{r}], \mathsf{prFE.ct})$ and parse the output as $\{\mathsf{lab}'_i\}_{i\in[L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}]}$.
- Compute $\mathsf{Lobf}_{\mathsf{on},1} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_1[\mathbf{r}], \mathsf{prFE.ct})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},2} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_2[\mathbf{r}, \mathsf{dig}_C], \mathsf{prFE.ct})$.
- Set $\mathsf{Lobf}_1 \coloneqq (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},1})$ and compute $\{\widetilde{\mathsf{lab}_i}\}_{i\in[L_{\mathsf{pub}}]} = \mathsf{LEval}(\{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}, \mathsf{Lobf}_1)$.
- Set $\mathsf{Lobf}_2 \coloneqq (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},2})$ and compute $\tilde{C} = \{\tilde{C}_i\}_{i\in[|C|]} = \mathsf{LEval}(\{(i, C_i)\}_{i\in|C|}, \mathsf{Lobf}_2)$.
- Set $\mathsf{lab}_i \coloneqq \begin{cases} \widetilde{\mathsf{lab}_i} & \text{if } i \in [L_{\mathsf{pub}}] \\ \mathsf{lab}'_i & \text{if } i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}] \end{cases}$ for $i \in [L_{\mathsf{pub}} + L_{\mathsf{priv}}]$.
- Compute $z = \mathsf{Eval}(\tilde{C}, \{\mathsf{lab}_i\}_{i\in[L_{\mathsf{pub}}+L_{\mathsf{priv}}]})$.
- Output z.

Correctness. We make the following observations.

- From the perfect correctness of prFE scheme and definition of $F_0[\mathbf{r}], F_1[\mathbf{r}], F_2[\mathbf{r}, \mathsf{dig}_C]$, we get

$$\begin{aligned}\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv},i-L_{\mathsf{pub}}}}\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_0, F_0[\mathbf{r}], \mathsf{prFE.ct}) \\ &= F_0[\mathbf{r}](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2) \\ &= \{\mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]}\end{aligned}$$

Circuit $F_0[\mathbf{r}]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 \coloneqq \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$.
- Compute $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$ for $i \in [L_{\mathsf{pub}} + 1, L_{\mathsf{pub}} + L_{\mathsf{priv}}]$.
- Output $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv}},i-L_{\mathsf{pub}}}\}_{i\in[L_{\mathsf{pub}}+1,L_{\mathsf{pub}}+L_{\mathsf{priv}}]}$.

Figure 5.4: Circuit to compute garbled labels corresponding to $\mathbf{x}_{\mathsf{priv}}$.

Circuit $F_1[\mathbf{r}]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 \coloneqq \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$ and $R_1 \coloneqq \mathsf{PRF}(\mathsf{sd}_1, \mathbf{r})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},1} \leftarrow \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}, E_{\mathsf{pub}}[R_0]; R_1)$, where $E_{\mathsf{pub}}[R_0]$ is a circuit that works as follows.
  - Take $(i, b) \in [L_{\mathsf{pub}}] \times \{0, 1\}$ as an input.
  - Compute $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$.
  - Output $\mathsf{lab}_{i,b}$.
- Output $\mathsf{Lobf}_{\mathsf{on},1}$.

Figure 5.5: The Circuit $F_1[\mathbf{r}]$.

$$\begin{aligned}\mathsf{Lobf}_{\mathsf{on},1} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_1, F_1[\mathbf{r}], \mathsf{prFE.ct})\\ &= F_1[\mathbf{r}](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)\\ &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, E_{\mathsf{pub}}[R_0]; R_1),\end{aligned}$$

$$\begin{aligned}\mathsf{Lobf}_{\mathsf{on},2} &= \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_2, F_2[\mathbf{r}, \mathsf{dig}_C], \mathsf{prFE.ct})\\ &= F_2[\mathbf{r}, \mathsf{dig}_C](\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)\\ &= \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_C, E_{\mathsf{cir}}[R_0]; R_2),\end{aligned}$$

where $R_b \coloneqq \mathsf{PRF}(\mathsf{sd}_b, \mathbf{r})$ for $b \in \{0, 1, 2\}$ and $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$ are the circuits as defined in Figure 5.5 and Figure 5.6, respectively.

Circuit $F_2[\mathbf{r}, \mathsf{dig}_C]$:

- Parse the input as $(\mathsf{dig}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2)$.
- Compute $R_0 \coloneqq \mathsf{PRF}(\mathsf{sd}_0, \mathbf{r})$ and $R_2 \coloneqq \mathsf{PRF}(\mathsf{sd}_2, \mathbf{r})$.
- Compute $\mathsf{Lobf}_{\mathsf{on},2} \leftarrow \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_C, E_{\mathsf{cir}}[R_0]; R_2)$, where $E_{\mathsf{cir}}[R_0]$ is a circuit that works as follows.
  - Take $(i, G) \in [|C|] \times \{0,1\}^{4\lambda}$ as an input, where G encodes the information of a gate.
  - Compute $\tilde{G} \leftarrow \mathsf{Garble}_i(1^\lambda, G; R_0)$.
  - Output $\tilde{G}$.
- Output $\mathsf{Lobf}_{\mathsf{on},2}$.

Figure 5.6: The Circuit $F_2[\mathbf{r}, \mathsf{dig}_C]$.

– Next, parsing $\mathsf{Lobf}_1 \coloneqq (\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on},1})$, we get

$$\begin{aligned}\{\widetilde{\mathsf{lab}_i}\}_{i\in[L_{\mathsf{pub}}]} &= \mathsf{LEval}(\{(i, \mathbf{x}_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}, \mathsf{Lobf}_1)\\ &= \left\{E_{\mathsf{pub}}[R_0](i, \mathbf{x}_{\mathsf{pub},i})\right\}_{i\in[L_{\mathsf{pub}}]}\\ &= \mathsf{lab}_{i,\mathbf{x}_{\mathsf{pub},i}}\end{aligned}$$

where $(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}) \leftarrow \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R_0)$ and

$$\begin{aligned}\tilde{C} &= \{\tilde{C}_i\}_{i\in[|C|]}\\ &= \mathsf{LEval}(\{(i, C_i)\}_{i\in|C|}, \mathsf{Lobf}_2)\\ &= \{E_{\mathsf{cir}}[R_0](i, C_i)\}_{i\in|C|}\\ &= \{\mathsf{Garble}_i(1^\lambda, C_i; R_0)\}_{i\in|C|}\end{aligned}$$

from the perfect correctness of the laconic pPRIO scheme and the definition of $E_{\mathsf{pub}}[R_0]$ and $E_{\mathsf{cir}}[R_0]$.

– Next, from the perfect correctness of the garbling scheme bGC, we have, for $\mathsf{lab}_i = (\mathsf{lab}_{\mathbf{x}_{\mathsf{pub}}}, \mathsf{lab}_{\mathbf{x}_{\mathsf{priv}}})$,

$$\mathsf{Eval}(\tilde{C}, \{\mathsf{lab}_i\}_{i\in[L_{\mathsf{pub}}+L_{\mathsf{priv}}]}) = C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$$

and hence the perfect correctness of the PHprFE scheme.

Efficiency Here, we show

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}}), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda, L_{\mathsf{priv}}), \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}}).$$

To do so, we first show that the sizes of the circuits $F_0[\mathbf{r}]$, $F_1[\mathbf{r}]$, and $F_2[\mathbf{r}, \mathsf{dig}_C]$ are all bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- In $F_0[\mathbf{r}]$, the computation of the PRF and $\mathsf{Garble}_{\mathsf{inp},i}$ are performed, where both of them can be implemented by circuits of size $\mathrm{poly}(\lambda)$. Since the latter is repeated for L_{priv} times, the overall size of $F_0[\mathbf{r}]$ is $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- To bound the size of $F_1[\mathbf{r}]$, we first bound the size of $E_{\mathsf{pub}}[R_0]$. $E_{\mathsf{pub}}[R_0]$ performs the computation of $\mathsf{Garble}_{\mathsf{inp},i}$ on input $i \in [L_{\mathsf{pub}}]$ (in binary), which can be implemented by a circuit of size $\mathrm{poly}(\log L_{\mathsf{pub}}, \lambda)$. This can be further bounded by $\mathrm{poly}(\lambda)$ since $L_{\mathsf{pub}} \leq 2^\lambda$. We then observe that $F_1[\mathbf{r}]$ consists of the evaluation of two PRF values and LObfOn on input Lst, dig, and $E_{\mathsf{pub}}[R_0]$. We can bound the input length to LObfOn by a fixed polynomial, since we have $|\mathsf{Lst}| = \lambda$ and $|\mathsf{dig}| = \mathrm{poly}(\lambda)$ and LObfOn runs in polynomial time in the input length, its size is bounded by $\mathrm{poly}(\lambda)$. Therefore, the overall size of $F_1[\mathbf{r}]$ is $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$, where we take into account the input $\mathbf{x}_{\mathsf{priv}}$, which is ignored in the computation.

- To bound the size of $F_2[\mathbf{r}, \mathsf{dig}_C]$, we first bound the size of $E_{\mathsf{cir}}[R_0]$. $E_{\mathsf{cir}}[R_0]$ performs the computation of Garble_i on input $i \in [|C|]$ (in binary) and $G \in \{0,1\}^{4\lambda}$, which can be implemented by a circuit of size $\mathrm{poly}(\log |C|, \lambda)$. This can be further bounded by $\mathrm{poly}(\lambda)$, since $|C| \leq 2^\lambda$. Similarly to the case of $F_1[\mathbf{r}]$, the size of $F_2[\mathbf{r}]$ can be bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

We then move to discuss the size of the parameters.

- We can bound $|\mathsf{mpk}| = |\mathsf{prFE.mpk}|$ by $\mathrm{poly}(\lambda, \mathsf{inp}, \mathsf{dep}, \mathsf{out})$, since it is output by $\mathsf{prFE.Setup}(1^\lambda, \mathsf{prm} = (1^{\mathsf{inp}}, 1^{\mathsf{dep}}, 1^{\mathsf{out}}))$. We have $\mathsf{inp}(\lambda) \leq L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda \leq \mathrm{poly}(\lambda, L_{\mathsf{priv}})$ and $\mathsf{dep}(\lambda)$ and $\mathsf{out}(\lambda)$ are bounded by the maximum size of the circuits E_{cir} and E_{pub}, which in turn is bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We therefore have $|\mathsf{mpk}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- We have $|\mathsf{sk}_C| = \lambda + |\mathsf{prFE.sk}_0| + |\mathsf{prFE.sk}_1| + |\mathsf{prFE.sk}_2|$. We can bound the size of $|\mathsf{prFE.sk}_0|$ by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$, since it is generated by $\mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}])$, where the input length to $\mathsf{prFE.KeyGen}$ is bounded by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. Similarly, we can bound $|\mathsf{prFE.sk}_1|$ and $|\mathsf{prFE.sk}_2|$ by $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. Therefore, we have $|\mathsf{sk}_C| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

- We have $|\mathsf{ct}| = |\mathsf{Lobf}_{\mathsf{off}}| + |\mathsf{prFE.ct}|$. We first bound $|\mathsf{Lobf}_{\mathsf{off}}|$. From the above discussion, we have $S = \max\{|E_{\mathsf{pub}}[R_0]|, |E_{\mathsf{cir}}[R_0]|\} = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$. This implies $|\mathsf{Lobf}_{\mathsf{off}}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$, since $\mathsf{Lobf}_{\mathsf{off}}$ is output by $\mathsf{LObfOff}(1^\lambda, 1^S)$. We then bound $|\mathsf{prFE.ct}|$, which is output by

$\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}_{\mathsf{pub}}}, \mathbf{x}_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}_0, \mathsf{sd}_1, \mathsf{sd}_2))$. Since the input length of $\mathsf{prFE.Enc}$ is $\mathsf{inp}(\lambda) = L_{\mathsf{priv}} + L_{\mathsf{LDigest}}(\lambda) + 4\lambda = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We therefore have $|\mathsf{ct}| = |\mathsf{Lobf}_{\mathsf{off}}| + |\mathsf{prFE.ct}| = \mathrm{poly}(\lambda, L_{\mathsf{priv}})$.

5.5.3 Security

Before proving the security of our scheme, we prove the following useful lemma. The lemma essentially says that if a part of the auxiliary information is pseudorandom in the pre-condition distribution, then it is pseudorandom in the corresponding post-condition distribution where we apply laconic pPRIO security. A conceptually similar lemma is proven in Lemma 3.4 of [22] in the context of evasive LWE.

Lemma 5.23. *Let* $\mathsf{LprIO} = (\mathsf{LDigest}, \mathsf{LObfuscate}, \mathsf{LEval})$ *be a laconic* pPRIO *scheme and* Samp *be a PPT algorithm that takes as input* 1^λ *and outputs*

$$\left(\mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \{0,1\}^* \times \mathcal{X}, 1^S, X^1 = \{X_i^1\}_{i\in[N^1]}, \ldots, X^Q = \{X_i^Q\}_{i\in[N^Q]}, E^1, \ldots, E^Q\right)$$

for some set $\mathcal{X}$. *Here* $X_i^k \in \{0,1\}^{\ell^k}$, $E^k : \{0,1\}^{\ell^k} \to \{0,1\}^{L^k}$ *and* $|E^k| \leq S$ *for* $k \in [Q], i \in [N^k]$.

Let us assume that

$$\begin{aligned}&\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^S, X^1, \ldots, X^Q, \{E^1(X_i^1)\}_{i\in[N^1]}, \ldots, \{E^Q(X_i^Q)\}_{i\in[N^Q]}, \right)\\ \approx_c &\left((\mathsf{aux}_1, \mathbf{x}), 1^S, X^1, \ldots, X^Q, \{\Delta_i^1\}_{i\in[N^1]}, \ldots, \{\Delta_i^Q\}_{i\in[N^Q]}\right)\end{aligned}$$

holds for $\mathbf{x} \leftarrow \mathcal{X}, \Delta_i^k \leftarrow \{0,1\}^{L^k}$ *for* $k \in [Q], i \in [N^k]$ *and also assume the security of* LprIO *with respect to* Samp. *We then have*

$$\begin{aligned}&\left((\mathsf{aux}_1, \mathsf{aux}_2), X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \ldots, \mathsf{Lobf}_{\mathsf{on}}^Q\right)\\ &\qquad\approx_c \left((\mathsf{aux}_1, \mathbf{x}), X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right),\end{aligned} \tag{5.39}$$

where $(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$, $\mathsf{dig}^k \leftarrow \mathsf{LDigest}(1^\lambda, X^k)$, $\mathsf{Lobf}_{\mathsf{on}}^k \leftarrow \mathsf{LObfOn}(\mathsf{dig}^k, E^k), \delta^k \leftarrow \mathcal{O}_{\mathsf{on}}$ *for* $k \in [Q]$.

Proof. From the assumption, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^S, X^1, \ldots, X^Q, \{E^j(X_i^j)\}_{i\in[N^j],j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^S, X^1, \ldots, X^Q, \{\Delta_i^j\}_{i\in[N^j],j\in[Q]}\right) \tag{5.40}$$

which implies $(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^S, X^1, \ldots, X^Q)$. This further implies

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{\Delta_i^j\}_{i\in[N^j],j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^S, X^1, \ldots, X^Q, \{\Delta_i^j\}_{i\in[N^j],j\in[Q]}) \tag{5.41}$$

since adding independently sampled random terms Δ_i^j does not make the task of distinguishing the distributions easier. Equation (5.40) and Equation (5.41) implies

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{E^j(X_i^j)\}_{i\in[N^j],j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q, \{\Delta_i^j\}_{i\in[N^j],j\in[Q]}).$$

Applying LprIO security definition with respect to Samp, we get

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \ldots, \mathsf{Lobf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right). \tag{5.42}$$

Next, from $(\mathsf{aux}_1, \mathsf{aux}_2, 1^S, X^1, \ldots, X^Q) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^S, X^1, \ldots, X^Q)$ we have

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right), \tag{5.43}$$

since $\mathsf{Lobf}_{\mathsf{off}}$ can be sampled using 1^S and $\{\delta^j\}_{j\in[Q]}$ can be sampled independently.

From Equation (5.42) and Equation (5.43) we deduce

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lobf}_{\mathsf{on}}^1, \ldots, \mathsf{Lobf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, X^1, \ldots, X^Q, \mathsf{Lobf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right).$$

hence the lemma. ∎

The following theorem asserts the security of our construction of PHprFE scheme. This in particular implies that the construction satisfies the security notion as per Definition 5.19.

Theorem 5.24. *The above construction satisfies reusable security as per Definition 5.20.*

Proof. Consider a sampler Samp that generates the following:

1. **Key Queries.** It issues Q_{key} key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$.
2. **Ciphertext Queries.** It issues messages $\mathbf{x}^1 = (\mathbf{x}^1_{\mathsf{pub}}, \mathbf{x}^1_{\mathsf{priv}}), \ldots, \mathbf{x}^{Q_{\mathsf{msg}}} = (\mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, \mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{priv}})$.
3. **Auxiliary Information.** It outputs the auxiliary information aux.

To prove the security as per Definition 5.20, we prove

$$\left(\begin{array}{l}\mathsf{mpk} = \mathsf{prFE.mpk},\ \mathsf{aux},\ \{C^k\}_{k\in[Q_{\mathsf{key}}]},\\ \left\{\mathsf{sk}^k := (\mathbf{r}^k, \mathsf{prFE.sk}^k_0, \mathsf{prFE.sk}^k_1, \mathsf{prFE.sk}^k_2)\right\}_{k\in[Q_{\mathsf{key}}]},\\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \mathsf{prFE.ct}^j\right\}_{j\in[Q_{\mathsf{msg}}]}\end{array}\right) \approx_c \left(\begin{array}{l}\mathsf{mpk} = \mathsf{prFE.mpk},\ \mathsf{aux},\ \{C^k\}_{k\in[Q_{\mathsf{key}}]},\\ \left\{\mathsf{sk}^k := (\mathbf{r}^k, \mathsf{prFE.sk}^k_0, \mathsf{prFE.sk}^k_1, \mathsf{prFE.sk}^k_2)\right\}_{k\in[Q_{\mathsf{key}}]},\\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \delta^j \leftarrow \mathcal{CT}_{\mathsf{prFE}}\right\}_{j\in[Q_{\mathsf{msg}}]}\end{array}\right) \tag{5.44}$$

assuming we have

$$\begin{aligned}&(1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C^k,\ C^k(\mathbf{x}^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})\\ \approx_c\ &(1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C^k,\ \Delta^{j,k} \leftarrow \{0,1\}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]})\end{aligned} \tag{5.45}$$

where

$$\left(\{C^k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}^j = (\mathbf{x}^j_{\mathsf{pub}}, \mathbf{x}^j_{\mathsf{priv}})\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux} \in \{0,1\}^*\right) \leftarrow \mathrm{Samp}(1^\lambda),$$

$$(\mathsf{prFE.mpk}, \mathsf{prFE.msk}) \leftarrow \mathsf{prFE.Setup}(1^\lambda, \mathsf{prm}),$$

$$\mathbf{r}^k \leftarrow \{0,1\}^\lambda,\ \mathsf{prFE.sk}^k_0 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_0[\mathbf{r}^k]),\ \text{for } F_0[\mathbf{r}^k] \text{ as defined in Figure 5.4,}$$

$$\mathsf{prFE.sk}^k_1 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_1[\mathbf{r}^k]),\ \text{for} F_1[\mathbf{r}^k, \mathsf{dig}_{C^k}] \text{ as defined in Figure 5.5}$$

$$\mathsf{dig}_{C^k} \leftarrow \mathsf{LDigest}(\{(i, C^k_i)\}_{i\in|C^k|})$$

$$\mathsf{prFE.sk}^k_2 \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}]),\ \text{for } F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}] \text{ as defined in Figure 5.6,}$$

$$(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S),$$

$$\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}} \leftarrow \mathsf{LDigest}(1^\lambda, \{(i, \mathbf{x}^j_{\mathsf{pub},i})\}_{i\in[L_{\mathsf{pub}}]}), \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2 \leftarrow \{0,1\}^\lambda,$$

$$\mathsf{prFE.ct}^j \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, \mathbf{x}^j_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2)) \qquad \text{for } j \in [Q_{\mathsf{msg}}].$$

We invoke the security of prFE with sampler $\mathsf{Samp}_{\mathsf{prFE}}$ that outputs

$$\begin{pmatrix} \text{Functions:} & \left\{F_0[\mathbf{r}^k],\ F_1[\mathbf{r}^k],\ F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}]\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \text{Inputs:} & \left\{\mathbf{x}^j_{\mathsf{prFE}} := \left(\mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, \mathbf{x}^j_{\mathsf{priv}}, \mathsf{Lst}, \mathsf{sd}^j_0, \mathsf{sd}^j_1, \mathsf{sd}^j_2\right)\right\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{prFE}} := \left(\mathsf{aux},\ \left\{\mathbf{x}^j_{\mathsf{pub}}\right\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k,\}_{k\in[Q_{\mathsf{key}}]}\right) \end{pmatrix}$$

By the security guarantee of prFE with sampler $\mathsf{Samp}_{\mathsf{prFE}}$, Equation (5.44) holds if

$$\begin{pmatrix} \left\{\begin{array}{c} F_0[\mathbf{r}^k](\mathbf{x}^j_{\mathsf{prFE}}) = \left\{\mathsf{lab}^{j,k}_{i,\mathbf{x}^j_{\mathsf{priv},i-L_{\mathsf{pub}}}}\right\}_{i\in[L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}]} \\ F_1[\mathbf{r}^k](\mathbf{x}^j_{\mathsf{prFE}}) = \mathsf{Lobf}^{j,k}_{\mathsf{on},1} \\ F_2[\mathbf{r}^k, \mathsf{dig}_{C^k}](\mathbf{x}^j_{\mathsf{prFE}}) = \mathsf{Lobf}^{j,k}_{\mathsf{on},2} \end{array}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}, \\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q]} \end{pmatrix} \approx_c \begin{pmatrix} \left\{\begin{array}{c} \left\{\delta^{j,k}_{0,i}\right\}_{i\in[L_{\mathsf{priv}}]} \\ \delta^{j,k}_1 \\ \delta^{j,k}_2 \end{array}\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}, \\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \mathsf{Lobf}_{\mathsf{off}},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]} \end{pmatrix}, \tag{5.46}$$

where

$$R^{j,k}_b := \mathsf{PRF}(\mathsf{sd}^j_b, \mathbf{r}^k) \quad \text{for} \quad b = 0,1,2,$$

$$(\mathsf{lab}^{j,k}_{i,0}, \mathsf{lab}^{j,k}_{i,1}) = \mathsf{Garble}_{\mathsf{inp},i}(1^\lambda; R^{j,k}_0) \quad \text{for} \quad i \in [L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}],$$

$$(\mathsf{Lobf}_{\mathsf{off}}, \mathsf{Lst}) \leftarrow \mathsf{LObfOff}(1^\lambda, 1^S)$$

$$\mathsf{Lobf}^{j,k}_{\mathsf{on},1} = \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{\mathbf{x}^j_{\mathsf{pub}}}, E_{\mathsf{pub}}[R^{j,k}_0]; R^{j,k}_1)$$

$$\mathsf{Lobf}^{j,k}_{\mathsf{on},2} = \mathsf{LObfOn}(\mathsf{Lst}, \mathsf{dig}_{C^k}, E_{\mathsf{cir}}[R^{j,k}_0]; R^{j,k}_2) \quad \text{for} \quad j \in [Q_{\mathsf{msg}}],\ k \in [Q_{\mathsf{key}}].$$

Therefore, it suffices to prove Equation (5.46). We observe that it suffices to prove a variant of Equation (5.46) where $R_0^{j,k}, R_1^{j,k}, R_2^{j,k}$ are replaced with independently chosen truly random strings, since (the original version of) Equation (5.46) then follows. This can be seen by the following indistinguishability:

$$\begin{aligned}
&\left\{R_b^{j,k} = \mathsf{PRF}(\mathsf{sd}_b^j, \mathbf{r}^k) : \mathsf{sd}_b^j \leftarrow \{0,1\}^\lambda, \mathbf{r}^k \leftarrow \{0,1\}^\lambda\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]} \\
\approx_s &\left\{R_b^{j,k} = \mathsf{PRF}(\mathsf{sd}_b^j, \mathbf{r}^k) : \mathsf{sd}_b^j \leftarrow \{0,1\}^\lambda, \mathbf{r}^k \leftarrow \{0,1\}^\lambda \backslash \{\mathbf{r}^1, \ldots, \mathbf{r}^{k-1}\}\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]} \\
\approx_c &\left\{R_b^{j,k} \leftarrow \{0,1\}^\lambda\right\}_{b\in\{0,1,2\}, j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]},
\end{aligned}$$

where in the second distribution, $\{\mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}$ is distributed uniformly at random over $(\{0,1\}^\lambda)^{Q_{\mathsf{key}}}$ with the constraint that there is no collision among them. We observe that the first indistinguishability holds since there is a colliding pair in $\{\mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}$ with probability at most $Q_{\mathsf{key}}^2/2^\lambda$ in the first distribution and the second indistinguishability holds by the security of PRF, since each $R_b^{j,k}$ is generated by fresh pair of seed and input. We then invoke the security of LprIO with sampler $\mathsf{Samp}_{\mathsf{LprIO}}$ that outputs

$$\left(\begin{array}{rc}
\text{Inputs:} & \left\{\begin{array}{c} X^{j,k} := \{X_i^{j,k} := (i, C_i^k)\}_{i\in[|C^k|]}, \\ \bar{X}^{j,k} := \{\bar{X}_i^{j,k} := (i, \mathbf{x}_{\mathsf{pub},i}^j)\}_{i\in[L_{\mathsf{pub}}]} \end{array}\right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]} \\
\text{Functions:} & \left\{ E^{j,k} := E_{\mathsf{cir}}[R_0^{j,k}],\ \bar{E}^{j,k} := E_{\mathsf{pub}}[R_0^{j,k}] \right\}_{j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}, \\
\text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{LprIO},1} := \left\{\mathsf{lab}^{j,k}_{i,\mathbf{x}^j_{\mathsf{priv},i-L_{\mathsf{pub}}}}\right\}_{i\in[L_{\mathsf{pub}}+1, L_{\mathsf{pub}}+L_{\mathsf{priv}}], j\in[Q_{\mathsf{msg}}], k\in[Q_{\mathsf{key}}]}, \\
& \mathsf{aux}_{\mathsf{LprIO},2} := \left(\mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k,\ \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]}\right)
\end{array}\right),$$

where we consider $2Q_{\mathsf{msg}}Q_{\mathsf{key}}$ inputs and functions. For simplifying the notations, we use two indices $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$ and consider barred and unbarred symbols to represent different variables instead of using the single index for inputs and functions. In the rest of the proof, we prove

$$\left(\left\{E^{j,k}(X_i^{j,k}) = \tilde{C}_i^{j,k}\right\}_{j,k,i\in[|C^k|]}, \left\{\bar{E}^{j,k}(\bar{X}_i^{j,k}) = \mathsf{lab}^{j,k}_{i,\mathbf{x}^j_{\mathsf{pub},i}}\right\}_{j,k,i\in[L_{\mathsf{pub}}]}, \mathsf{aux}_{\mathsf{LprIO},1}, \mathsf{aux}_{\mathsf{LprIO},2}\right)$$

$$\approx_c \left(\left\{ \gamma_i^{j,k} \right\}_{j,k,i\in[|C^k|]}, \left\{ \bar{\gamma}_i^{j,k} \right\}_{j,k,i\in[L_{\mathsf{pub}}]}, \alpha, \mathsf{aux}_{\mathsf{LprIO},2} \right) \tag{5.47}$$

where $\gamma_i^{j,k} \leftarrow \mathcal{CT}_{\mathsf{bGC}}^i$ for $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}], i \in [|C^k|]$, $\bar{\gamma}_i^{j,k} \leftarrow \{0,1\}^\lambda$ for $j \in [Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}], i \in [L_{\mathsf{pub}}]$, and $\alpha \leftarrow \{0,1\}^{\lambda L_{\mathsf{priv}} Q_{\mathsf{msg}} Q_{\mathsf{key}}}$. Note that the length of α is the same as that of $\mathsf{aux}_{\mathsf{LprIO}}$. Here $\mathcal{CT}_{\mathsf{bGC}}^i$ is the co-domain of Garble_i algorithm. This suffices to conclude the proof, since Equation (5.47) implies Equation (5.46) by the security of LprIO and Theorem 5.23.

To prove Equation (5.47), we introduce the following sequence of hybrids.

Hyb_1. This is the LHS distribution of Equation (5.47). By unrolling the definition of the functions and rearranging the terms, we can see that this is equivalent to the following distribution:

$$\left(\left\{ \left\{ \mathsf{lab}_{i,\mathbf{x}_i^j}^{j,k} \right\}_{i\in[\mathrm{L}]}, \tilde{C}^{j,k} \right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}, \mathsf{aux}, \{\mathbf{x}_{\mathsf{pub}}^j\}_{j\in[Q_{\mathsf{msg}}]}, \{C^k, \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]} \right)$$

where $(\{\mathsf{lab}_{i,b}^{j,k}\}_{i\in[\mathrm{L}],b\in\{0,1\}}, \tilde{C}^{j,k}) \leftarrow \mathsf{Garble}(1^\lambda, C^k; R_0^{j,k})$, $\mathbf{r}^k \leftarrow \{0,1\}^\lambda$ for $j \in [Q_{\mathsf{msg}}]$ and $k \in [Q_{\mathsf{key}}]$, $(\mathsf{aux}, \{\mathbf{x}_{\mathsf{pub}}^j\}_j, \{C^k\}_k) \leftarrow \mathrm{Samp}(1^\lambda)$, and $\mathbf{x}_i^j$ is the i-th bit of $\mathbf{x}^j = (\mathbf{x}_{\mathsf{pub}}^j, \mathbf{x}_{\mathsf{priv}}^j)$. Note that here, we merge the process of generating $\{\tilde{C}_i^{j,k}\}_i$, $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{pub},i}}^{j,k}\}_i$, and $\{\mathsf{lab}_{i,\mathbf{x}_{\mathsf{priv},i}}^{j,k}\}_i$ into a single process of running $\mathsf{Garble}(1^\lambda, C^k; R_0^{j,k})$. This does not change the distribution due to the decomposability of bGC, since the former processes are run on input the common randomness $R_0^{j,k}$ for each j and k.

Hyb_2. This hybrid is same as the previous one except that we compute the labels and the garbled circuits using the simulation algorithm. Namely, the view of the adversary in this hybrid is

$$\left(\left\{ \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]}, \tilde{C}^{j,k} \right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}, \mathsf{aux}, \{\mathbf{x}_{\mathsf{pub}}^j\}_{j\in[Q_{\mathsf{msg}}]}, \{C^k, \mathbf{r}^k\}_{k\in[Q_{\mathsf{key}}]} \right)$$

where we compute $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]}) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, C^k(\mathbf{x}^j))$ for all $j \in$

$[Q_{\mathsf{msg}}], k \in [Q_{\mathsf{key}}]$. Due to the simulation security of bGC, this hybrid is computationally indistinguishable from the previous one.

Hyb_3. This hybrid is same as the previous one except that we input random strings into the simulator of the blind garbled circuit. Namely, we compute $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]})$ as $(\tilde{C}^{j,k}, \{\mathsf{lab}_i^{j,k}\}_{i\in[\mathrm{L}]}) \leftarrow \mathsf{bGC.Sim}(1^\lambda, 1^{\mathrm{L}}, \Delta^{j,k})$, where $\Delta^{j,k} \leftarrow \{0,1\}$ for all j and k. We can see that this hybrid is indistinguishable from the previous one by Equation (5.45).

Hyb_4. This hybrid is same as the previous one except that we replace the output of the simulator for bGC with random strings. By the blindness of bGC, this game is indistinguishable from the previous hybrid.

By rearranging the terms, we can see that the distribution in Hyb_4 is equivalent to that of the RHS of Equation (5.47). We therefore have that the LHS and RHS of Equation (5.47). This completes the proof of Theorem 5.22. ■

5.5.4 Reducing the Dependency on Private Input Length

The sizes of the master public key, ciphertexts, and the secret keys of our construction in Section 5.5.2 are all independent from the length of $\mathbf{x}_{\mathsf{pub}}$ and C. However, they still depend on the length of $\mathbf{x}_{\mathsf{priv}}$. Here, we show a simple conversion that removes this dependency from the sizes of the master public key and secret key. The size of the ciphertext inherently depends on the length of the private input since it should be hidden, but we can make this dependency minimal if we start from the scheme with the ciphertext size being independent of the length of the public input. In particular, the size of the ciphertext only additively depends on the length of $\mathbf{x}_{\mathsf{priv}}$. By applying the conversion in this section to our construction in Section 5.5.2, we obtain a construction of partial-hiding FE with the optimal parameter size.

Building Blocks. We use the following ingredients for our construction.

1. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ with the message space $\{0,1\}^{L_{\mathsf{priv}}}$ with pseudorandom ciphertext space as per Definition 5.1. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE}}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE}}$. Without loss of generality, we assume $\mathcal{K}_{\mathsf{SKE}} = \{0,1\}^{\lambda}$. Furthermore, we assume that the ciphertext space of SKE is $\mathcal{CT}_{\mathsf{SKE}} = \{0,1\}^{L_{\mathsf{priv}}+\lambda}$. Such a construction can be obtained by using PRF for example.

2. A PHprFE scheme $\mathsf{PHprFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn}), \mathsf{Dec})$ whose sizes of the master public key, ciphertext, and secret key are all $\mathrm{poly}(\lambda, L_{\mathsf{priv}})$. We can construct such a scheme assuming LWE and evasive LWE as is shown in Section 5.5. We denote the online part of the ciphertext space by $\mathcal{CT}_{\mathsf{PHprFE}} = \{0,1\}^{\ell^{\mathsf{ct_{on}}}_{\mathsf{PHprFE}}}$.

We describe the new scheme $\mathsf{PHprFE}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}' = (\mathsf{EncOff}', \mathsf{EncOn}'), \mathsf{Dec}')$ for circuit class $C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}$ in the following.

$\mathsf{Setup}'(1^{\lambda}, 1^{L_{\mathsf{pub}}}, 1^{L_{\mathsf{priv}}})$. It does the following.

- Run $\mathsf{Setup}(1^{\lambda}, 1^{L_{\mathsf{pub}}+L_{\mathsf{priv}}+\lambda}, 1^{\lambda}) \to (\mathsf{mpk}, \mathsf{msk})$.
- Output the master public key mpk and the master secret key msk.

$\mathsf{KeyGen}'(\mathsf{msk}, C)$. It does the following.

- Define the circuit C' as follows.
  On input $(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$, output
  $$C'(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct}, \mathsf{SKE.sk}) = C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})).$$
- It runs $\mathsf{KeyGen}(\mathsf{msk}, C') \to \mathsf{sk}_{C'}$ and outputs $\mathsf{sk}_{C'}$.

$\mathsf{Enc}'(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. The encryption algorithm does the following. We divide the algorithm into the following two steps.

$\mathsf{EncOff}'(\mathsf{mpk})$. It runs $\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$ and outputs the offline part of the ciphertext $\mathsf{ct}_{\mathsf{off}}$ and state st.

$\mathsf{EncOn}'(\mathsf{st}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$. It does the following.

- Run $\mathsf{SKE.Setup}(1^\lambda) \to \mathsf{SKE.sk}$.
- Run $\mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{priv}}) \to \mathsf{SKE.ct}$.
- Set $\mathbf{y} := (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and $\mathbf{z} := \mathsf{SKE.sk}$ and run $\mathsf{EncOn}(\mathsf{mpk}, (\mathbf{y}, \mathbf{z})) \to \mathsf{ct}_{\mathsf{on}}$.
- Output $\mathsf{ct}_{\mathsf{on}}' := (\mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$.

The final output of $\mathsf{Enc}'(\mathsf{mpk}, \mathbf{x} = (\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}}))$ is $\mathsf{ct}' := (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$.

$\mathsf{Dec}'(\mathsf{mpk}, \mathbf{x}_{\mathsf{pub}}, \mathsf{sk}_{C'}, C, \mathsf{ct}')$. It does the following.

- Parse $\mathsf{ct}' \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$ and set $\mathbf{y} := (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and $\mathsf{ct} := (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.
- Define C' as in the key generation algorithm.
- Run $\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_C, C', \mathsf{ct}) \to w$ and output w.

Correctness. We make the following observations.

- From the perfect correctness of underlying PHprFE scheme, with public input $\mathbf{y} = (\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct})$ and private inpute $\mathsf{SKE.sk}$, we have

$$\begin{aligned}\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_{C'}, C', \mathsf{ct}) &= C'(\mathbf{y}, \mathsf{SKE.sk}) \\ &= C'(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.ct}, \mathsf{SKE.sk}) \\ &= C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})) \quad (\text{ by definition of } C')\end{aligned}$$

- Next, from the perfect correctness of the SKE scheme, we have

$$\mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}) = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_{\mathsf{priv}})) = \mathbf{x}_{\mathsf{priv}}$$

Thus $\mathsf{Dec}(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}_{C'}, C', \mathsf{ct}) = C(\mathbf{x}_{\mathsf{pub}}, \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct})) = C(\mathbf{x}_{\mathsf{pub}}, \mathbf{x}_{\mathsf{priv}})$ and hence the correctness.

Efficiency. Here, we show $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}_{C'}| = \mathrm{poly}(\lambda)$, $|\mathsf{ct}'| = \mathrm{poly}(\lambda) + L_{\mathsf{priv}}$.

Given that the length of the private input for the underlying PHprFE is λ, it is straightforward to see the first two equations above hold by the efficiency of PHprFE. To bound the length of the ciphertext ct', we first observe that $\mathsf{ct}' = (\mathsf{ct}_{\mathsf{off}}, \mathsf{SKE.ct}, \mathsf{ct}_{\mathsf{on}})$, where $\mathsf{ct} \coloneqq (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$ constitutes a ciphertext of the underlying PHprFE encrypting $(\mathbf{y}, \mathbf{z})$. We have $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{z}|) = \mathrm{poly}(\lambda)$ and $|\mathsf{SKE.ct}| = \mathrm{poly}(\lambda) + L_{\mathsf{priv}}$. Therefore, the last equation above follows as well.

We therefore have the following theorem. The security of the scheme is proven in Theorem 5.27.

Theorem 5.25. *Assuming LWE and evasive LWE assumptions, there exists a partially hiding pseudorandom FE scheme, for circuit class* $C = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}\}$*, that satisfies reusable security (as per Definition 5.20) whose sizes of the master public key and the secret key are fixed polynomial* $\mathrm{poly}(\lambda)$*. Furthermore, the size of the ciphertext is* $L_{\mathsf{priv}} + \mathrm{poly}(\lambda)$.

Optimal prFE. Using Remark 24, we get a prFE scheme as a special case of PHprFE, for $\mathbf{x}_{\mathsf{pub}} = \bot$ and $\mathbf{x}_{\mathsf{priv}} = \mathbf{x}$. Next, observing the fact that (1) with $\mathbf{x}_{\mathsf{pub}} = \bot$ the security of PHprFE (Definition 5.19) is equivalent to single-challenge security of prFE (Definition 5.13, with $Q_{\mathsf{msg}} = 1$) and (2) Definition 5.19 is implied by Definition 5.20, we get a prFE scheme with optimal parameters. We formalise this in the following theorem.

Theorem 5.26. *Assuming LWE and evasive LWE assumptions, there exists a* prFE *scheme, for circuit class* $C = \{C : \{0,1\}^{L_{\mathsf{inp}}} \to \{0,1\}\}$*, that satisfies security (as per Definition 5.13, with* $Q_{\mathsf{msg}} = 1$*) and efficiency* $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda)$.

Security The following theorem asserts the security of the construction.

Theorem 5.27. *The above construction* PHprFE' *satisfies reusable security as per Definition 5.20 if so does* PHprFE *and* SKE *is secure as per Definition 5.1.*

Proof. Consider a sampler $\mathsf{Samp}_{\mathsf{PHprFE}'}$ that generates the following:

1. **Key Queries.** It issues Q_{key} key queries $C_1, \ldots, C_{Q_{\mathsf{key}}}$.
2. **Ciphertext Queries.** It issues messages $\mathbf{x}^1 = (\mathbf{x}^1_{\mathsf{pub}}, \mathbf{x}^1_{\mathsf{priv}}), \ldots, \mathbf{x}^{Q_{\mathsf{msg}}} = (\mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{pub}}, \mathbf{x}^{Q_{\mathsf{msg}}}_{\mathsf{priv}})$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 5.20, we prove

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{msg}}]}, \\ \left\{\mathsf{sk}_k \coloneqq \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{msg}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ \mathsf{ct}^j_{\mathsf{on}}\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{msg}}]}, \\ \left\{\mathsf{sk}_k \coloneqq \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{msg}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \gamma^j,\ \delta^j\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{5.48}$$

where $\gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}}$ and $\delta^j \leftarrow \mathcal{CT}_{\mathsf{PHprFE}}$ for $j \in [Q_{\mathsf{msg}}]$, assuming we have

$$(1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C_k,\ C_k(\mathbf{x}^j)\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \approx_c (1^\lambda,\ \mathsf{aux},\ \{\mathbf{x}^j_{\mathsf{pub}},\ C_k,\ \Delta^j_k \leftarrow \{0,1\}\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}) \tag{5.49}$$

where

$$\left(\{C_k\}_{k\in[Q_{\mathsf{key}}]}, \{\mathbf{x}^j = (\mathbf{x}^j_{\mathsf{pub}}, \mathbf{x}^j_{\mathsf{priv}})\}_{j\in[Q_{\mathsf{msg}}]}, \mathsf{aux} \in \{0,1\}^*\right) \leftarrow \mathsf{Samp}_{\mathsf{PHprFE}'}(1^\lambda),$$

$$\mathsf{SKE.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda), \quad \mathsf{SKE.ct}^j \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}^j_{\mathsf{priv}}),$$

$$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{L_{\mathsf{pub}}+L_{\mathsf{priv}}+\lambda}, 1^\lambda),$$

$\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{msk}, C'_k)$ for $k \in [Q_{\mathsf{key}}]$, where C'_k is defined from C_k as in the construction,

$$(\mathsf{ct}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{EncOff}(\mathsf{mpk}),$$

$$\mathsf{ct}_{\mathsf{on}}{}^j \leftarrow \mathsf{EncOn}(\mathsf{st}, \mathbf{x}^j_{\mathsf{priv}}) \text{ for } j \in [Q_{\mathsf{msg}}].$$

We invoke the security of PHprFE with sampler $\mathsf{Samp}_{\mathsf{PHprFE}}$ that outputs

$$\begin{pmatrix} \text{Functions:} & \{C'_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \text{Inputs:} & \left\{X_j := \left(X^j_{\mathsf{pub}} := (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), X^j_{\mathsf{priv}} := \mathsf{SKE.sk}^j\right)\right\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathsf{prFE}} := \left(\mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}\right) \end{pmatrix}$$

By the security guarantee of PHprFE with sampler $\mathsf{Samp}_{\mathsf{PHprFE}}$,

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}}, \left\{X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), \mathsf{ct}^j_{\mathsf{on}}\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}}, \left\{X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), \delta^j\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{5.50}$$

holds if

$$\left(\mathsf{aux}, \left\{C_k,\ X^j_{\mathsf{pub}},\ C'_k(X^j)\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right) \approx_c \left(\mathsf{aux}, \left\{C_k,\ X^j_{\mathsf{pub}},\ \Delta^j_k\right\}_{j\in[Q_{\mathsf{msg}}],k\in[Q_{\mathsf{key}}]}\right). \tag{5.51}$$

We first observe that Equation (5.50) implies Equation (5.48), since we can invoke the security of SKE to conclude that

$$\begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \mathsf{SKE.ct}^j,\ \delta^j\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{mpk},\ \mathsf{aux},\ \{C_k\}_{k\in[Q_{\mathsf{key}}]}, \\ \left\{\mathsf{sk}_k := \mathsf{sk}_{C'_k}\right\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{Lobf}_{\mathsf{off}},\ \left\{\mathbf{x}^j_{\mathsf{pub}},\ \gamma^j,\ \delta^j\right\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix}$$

holds by noting that $\mathsf{SKE.sk}^j$ is used only for computing $\mathsf{SKE.ct}^j$ and not used anywhere else.

Therefore, it suffices to prove Equation (5.51) to conclude the proof. We have

$$\left(\mathsf{aux}, \left\{C_k, X^j_{\mathsf{pub}} = (\mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j), C'_k(X^j)\right\}_{j,k}\right) = \left(\mathsf{aux}, \left\{C_k, \mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j, C_k(\mathbf{x}^j)\right\}_{j,k}\right)$$
$$\approx_c \left(\mathsf{aux}, \left\{C_k,\ \mathbf{x}^j_{\mathsf{pub}},\ \gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ C_k(\mathbf{x}^j)\right\}_{j,k}\right)$$

$$\approx_c \left(\mathsf{aux}, \left\{C_k, \mathbf{x}^j_{\mathsf{pub}}, \gamma^j \leftarrow \mathcal{CT}_{\mathsf{SKE}}, \Delta^j_k \leftarrow \{0,1\}\right\}_{j,k}\right)$$
$$\approx_c \left(\mathsf{aux}, \left\{C_k, \mathbf{x}^j_{\mathsf{pub}}, \mathsf{SKE.ct}^j, \Delta^j_k \leftarrow \{0,1\}\right\}_{j,k}\right)$$

where the first line follows from the definition of C'_k and X^j, the second from the security of SKE noting that $\mathsf{SKE.sk}^j$ is used only for computing $\mathsf{SKE.ct}^j$ and not used anywhere else, the third from Equation (5.49), noting that adding random string $\{\gamma^j\}_j$ to the distributions in Equation (5.49) does not make the task of distinguishing the distributions any easier, and the fourth from the security of SKE again. This proves Equation (5.51) and therefore completes the proof. ■

5.5.5 Handling Longer Output

So far we have only considered the case where C is a circuit that outputs a single-bit string. Here, we discuss more general case where the output of the circuit C is longer. To handle such circuits in the construction, we only change the key generation algorithm. To generate a secret key for $C : \{0,1\}^{\mathsf{L}} \to \{0,1\}^{\mathsf{out}}$, we first consider circuits $\{C_j\}_{j\in[\mathsf{out}]}$, where C_j is the circuit that outputs the j-th bit of C's output and then generate secret keys for $\{C_j\}_j$. It is then easy to see that the all bits of $C(\mathbf{x})$ can be recovered by using the secret keys for $\{C_j\}_j$ by decrypting a ciphertext encrypting $\mathbf{x}$. This does not change the size of the master public key and ciphertext, but makes the secret key linearly dependent on the output length of C. Thus, we get the following theorems.

Theorem 5.28. *Assuming LWE and evasive LWE assumptions, there exists a* PHprFE *scheme, for circuit class* $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{pub}}} \times \{0,1\}^{L_{\mathsf{priv}}} \to \{0,1\}^{\mathsf{out}}\}$*, that satisfies reusable security (as per Definition 5.20) and efficiency*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = \mathsf{out} \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = L_{\mathsf{priv}} + \mathrm{poly}(\lambda).$$

Theorem 5.29. *Assuming LWE and evasive LWE assumptions, there exists a* prFE *scheme, for circuit class* $\mathcal{C} = \{C : \{0,1\}^{L_{\mathsf{inp}}} \to \{0,1\}\}$*, that satisfies security (as per*

Definition 5.13, with $Q_{\mathsf{msg}} = 1$) and efficiency

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = \mathsf{out} \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda).$$

5.6 KP-ABE/PE FOR UNBOUNDED DEPTH CIRCUITS WITH OPTIMAL PARAMETERS

Here we construct KP-ABE and KP-PE schemes supporting unbounded depth circuits and achieving optimal parameters.

5.6.1 Construction of KP-ABE with Optimal Parameters

In this section we construct an ABE scheme $\mathsf{kpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\{0,1\}$ and circuit family $\mathcal{C}_\ell$ consisting of circuits with input space $\{0,1\}^{\mathrm{L}}$ and output space $\{0,1\}$. For the purpose of application in Section 5.7, we consider a decomposed encryption algorithm as $\mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn})$ in our construction where $\mathsf{EncOff}(\mathsf{mpk}) \to (\mathsf{ct}_{\mathsf{off}}, \mathsf{st})$, $\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu) \to \mathsf{ct}_{\mathsf{on}}$ and Enc outputs $(\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.

Building Blocks. We require the following building blocks for our construction.

1. A PHprFE scheme $\mathsf{PHprFE} = \mathsf{PHprFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc} = (\mathsf{EncOff}, \mathsf{EncOn}), \mathsf{Dec})$ from Section 5.5.4 for circuit family $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to \mathcal{Y}\}\}_{\mathsf{prm}}$ where $\mathcal{X}_{\mathsf{pub}} = \{0,1\}^{\ell}, \mathcal{X}_{\mathsf{priv}} = \{0,1\}^{\lambda+1}$ and $\mathcal{Y} = \{0,1\}$. We denote the online part of the ciphertext space by $\mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}}$. We require the PHprFE scheme to satisfy reusable security (Definition 5.20). As we show in Theorem 5.25, such a PHprFE scheme can be constructed assuming LWE and evasive LWE assumptions.

2. A PRF scheme $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}$. It is known that PRF can be constructed from one-way functions.

Now, we describe our construction.

$\mathsf{Setup}(1^{\lambda}, 1^{\mathrm{L}})$. The setup algorithm does the following.

- Generate $(\mathsf{PHprFE.msk}, \mathsf{PHprFE.mpk}) \leftarrow \mathsf{PHprFE.Setup}(1^{\lambda}, (1^{\ell}, 1^{\lambda+1}))$.

– Output $\mathsf{msk} = \mathsf{PHprFE.msk}$ and $\mathsf{mpk} = \mathsf{PHprFE.mpk}$.

$\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm does the following.

– Parse $\mathsf{msk} = \mathsf{PHprFE.msk}$.

– Sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and define the circuit $C[\mathbf{r}]$, with $\mathbf{r}$ hardwired, as follows. On input $(\mathbf{x}, \mu, \mathsf{sd})$,

$$C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \begin{cases} \mu & \text{if } C(\mathbf{x}) = 0 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}$$

– Compute $\mathsf{PHprFE.sk} \leftarrow \mathsf{PHprFE.KeyGen}(\mathsf{PHprFE.msk}, C[\mathbf{r}])$.

– Output $\mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$. The encryption algorithm works as follows.

$\mathsf{EncOff}(\mathsf{mpk})$. The offline phase of encryption does the following.

– Parse $\mathsf{mpk} = \mathsf{PHprFE.mpk}$.

– Compute $(\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.st}) \leftarrow \mathsf{PHprFE.EncOff}(\mathsf{PHprFE.mpk})$.

– Output $\mathsf{ct}_{\mathsf{off}} = \mathsf{PHprFE.ct}_{\mathsf{off}}$ and $\mathsf{st} = \mathsf{PHprFE.st}$.

$\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu)$. The online phase of encryption does the following.

– Parse $\mathsf{st} = \mathsf{PHprFE.st}$.

– Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, set $X_{\mathsf{pub}} = \mathbf{x}$ and $X_{\mathsf{priv}} = (\mu, \mathsf{sd})$.

– Compute $\mathsf{PHprFE.ct}_{\mathsf{on}} \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.st}, X_{\mathsf{pub}}, X_{\mathsf{priv}})$.

– Output $\mathsf{ct}_{\mathsf{on}} \coloneqq \mathsf{PHprFE.ct}_{\mathsf{on}}$.

Output $\mathsf{ct} \coloneqq (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}, \mathbf{x})$. The decryption algorithm does the following.

– Parse $\mathsf{mpk} = \mathsf{PHprFE.mpk}, \mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}}) = (\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.ct}_{\mathsf{on}})$.

– Compute $y = \mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \mathbf{x}, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct})$, where $C[\mathbf{r}]$ is as defined in the key generation algorithm.

– Output y.

Correctness. For $\mathsf{sk}_C = (\mathsf{PHprFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}}) = (\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.ct}_{\mathsf{on}}) = \mathsf{PHprFE.ct}$, we have

$$\mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \mathbf{x}, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct}) = C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \begin{cases} \mu & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}$$

from the correctness of PHprFE scheme. Now, if $C(\mathbf{x}) = 1$, then from the definition of $C[\mathbf{r}]$, we get $C[\mathbf{r}](\mathbf{x}, \mu, \mathsf{sd}) = \mu$ and hence the decryption outputs $y = \mu$ correctly.

Efficiency. Instantiating the PHprFE scheme from Section 5.5.4 with $|\mathsf{PHprFE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{PHprFE.sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{PHprFE.ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}_{\mathsf{priv}}|$, our kpABE scheme satisfies $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda)$. We formalise this instantiation using the following theorem. The security is proved in Section 5.6.2.

Theorem 5.30. *Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-ABE scheme supporting circuits* $\{C : \{0,1\}^{\ell} \to \{0,1\}\}$ *with unbounded depth and single bit message space with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda). \tag{5.52}$$

5.6.2 Security of KP-ABE

For our application in Section 5.7, we introduce the following security notion.

Definition 5.21 (VerSel-INDr Reusable Security)**.** A kpABE scheme for circuit family

$C_\ell = \{C : \{0,1\}^\ell \to \{0,1\}\}$ is said to satisfy VerSel-IND reusable security if for all stateful PPT adversary $\mathcal{A}$, the following holds

$$\Pr\left[\beta' = \beta : \begin{array}{l} (\mathsf{aux}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}, \mu) \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); \\ (\mathsf{ct}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{EncOff}(\mathsf{mpk}); \\ \{\mathsf{ct}^j_{\mathsf{on},0} \leftarrow \mathsf{EncOn}(\mathsf{st}, \mathbf{x}^j, \mu), \mathsf{ct}^j_{\mathsf{on},1} \leftarrow \mathcal{CT}_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]}, \beta \leftarrow \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(\mathsf{aux}_{\mathcal{A}}, \mathsf{mpk}, \{C^k, \mathsf{sk}_{C^k}\}_{k\in Q_{\mathsf{key}}}, \mathsf{ct}_{\mathsf{off}}, \{\mathbf{x}^j, \mathsf{ct}^j_{\mathsf{on},\beta}\}_{j\in[Q_{\mathsf{msg}}]}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{CT}_{\mathsf{on}}$ is the ciphertext space of EncOn. We require that for all key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$ and challenge attribute queries $\mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}$ we have $C^k(\mathbf{x}^j) = 0$.

Remark 28. We note that the above security definition implies more standard VerSel-IND security for ABE. This can be seen by considering the case of $Q_{\mathsf{msg}} = 1$ and recalling that $\mathsf{ct} = (\mathsf{ct}_{\mathsf{off}}, \mathsf{ct}_{\mathsf{on}})$. The above security definition in this special case implies that the message carrying part of the ciphertext is pseudorandom in VerSel-IND security game. This immediately implies VerSel-IND security.

We prove the above security of our scheme using the following theorem.

Theorem 5.31. *Assume the* PHprFE *scheme satisfies reusable security (Definition 5.20) w.r.t. the sampler class containing the sampler* Samp *as defined in Eq. 5.54 and* PRF *is secure. Then the above construction of* kpABE *scheme is secure (Definition 5.21).*

Proof. Consider a PPT adversary $\mathcal{A}$ that outputs $\mathsf{coins}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}, \mu$. To prove reusable security as per Definition 5.21, we show

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = \mathsf{PHprFE.mpk}, \\ \{\mathsf{sk}^k = (\mathsf{PHprFE.sk}^k, \mathbf{r}^k), C^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}}, \{\mathbf{x}^j, \mathsf{PHprFE.ct}^j_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = \mathsf{PHprFE.mpk}, \\ \{\mathsf{sk}^k = (\mathsf{PHprFE.sk}^k, \mathbf{r}^k), C^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{\mathbf{x}^j, \delta^j \leftarrow \mathsf{PHprFE.}\mathcal{CT}_{\mathsf{on}}\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{5.53}$$

Also for all the key queries $C^1, \ldots, C^{Q_{\mathsf{key}}}$ and challenge attribute queries $\mathbf{x}^1, \ldots, \mathbf{x}^{Q_{\mathsf{msg}}}$

issued by the adversary, we have $C^k(\mathbf{x}^j) = 0$.

We invoke the security of PHprFE scheme with sampler Samp that outputs

$$\begin{pmatrix} \text{circuits:} & \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \text{Inputs:} & \{X^j_{\mathsf{pub}} = \mathbf{x}^j, X^j_{\mathsf{priv}} = (\mu, \mathsf{sd}^j)\}_{j\in[Q_{\mathsf{msg}}]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}} = (\mathsf{coins}_{\mathcal{A}}, C^1, \ldots, C^{Q_{\mathsf{key}}}, \mathbf{r}^1, \ldots, \mathbf{r}^{Q_{\mathsf{key}}}, \mathsf{coins}_{\mathcal{A}}) \end{pmatrix} \tag{5.54}$$

Using the guarantee of PHprFE scheme with sampler Samp we have that

$$\begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{PHprFE.mpk},\ \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{PHprFE.sk}^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{X^j_{\mathsf{pub}}, \mathsf{PHprFE.ct}_{\mathsf{on}}{}^j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{aux}_{\mathcal{A}},\ \mathsf{PHprFE.mpk},\ \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]}, \\ \{\mathsf{PHprFE.sk}^k\}_{k\in[Q_{\mathsf{key}}]}, \\ \mathsf{PHprFE.ct}_{\mathsf{off}},\ \{X^j_{\mathsf{pub}}, \delta^j\}_{j\in[Q_{\mathsf{msg}}]} \end{pmatrix} \tag{5.55}$$

$$\begin{aligned} \text{if} \quad & \left(\mathsf{aux}_{\mathcal{A}}, \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]},\ \{X^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{C^k[\mathbf{r}^k](X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}})\}_{k\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\right) \\ & \approx_c \left(\mathsf{aux}_{\mathcal{A}}, \{C^k[\mathbf{r}^k]\}_{k\in[Q_{\mathsf{key}}]},\ \{X^j_{\mathsf{pub}}\}_{j\in[Q_{\mathsf{msg}}]},\ \{\Delta_{k,j} \leftarrow \{0,1\}^\lambda\}_{k\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\right) \end{aligned} \tag{5.56}$$

where $X^j_{\mathsf{pub}} = \mathbf{x}^j, X^j_{\mathsf{priv}} = (\mu, \mathsf{sd}^j)$ for $\mathsf{sd}^j \leftarrow \{0,1\}^\lambda$, $j \in [Q_{\mathsf{msg}}]$,

$(\mathsf{PHprFE.mpk}, \mathsf{PHprFE.msk}) \leftarrow \mathsf{PHprFE.Setup}(1^\lambda, (1^\ell, 1^{\lambda+1}))$,

$\mathsf{PHprFE.sk}^k \leftarrow \mathsf{PHprFE.KeyGen}(\mathsf{PHprFE.msk}, C^k[\mathbf{r}^k])$, for $\mathbf{r}^k \leftarrow \{0,1\}^\lambda$,

$(\mathsf{PHprFE.ct}_{\mathsf{off}}, \mathsf{PHprFE.st}) \leftarrow \mathsf{PHprFE.EncOff}(\mathsf{PHprFE.mpk})$,

$\mathsf{PHprFE.ct}_{\mathsf{on}}{}^j \leftarrow \mathsf{PHprFE.Enc}(\mathsf{PHprFE.st}, X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}})$,

$\delta^j \leftarrow \mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}}$ for $j \in [Q_{\mathsf{msg}}]$ where $\mathsf{PHprFE}.\mathcal{CT}_{\mathsf{on}}$ is the ciphertext space of $\mathsf{PHprFE.EncOn}$.

First we note that rearranging the terms of Equation (5.55), it is same as distribution in Equation (5.53). Thus, to prove Equation (5.53), it suffices to prove Equation (5.56). Equation (5.56) holds from the security of the underlying PRF scheme. To see this note

that

$$C^k[\mathbf{r}^k](X^j_{\mathsf{pub}}, X^j_{\mathsf{priv}}) = C^k[\mathbf{r}^k](\mathbf{x}^j, \mu, \mathsf{sd}^j) = \begin{cases} \mu & \text{if } C^k(\mathbf{x}^j) = 1 \\ \mathsf{PRF}(\mathsf{sd}^j, \mathbf{r}^k) & \text{otherwise.} \end{cases}$$

By the admissibility of the adversary into the kpABE security game, we have $C^k(\mathbf{x}^j) = 0$ for all $k \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$. So, we get

$$\begin{aligned} C^k[\mathbf{r}^k](\mathbf{x}^j, \mu, \mathsf{sd}^j) &= \mathsf{PRF}(\mathsf{sd}^j, \mathbf{r}^k) \\ &\approx_c \ \Delta_{j,k} \leftarrow \{0,1\} \end{aligned}$$

where the last equation follows from the security of the PRF scheme. ■

5.6.3 Predicate Encryption with Optimal Parameters

In this section we sketch out the construction of a PE scheme $\mathsf{PE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ supporting unbounded depth circuits and achieving optimal parameters. The construction is same as that in Section 5.6.1 with the following changes.

1. In the building blocks we use a PHprFE scheme with $\mathcal{X}_{\mathsf{pub}} = \{\perp\}, \mathcal{X}_{\mathsf{priv}} = \{0,1\}^{\ell+\lambda+1}$ and $\mathcal{Y} = \{0,1\}$.

2. In $\mathsf{KeyGen}(\mathsf{msk}, C)$ algorithm the circuit $C[\mathbf{r}]$ is defined as follows.
   On input $(\perp, \mathbf{x}, \mu, \mathsf{sd})$,

$$C[\mathbf{r}](\perp, \mathbf{x}, \mu, \mathsf{sd}) = \begin{cases} \mu & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}) & \text{otherwise.} \end{cases}$$

3. In EncOn algorithm we set $\mathbf{x}_{\mathsf{pub}} = \perp$ and $\mathbf{x}_{\mathsf{priv}} = (\mathbf{x}, \mu, \mathsf{sd})$.

4. In $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct})$[7], we compute $y = \mathsf{PHprFE.Dec}(\mathsf{PHprFE.mpk}, \perp, \mathsf{PHprFE.sk}, C[\mathbf{r}], \mathsf{PHprFE.ct})$.

It is easy to see that the above PE scheme satisfies correctness. The PE scheme also satisfies VerSel-INDr reusable security as defined in Definition 5.21 with a similar security proof as in Section 5.6.2 with the above specified changes. Note that since the

[7] Here we do not give $\mathbf{x}$ as an input.

online part of the encryption $\mathsf{EncOn}(\mathsf{st}, \mathbf{x}, \mu)$ encodes both the attribute and the message μ, replacing it with a random string hides the information of $\mathbf{x}$ and μ and thus it satisfies the security requirement for a PE scheme. As for the efficiency, since we encode $\mathbf{x}$ into the private part of the input, the ciphertext size is $|\mathbf{x}| + \mathrm{poly}(\lambda)$. Summarizing the above discussion, we have the following theorem.

Theorem 5.32. *Under the LWE and Evasive LWE assumption, there exists very selectively secure KP-PE scheme supporting circuits* $\{C : \{0,1\}^\ell \to \{0,1\}\}$ *with unbounded depth with and single bit message space with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_C| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda) + |\mathbf{x}|. \tag{5.57}$$

where $\mathbf{x} \in \{0,1\}^\ell$.

5.6.4 Extending the Message Space

Our construction of ABE and PE only allows us to encrypt a single-bit message. Here, we explain how to extend the message space while maintaining the optimal parameter size. In both cases, it suffices to encrypt message of length λ, since this allows us to employ the hybrid encryption approach, where we encrypt a secret key $\mathsf{SKE.sk} \in \{0,1\}^\lambda$ of an SKE scheme and then use this to encrypt the message.

In the case of ABE, we simply encrypt each bit of $\mathsf{SKE.sk}$, which blows up the ciphertext size by a factor of λ, but this still results in the optimal parameter size of Equation (5.52). In the case of PE, this approach leads to a ciphertext of size $\mathrm{poly}(\lambda) + \lambda|\mathbf{x}|$, which ruins the optimal ciphertext size of only additively depending on the length of the attribute. Instead, we change the construction of PE from PHprFE by setting $\mathbf{x}_{\mathsf{pub}} = \bot$ and $\mathbf{x}_{\mathsf{priv}} = (\mathbf{x}, \mathsf{SKE.sk}, \mathsf{sd})$ and considering a circuit $C[i, \mathbf{r}]$ for $i \in [\lambda]$ that is defined as

$$C[i,\mathbf{r}](\bot, \mathbf{x}, \mathsf{SKE.sk}, \mathsf{sd}) = \begin{cases} \mathsf{SKE.sk}_i & \text{if } C(\mathbf{x}) = 1 \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_i) & \text{otherwise} \end{cases},$$

where $\mathsf{SKE.sk}_i$ is the i-th bit of $\mathsf{SKE.sk}$. For generating a secret key of PE for circuit

C, we generate PHprFE secret keys for $C[i, \mathbf{r}_i]$ for all $i \in [\lambda]$ with freshly chosen $\mathbf{r}_i$. This allows the decryptor to recover $\mathsf{SKE.sk}$ in a bit-by-bit manner. It is not difficult to see that the construction achieves optimal parameter size of Equation (5.57) and still maintains the security.

5.7 COMPILING KP-ABE TO CP-ABE USING prFE

In this section we give a compiler that converts kpABE scheme for circuits of unbounded depth to a cpABE scheme for circuits of unbounded depth using a prFE scheme for pseudorandom functionality. In particular, if we start from a kpABE scheme with optimal parameter size, the resulting cpABE scheme achieves the optimal parameter size as well. By combining the kpABE and cpABE, we obtain an ABE scheme for Turing machines from LWE and evasive LWE. This improves [13] in terms of the assumption, which requires the non-standard tensor circular LWE assumption additionally.

5.7.1 Construction

Building Blocks. We require the following building blocks for our construction.

1. A key-policy ABE scheme $\mathsf{kpABE} = \mathsf{kpABE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EncOff}, \mathsf{EncOn}, \mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathrm{L}(\lambda)} = \{C : \{0,1\}^{\ell} \to \{0,1\}\}$ consisting of circuits with input length $\mathrm{L}(\lambda)$ that satisfies $\mathsf{VerSel\text{-}INDr}$ reusable security (Definition 5.21). We require that the scheme satisfies optimal parameter size, namely, the size of the master public key, secret keys, and ciphertexts are all fixed polynomial. We denote the online ciphertext space of kpABE scheme by $\mathcal{CT}_{\mathsf{on}} := \{0,1\}^{\ell_{\mathsf{on}}}$, online ciphertext size by ℓ_{on}. We also assume that the randomness used by $\mathsf{kpABE.EncOn}$ is of length λ without loss of generality. If it requires longer randomness, we can derive it by a PRF. We can instantiate such kpABE scheme by our construction in Section 5.6.1.

2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE.Setup}, \mathsf{prFE.KeyGen}, \mathsf{prFE.Enc}, \mathsf{prFE.Dec})$ for circuit class $\mathcal{C} = \{C : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell_{\mathsf{on}}}\}$. Here, the depth of the circuit is unbounded, but the input and the output lengths are fixed. For our compiler, we set $\mathrm{L}(\lambda) = \mathrm{poly}(\lambda)$ for some fixed polynomial $\mathrm{poly}(\cdot)$. We require the size of the master public key and the ciphertext to be fixed polynomial in λ and the size of the secret key to be $\ell_{\mathsf{on}} \cdot \mathrm{poly}(\lambda)$. Such a construction can be obtained by applying the conversion described in Section 5.5.5 to our construction of PHprFE in Section 5.5.4. We use $\mathcal{CT}_{\mathsf{prFE}}$ to denote the ciphertext space of the scheme and

$\mathsf{prm} = (1^{\mathrm{L}}, 1^{\ell_{\mathsf{on}}})$ to denote the parameters describing the circuit class.

3. A pseudorandom function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$. It is known that PRF can be constructed from one-way functions.

Now, we describe our compiler for constructing a ciphertext-policy ABE scheme $\mathsf{cpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuits of unbounded depth with attribute length L.

$\mathsf{Setup}(1^{\lambda}, 1^{\mathrm{L}}) \to (\mathsf{cpABE.mpk}, \mathsf{cpABE.msk})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.mpk}, \mathsf{prFE.msk}) \leftarrow \mathsf{prFE.Setup}(1^{\lambda}, \mathsf{prm})$.
- Set $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}$ and $\mathsf{cpABE.msk} = \mathsf{prFE.msk}$. Output $(\mathsf{cpABE.mpk}, \mathsf{cpABE.msk})$.

$\mathsf{KeyGen}(\mathsf{cpABE.msk}, \mathbf{x}) \to \mathsf{cpABE.sk}_{\mathbf{x}}$. The key generation algorithm does the following.

- Parse $\mathsf{cpABE.msk} = \mathsf{prFE.msk}$ and sample $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$.
- Define circuit $\mathrm{F}[\mathbf{x}, \mathbf{r}]$, with $\mathbf{x}, \mathbf{r}$ hardwired, as follows.
  On input $(\mathsf{kpABE.st}, \mathsf{sd}, \mu)$:
  - Compute and output $\mathsf{kpABE.ct}_{\mathsf{on}}$
    where $\mathsf{kpABE.ct}_{\mathsf{on}} = \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
- Compute $\mathsf{prFE.sk} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathbf{x}, \mathbf{r}])$.
- Output $\mathsf{cpABE.sk}_{\mathbf{x}} \coloneqq (\mathbf{r}, \mathsf{prFE.sk})$.

$\mathsf{Enc}(\mathsf{cpABE.mpk}, C, \mu) \to \mathsf{cpABE.ct}$. The encryption algorithm does the following.

- Parse $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^{\lambda}$.
- Generate $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^{\lambda}, 1^{\mathrm{L}})$.
- Generate $(\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu))$.

- Compute $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.
- Output $\mathsf{cpABE.ct} := (\mathsf{prFE.ct},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \mathsf{kpABE.sk}_C)$.

$\mathsf{Dec}(\mathsf{cpABE.mpk}, \mathsf{cpABE.sk}_{\mathbf{x}}, \mathbf{x}, \mathsf{cpABE.ct}, C)$. The decryption algorithm does the following.

- Parse $\mathsf{cpABE.mpk} = \mathsf{prFE.mpk}$, $\mathsf{cpABE.sk}_{\mathbf{x}} = \mathsf{prFE.sk}$, and $\mathsf{cpABE.ct} = (\mathsf{prFE.ct},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \mathsf{kpABE.sk}_C)$.
- Compute $\mathbf{y} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}, F[\mathbf{x}, \mathbf{r}], \mathsf{prFE.ct})$.
- Compute and output $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, (\mathsf{kpABE.ct}_{\mathsf{off}}, \mathbf{y}), \mathbf{x})$.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 5.33. *Assume* kpABE *is perfectly correct. Then the above construction of* cpABE *scheme is correct.*

Proof. From the correctness of prFE scheme, with probability 1 we have

$$\mathbf{y} = \mathrm{F}[\mathbf{x}, \mathbf{r}](\mathsf{kpABE.st}, \mathsf{sd}, \mu) = \mathsf{kpABE.ct}_{\mathsf{on}}$$

where $\mathsf{kpABE.ct}_{\mathsf{on}} = \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$. Next, from the correctness of kpABE, if $C(\mathbf{x}) = 1$, it follows that $\mathsf{kpABE.Dec}(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, C, \mathsf{kpABE.ct}, \mathbf{x}) = \mu$, where $\mathsf{kpABE.ct} = (\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.ct}_{\mathsf{on}})$. ∎

Efficiency. Our cpABE scheme satisfies

$$|\mathsf{cpABE.mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{cpABE.sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda),\ |\mathsf{cpABE.ct}| = \mathrm{poly}(\lambda).$$

To see the above we make the following observations:

1. Instantiating kpABE scheme as in Section 5.6.1, we have $|\mathsf{kpABE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{kpABE.sk}_C| = \mathrm{poly}(\lambda)$, $|\mathsf{kpABE.ct}_{\mathsf{off}}| = |\mathsf{kpABE.ct}_{\mathsf{on}}| = \mathrm{poly}(\lambda)$

2. Instantiating prFE scheme as in Theorem 5.29, we have $|\mathsf{prFE.mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{prFE.sk}| = \ell_{\mathsf{on}}\mathrm{poly}(\lambda) = \mathrm{poly}(\lambda)$, $|\mathsf{prFE.ct}| = L_{\mathsf{inp}} + \mathrm{poly}(\lambda)$, where L_{inp} is the input length of the prFE scheme. In our construction $L_{\mathsf{inp}} = |\mathsf{kpABE.st}| + |\mathsf{sd}| + |\mu| = \mathrm{poly}(\lambda)$.

Using the above we note that

- $|\mathsf{cpABE.mpk}| = |\mathsf{prFE.mpk}| = \mathrm{poly}(\lambda)$.
- $|\mathsf{cpABE.sk}_{\mathbf{x}}| = |\mathbf{r}| + |\mathsf{prFE.sk}| = \lambda + \mathrm{poly}(\lambda) = \mathrm{poly}(\lambda)$.
- $|\mathsf{cpABE.ct}| = |\mathsf{prFE.ct}| + |\mathsf{kpABE.mpk}| + |\mathsf{kpABE.ct}_{\mathsf{off}}| + |\mathsf{kpABE.sk}_C| = \mathrm{poly}(\lambda)$.

We formalise the instantiation using the following theorem.

Theorem 5.34. *Under the LWE and Evasive LWE assumption, there exists very selectively secure CP-ABE scheme supporting circuits with unbounded depth* $\{C : \{0,1\}^\ell \to \{0,1\}\}$ *and single bit message space with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda),\ |\mathsf{sk}_{\mathbf{x}}| = \mathrm{poly}(\lambda),\ |\mathsf{ct}| = \mathrm{poly}(\lambda) \tag{5.58}$$

where $\mathbf{x} \in \{0,1\}^\ell$.

5.7.2 Security

We prove the security of our scheme via the following theorem.

Theorem 5.35. *If the* prFE *scheme is secure (Definition 5.13, with* $Q_{\mathsf{msg}} = 1$*), with respect to the sampler class containing the sampler* Samp *as defined in Eq. 5.59 ,* kpABE *scheme satisfies* VerSel-INDr *reusable security (Definition 5.21) then the construction of* cpABE *satisfies* VerSel-IND *security.*

Proof. Suppose the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $C, \mu, \mathbf{x}_1, \ldots \mathbf{x}_Q$.

We want to prove

$$\left(\begin{array}{r} \mathsf{coins}_{\mathcal{A}},\ \mathsf{cpABE.mpk} = \mathsf{prFE.mpk}, \\ \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]},\ \mathsf{cpABE.sk}_{x_k} = \{\mathsf{prFE.sk}_k\}_{k\in[Q]}, \\ \mathsf{cpABE.ct} = (\mathsf{prFE.ct}, \mathsf{kpABE.mpk}, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.sk}_C) \end{array}\right)$$
$$\approx_c \left(\begin{array}{r} \mathsf{coins}_{\mathcal{A}},\ \mathsf{cpABE.mpk} = \mathsf{prFE.mpk}, \\ \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]},\ \mathsf{cpABE.sk}_{x_k} = \{\mathsf{prFE.sk}_k\}_{k\in[Q]}, \\ \mathsf{cpABE.ct} = (\Delta, \mathsf{kpABE.mpk}, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.sk}_C) \end{array}\right)$$

where $\Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}}$, assuming we have $C(\mathbf{x}_k) = 1$ for all the key queries $\mathbf{x}_1, \ldots, \mathbf{x}_Q$ and the challenge circuit C issued by the adversary. Observe that the equation on the left is the view of the adversary in the real world and that on the right is the view of the adversary in the ideal world[8]. Here $\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]$ denotes the functions corresponding to k-th key query $\mathbf{x}_k$ as defined in the KeyGen algorithm, $\mathsf{prFE.sk}_k \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k])$ for $k \in [Q]$, $(\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$, $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu))$ for $\mathsf{sd} \leftarrow \{0,1\}^\lambda$ and $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(\mathsf{kpABE.msk}, C)$.

We invoke the security of prFE with sampler Samp that outputs

$$\left(\begin{array}{rc} \text{Functions:} & \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k]\}_{k\in[Q]}, \\ \text{Input:} & (\mathsf{kpABE.st}, \mathsf{sd}, \mu), \\ \text{Auxiliary Information:} & \mathsf{aux} = (\{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{coins}_{\mathcal{A}}) \end{array}\right) \tag{5.59}$$

By the security guarantee of prFE with sampler Samp we have that

$$\text{if}\quad \left(\mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k], \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)\}_{k\in[Q]}\right) \approx_c \left(\mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \delta_k \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}\}_{k\in[Q]}\right)$$

[8] Note that the information about μ is encoded in $\mathsf{prFE.ct}$.

$$\text{then} \quad \left(\begin{array}{c} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \mathsf{prFE.sk}_k\}_{k\in[Q]} \\ \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{kpABE.st}, \mathsf{sd}, \mu)) \end{array}\right)$$
$$\approx_c \left(\begin{array}{c} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k],\ \mathsf{prFE.sk}_k\}_{k\in[Q]} \\ \Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{array}\right),$$

where one can see that the latter equation is equivalent to Section 5.7.2. Thus to prove Section 5.7.2, it suffices to prove

$$\left(\mathsf{aux}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} = \mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)\}_{k\in[Q]},\ \mathsf{kpABE.sk}_C\right)$$

$$\approx_c \left(\mathsf{aux}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \{\delta_k \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}\}_{k\in[Q]},\ \mathsf{kpABE.sk}_C\right).$$

For clarity in the further steps of security proof we write the equation on L.H.S. of the above equation as

$$\left(\begin{array}{c} \mathsf{coins}_{\mathcal{A}},\ \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.sk}_C, \\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} \coloneqq \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))\}_{k\in[Q]} \end{array}\right), \tag{5.60}$$

where we unroll aux and $\mathrm{F}[\mathbf{x}_k, \mathbf{r}_k](\mathsf{kpABE.st}, \mathsf{sd}, \mu)$ based on their definitions. We prove the pseudorandomness of $\{\mathsf{kpABE.ct}_{\mathsf{on},k}\}_k$ in Equation (5.60) via the following sequence of hybrids.

Hyb_0. This is the distribution in Equation (5.60).

Hyb_1. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}_k\}_{k\in[Q]}$ contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q]$ such that $k \neq k'$ and $\mathbf{r}_k = \mathbf{r}_{k'}$. The probability of this happening can be bounded by $Q^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is

$Q^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_2. This hybrid is same as the previous hybrid except that we change all the PRF values $\{\mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)\}_k$ to truly random values $\{R_k \leftarrow \{0,1\}^\lambda\}_k$. By the change introduced in the previous hybrid, $\mathsf{PRF}(\mathsf{sd}, \cdot)$ is invoked on fresh input for each k. Therefore, we can replace $\{\mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)\}_k$ with truly random $\{R_k\}_k$ by the security of PRF without being noticed by the adversary. We now consider the following distribution:

$$\left(\begin{array}{c} \mathsf{coins}_{\mathcal{A}},\ \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]},\ \mathsf{kpABE.mpk},\ \mathsf{kpABE.sk}_C, \\ \mathsf{kpABE.ct}_{\mathsf{off}},\ \{\mathsf{kpABE.ct}_{\mathsf{on},k} \leftarrow \mathsf{kpABE.EncOn}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu)\}_{k\in[Q]} \end{array}\right),$$

Hyb_3. In this hybrid we invoke the reusable security of kpABE scheme to switch $\{\mathsf{kpABE.ct}_{\mathsf{on},k}\}_k$ to be random strings in $\{0,1\}^{\ell_{\mathsf{on}}}$.

We claim that an adversary $\mathcal{A}$ who can distinguish Hyb_2 and Hyb_3 can be used to break reusable security of kpABE. The reduction $\mathcal{B}$ works as follows.

- $\mathcal{A}$ sends $\mathsf{coins}_{\mathcal{A}}, C, \mathbf{x}_1, \ldots, \mathbf{x}_Q, \mu$ to the reduction.
- $\mathcal{B}$ sends $(C, \{\mathbf{x}_k\}_k, \mu)$ to the kpABE challenger. The challenger does the following.
  - Generates $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk}) \leftarrow \mathsf{kpABE.Setup}(1^\lambda, 1^\ell)$.
  - Computes $\mathsf{kpABE.sk}_C \leftarrow \mathsf{kpABE.KeyGen}(1^\lambda, C)$ and $(\mathsf{kpABE.ct}_{\mathsf{off}}, \mathsf{kpABE.st}) \leftarrow \mathsf{kpABE.EncOff}(\mathsf{kpABE.mpk})$.
  - Computes $\mathsf{kpABE.ct}^0_{\mathsf{on},k} \leftarrow \mathsf{kpABE.Enc}(\mathsf{kpABE.st}, \mathbf{x}_k, \mu)$ and $\mathsf{kpABE.ct}^1_{\mathsf{on},k} \leftarrow \{0,1\}^{\ell_{\mathsf{on}}}$ for all $k \in [Q]$.
  - Samples a bit $\beta \leftarrow \{0,1\}$ and returns $(\mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \{\mathsf{kpABE.ct}^\beta_{\mathsf{on},k}\}_{k\in[Q]})$ to $\mathcal{B}$
- $\mathcal{B}$ returns $\left(\mathsf{coins}_{\mathcal{A}}, \{\mathbf{x}_k, \mathbf{r}_k\}_{k\in[Q]}, \mathsf{kpABE.mpk}, \mathsf{kpABE.sk}_C, \mathsf{kpABE.ct}_{\mathsf{off}}, \{\mathsf{kpABE.ct}^\beta_{\mathsf{on},k}\}_{k\in[Q]}\right)$ to $\mathcal{A}$.

– $\mathcal{A}$ outputs a guess bit β'. $\mathcal{B}$ outputs the same bit as its guess.

We note that if the challenger samples $\beta = 0$, then $\mathcal{B}$ simulates Hyb_2 with adversary else it simulates Hyb_3.

Admissibility of $\mathcal{B}$. Observe that by the admissibility of cpABE, $\mathcal{A}$ sends challenge queries $C, \mathbf{x}_1, \ldots, \mathbf{x}_Q, \mu$ such that $C(\mathbf{x}_k) = 0$ for all $k \in [Q]$. Thus the query $(C, \{\mathbf{x}_k\}_k, \mu)$ sent by $\mathcal{B}$ to the kpABE challenger satisfies $C(\mathbf{x}_k) = 0$ for all $k \in [Q]$. This establishes the admissibility of $\mathcal{B}$.

We observe that the view of the adversary in Hyb_3 is the same as the R.H.S of Section 5.7.2 and hence the proof. ■

Implications to ABE for Turing Machines. We note that our unbounded CP-ABE scheme Theorem 5.34 and unbounded KP-ABE scheme Theorem 5.30 can be used to instantiate ABE for Turing Machines ([13]).

Corollary 5.35.1. *Under the LWE and evasive LWE assumptions, there exists a very selectively secure ABE for TM with*

$$|\mathsf{mpk}| = \mathrm{poly}(\lambda), \quad |\mathsf{sk}| = |M| \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = |\mathbf{x}| \cdot t \cdot \mathrm{poly}(\lambda)$$

where the Turing machine M runs on input x for time step t.

[13] uses the LWE , evasive LWE and circular tensor LWE assumptions for their construction with $|\mathsf{mpk}| = \mathrm{poly}(\lambda)$, $|\mathsf{sk}| = \mathrm{poly}(|M|, \lambda)$, $|\mathsf{ct}| = \mathrm{poly}(\lambda, |\mathbf{x}|, t)$.

APPENDIX

5.A BLIND BATCH ENCRYPTION FROM LWE

In this section, we construct a blind batch encryption scheme from LWE that is required for the construction of the laconic pseudorandom poly-domain obfuscation scheme in Section 5.4.

5.A.1 Basic Scheme from LWE

Here, we provide the construction of basic blind batch encryption under LWE assumption. The construction is a slight variant of the hash encryption scheme proposed in [84], where we modify the construction so that it has perfect correctness and satisfies our notion of strong blindness (Definition 5.9). The construction here does not satisfy the efficiency requirement required in Section 5.4. However, we can bootstrap the construction to satisfy the requirement using the conversion in Section 5.A.2.

In the following, we use the rounding function $\lfloor \cdot \rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ defined as $\lfloor x \rceil_2 \stackrel{\text{def}}{=} \left\lfloor \frac{2}{q} \cdot x \right\rceil \bmod 2$.

$\mathsf{Setup}(1^\lambda, 1^N)$. The setup algorithm does the following.

- Sample $\mathbf{u}_{j,b} \leftarrow \mathbb{Z}_q^n$ for $j \in [N]$ and $b \in \{0,1\}$.
- Output $\mathsf{crs} \coloneqq \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.

$\mathsf{Gen}(\mathsf{crs}, X \in \{0,1\}^N)$. The generation algorithm does the following.

- Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.
- Compute $\mathbf{h} \coloneqq \sum_{j\in[N]} \mathbf{u}_{j,X_j} \in \mathbb{Z}_q^n$ where $X_j \in \{0,1\}$ is the j-th bit of X.
- Output $\mathbf{h}$.

$\mathsf{SingleEnc}(\mathsf{crs}, \mathbf{h}, i, (\mu_0, \mu_1))$. The single encryption algorithm, for $i \in [N]$ and messages $(\mu_0, \mu_1) \in \{0,1\}^2$, does the following.

- Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$.
- Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $e_{j,b} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$ for $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, and $e'_{i,0}, e'_{i,1} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}$.
- Compute $c_{j,b} \coloneqq \mathbf{s}^\top \mathbf{u}_{j,b} + e_{j,b}$ for $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$.
- Compute
$$c_{i,b} \coloneqq \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,b}\right) + e'_{i,b} \right\rceil_2 \oplus \mu_b$$
for $b \in \{0,1\}$.

– Check whether $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,b}) \in [q/4 - B, q/4 + B] \cup [3q/4 - B, 3q/4 + B]$ holds for $b = 0$ or $b = 1$. If so, replace each of $\{c_{j,b}\}_{j\in[N],b\in\{0,1\}}$ with $\perp$ and set $c_{i,b} = \mu_b$ for $b \in \{0, 1\}$ and some $B > 0$[9]. Otherwise, do nothing.

– Output $\mathsf{ct} \coloneqq \{c_{j,b}\}_{j\in[N],b\in\{0,1\}}$.

$\mathsf{SingleDec}(\mathsf{crs}, X, i, \mathsf{ct})$. The single decryption algorithm, for $X \in \{0,1\}^N$ and $i \in [N]$, does the following.

– Parse $\mathsf{crs} = \{\mathbf{u}_{j,b}\}_{j\in[N],b\in\{0,1\}}$, $\mathsf{ct} \to (\{c_{j,b}\}_{j\in[N],b\in\{0,1\}})$ where $c_{j,b} \in \mathbb{Z}_q \cup \{\perp\}$ for $j \in [N]\backslash\{i\}$ and $c_{i,b} \in \{0, 1\}$ for $b \in \{0, 1\}$.

– If $c_{j,b} = \perp$ for some j, b, set $\mu' \coloneqq c_{i,X_i}$.

– Otherwise, compute

$$\mu' \coloneqq c_{i,X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} c_{j,X_j} \right\rceil_2 .$$

– Output μ'.

Remark 29. We note that the crs in the above scheme is a uniformly random bit string.

Parameter. We set q to be a power of 2, so that a random string over $\mathbb{Z}_q$ can be interpreted as a binary random string. We set γ super-polynomially larger than σ so that the smudging is possible.

$$q = 2^{5\lambda}, \quad B = 2^{3\lambda}, \quad , \sigma = 2^{2\lambda}/N, \quad , \gamma = 2^{2\lambda}\lambda^{\omega(1)}$$

Succinctness. We can see that the hash is of fixed length and thus the construction is fully succinct.

[9] We introduce this step to remove the correctness error.

Correctness. With the above set parameters, we show that our scheme achieves perfect correctness.

- If $c_{j,b} = \perp$ for some $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, then $\mu' \coloneqq c_{i,X_i} = \mu_{X_i}$ with probability 1 and hence the correctness.
- If $c_{j,b} \neq \perp$ for any $j \in [N]\backslash\{i\}$ and $b \in \{0,1\}$, we compute

$$\begin{aligned}
\mu' &= c_{i,X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} c_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,X_i}\right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \mathbf{s}^\top \left(\sum_{j\in[N]} \mathbf{u}_{j,X_j} - \mathbf{u}_{i,X_i} \right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \oplus \left\lfloor \sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j} \right\rceil_2 \\
&= \mu_{X_i}
\end{aligned}$$

  where the last equality follows from our parameter setting, $|e'_{i,X_i}| < B$ and $\sum_{j\in[N]\backslash\{i\}} |e_{j,X_j}| \leq B$ and the guarantee that $\sum_{j\in[N]\backslash\{i\}} \mathbf{s}^\top \mathbf{u}_{j,X_j}$ is sufficiently far, from the second to the last step of the encryption algorithm, from the "unsafe zone" where small noise can change the rounded value (i.e., $[q/4 - B, q/4 + B] \cup [3q/4 - B, 3q/4 + B]$).

Security. Next, we prove the security of our scheme.

Theorem 5.36. *The above construction satisfies* SingleEnc *security (Definition 5.8) under the LWE assumption.*

Proof. We consider the following sequence of hybrids.

Hyb_0. Real game.

Hyb_1. Change the encryption algorithm so that it never sets $c_{j,b} = \perp$ (i.e., we erase the branch of the computation introduced to eliminate the possibility of the decryption

error). Since $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,0})$ and $\mathbf{s}^\top(\mathbf{h} - \mathbf{u}_{i,1})$ are distributed uniformly at random over $\mathbb{Z}_q$, the probability of $\perp$ being output is bounded by $4B/q$. Therefore, this hybrid is statistically indistinguishable from the previous one.

Hyb_2. In this hybrid we compute c_{i,X_i} as

$$c_{i,X_i} = \left\lfloor \sum_{j \in [N] \setminus \{i\}} c_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}.$$

By the smudging lemma, this hybrid is statistically close to the previous hybrid. To see this, we note the following.

- In Hyb_1 we have

$$c_{i,X_i} \coloneqq \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,X_i}\right) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}$$

- Substituting $\mathbf{h} = \sum_{j \in [N]} \mathbf{u}_{j,X_j}$, we get

$$c_{i,X_i} = \left\lfloor \mathbf{s}^\top \sum_{j \in [N] \setminus \{i\}} \mathbf{u}_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i}.$$

- From our parameter setting we have $|\sum_{j \in [N] \setminus \{i\}} e_{j,X_j}| \leq \lambda^{\omega(1)} \sum_{j \in [N] \setminus \{i\}} |e_{j,X_j}| \leq |e'_{i,X_i}|$ where $e_{j,b} \in \mathcal{D}_{\mathbb{Z},\sigma}$ and $e'_{i,X_i} \in \mathcal{D}_{\mathbb{Z},\gamma}$. Thus using noise flooding (Theorem 2.3) and Section 5.A.1 we have

$$\begin{aligned} c_{i,X_i} &= \left\lfloor \mathbf{s}^\top \sum_{j \in [N] \setminus \{i\}} \mathbf{u}_{j,X_j} + \sum_{j \in [N] \setminus \{i\}} e_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \\ &= \left\lfloor \sum_{j \in [N] \setminus \{i\}} (\mathbf{s}^\top \mathbf{u}_{j,X_j} + e_{j,X_j}) + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \\ &= \left\lfloor \sum_{j \in [N] \setminus \{i\}} c_{j,X_j} + e'_{i,X_i} \right\rceil_2 \oplus \mu_{X_i} \end{aligned}$$

with overwhelming probability.

Hyb_3. To compute $c_{i,1-X_i}$, we set it as

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i} + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i},$$

where $e_{i,1-X_i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$. By the smudging lemma, this hybrid is statistically close to the previous hybrid.

To see this, we note that initially we have

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i}$$

where $e'_{i,1-X_i} \in \mathcal{D}_{\mathbb{Z},\gamma}$. Next, from our parameter setting, we have $\lambda^{\omega(1)}|e_{i,1-X_i}| \leq |e'_{i,1-X_i}|$ where $e_{i,1-X_i} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$. Thus using noise flooding (Theorem 2.3), we can write Section 5.A.1 as

$$c_{i,1-X_i} = \left\lfloor \mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i} + e'_{i,1-X_i} \right\rceil_2 \oplus \mu_{1-X_i}.$$

Hyb_4. Replace $\{c_{j,b}\}_{j\neq i, b\in\{0,1\}}$ and $\mathbf{s}^\top \left(\mathbf{h} - \mathbf{u}_{i,1-X_i}\right) + e_{i,1-X_i}$ computed for $c_{i,1-X_i}$ with random elements in $\mathbb{Z}_q$. This game is computationally indistinguishable from the previous one by LWE.

Hyb_5. Now, $c_{i,1-X_i}$ is sampled as $c_{i,1-X_i} \leftarrow \{0,1\}$. This game is statistically close to the previous game, since the value inside the round function is already uniformly random over $\mathbb{Z}_q$.

■

Theorem 5.37. *The above construction satisfies strong blindness as per Definition 5.9 under the LWE assumption.*

Proof. The proof is almost the same as that of Theorem 5.36, where we consider the same sequence of the hybrids. We can skip Hyb_2, since c_{i,X_i} is already random by the definition of the blindness security game. In Hyb_5, the ciphertext is a random string.

Therefore, we can conclude the proof of the theorem. ■

5.A.2 Bootstrapping the Basic Scheme

Our construction of BBE in Section 5.A.1 has large CRS size and single ciphertext size, both of which linearly depend on N. However, we require a BBE scheme whose sizes of these parameters are independent of N in Section 5.4. Here, we obtain a scheme with the required properties from LWE by applying the conversion from [58] to our construction in Section 5.A.1.

Theorem 5.38 (Adapted from Appendix A.2 of [58])**.** *Assuming that there exists a* $1/2$*-succinct* BBE *scheme (defined in Definition 5.7) whose CRS size is* $\mathrm{poly}(\lambda, N)$ *and single-ciphertext size is* $\mathrm{poly}(\lambda, N)$*. Then, the construction can be converted into a fully-succinct* BBE *scheme with CRS size* $\mathrm{poly}(\lambda, \log N)$ *and single-ciphertext size* $\mathrm{poly}(\lambda, \log N)$*. Furthermore, the conversion preserves strong blindness property.*

Sketch of proof. Since the above theorem is not explicitly shown in [58], we provide an explanation on how to extract the above theorem from their result. There, they show a construction of fully-succinct BBE scheme starting from a $1/2$-succinct BBE scheme. They only provide the description of the encryption algorithm, but it is easy to extract a description of single encryption algorithm from it. For the reference for the readers, we sketch their construction and explain how to obtain the single encryption algorithm out of it.

To setup the system, they generate $d = \log(N/\lambda)$ number of CRSes, all of which are generated by $\mathsf{Setup}(1^\lambda, 1^{2\lambda})$. Each CRS string is assigned to each layer of the tree. To compute a hash value on input $X \in \{0,1\}^N$, they consider a Merkle tree of depth d. The leaves of the tree consist of N/λ nodes and X is evenly split and assigned to the corresponding node. Then, we assign hash values to the internal nodes of the tree starting from the layer of the tree right above the leaves to the root. To define a hash value h_v associated to a node v, we hash the values associated with its children, namely, $h_{v\|0}$ and

$h_{v\|1}$, where we use the CRS corresponding to that layer. The final output of the hash (i.e., $\mathsf{Gen}(\mathsf{crs}, X)$) is the hash value h_ϵ assigned to the root node ϵ.

We then explain how the encryption algorithm works. For each node v, they generate subct_1 part of the underlying BBE ciphertext and a garbled circuit.[10] We denote the former by $\mathsf{subct}_{v,1}$ and the latter $\tilde{C}_v$. For the garbled circuit corresponding to the root node, we additionally provide the input labels $\mathsf{lab}_{h_\epsilon}$ that corresponds to h_ϵ. The garbled circuits associated with internal nodes are obtained by garbling a circuit that takes as input h_v and outputs subct_2 part of the BBE ciphertext that encrypts the labels of the garbled circuits corresponding to its children under the public key h_v. For the leaf nodes, the garbled circuits encrypt the messages instead of the labels.

To decrypt a ciphertext, for each leaf v, we traverse the hash tree from the root to v and obtain the message corresponding to X_v as follows. We first obtain $\mathsf{subct}_{\epsilon,2}$ by evaluating $\tilde{C}_\epsilon$ on labels $\mathsf{lab}_{h_\epsilon}$. Then, combined with $\mathsf{subct}_{\epsilon,1}$, this recovers the entire BBE ciphertext corresponding to the root node ϵ that encrypts the labels of the garbled circuits of the next layer. This BBE ciphertext can be decrypted by using the string that concatenates h_0 and h_1. This in particular gives the labels $\mathsf{lab}_{h_{v_1}}$ corresponding to h_{v_1}, where v_1 is the first bit of v. This then allows us to recover $\mathsf{subct}_{v_1,2}$ by evaluating the garbled circuit. We traverse down the tree in this way until we reach at the leaf node v, where we recover the corresponding message.

We then explain how we define the single encryption algorithm. To encrypt a message for a position i, we consider the leaf node v corresponding to the index i. We then run the encryption algorithm and remove the ciphertext components that are not necessary for traversing down the tree to the leaf node v. Namely, we only include $\mathsf{lab}_{h_\epsilon}$ and $\mathsf{subct}_{w,1}, \tilde{C}_w$ for all w that is an ancestor of v in the ciphertext. We can see that these components are sufficient for the decryption to work correctly.

[10] We refer to Remark 17 for the explanation on subct_1 and subct_2.

Now, we can see that the construction achieves CRS size of $\mathrm{poly}(\lambda, \log N)$, since there are $d = \log(N/\lambda)$ number of CRSes of the base BBE. Similarly, the single-ciphertext size is of $\mathrm{poly}(\lambda, \log N)$, since there are d number of garbled circuits and $\mathsf{subct}_{v,1}$, each of which is of fixed polynomial size. We also observe that the conversion preserves the strong blindness. In [58], they show that the ciphertext is pseudorandom except for $\{\mathsf{subct}_{w,1}\}_{w:w \text{ is an ancestor of } v}$. If the underlying BBE satisfies strong blindness, $\mathsf{subct}_{w,1}$ is an empty string for all w. This indicates that the entire ciphertext of the final scheme is pseudorandom, as desired. This holds even if we consider single-ciphertext, since the single ciphertext is obtained by removing some part of the ciphertext. ■

By applying the conversion to our construction in Section 5.A.1, we obtain the following theorem.

Theorem 5.39. *There exists a fully-succinct* BBE *scheme with CRS size of* $\mathrm{poly}(\lambda, \log N)$ *and single-ciphertext size of* $\mathrm{poly}(\lambda, \log N)$ *from LWE where* N *denotes the length of the keys supported by the scheme*

5.B BOOTSTRAPPING AB-LFE TO KP-ABE WITH UNBOUNDED DEPTH USING prFE

In this section, we show a formal description of the construction of kpABE for unbounded circuits using $\mathsf{1ABE}$ sketched in Section 5.1.4. This provides an alternative pathway to obtain kpABE for unbounded depth circuits different from that given in Section 5.6. Instead of $\mathsf{1ABE}$, our construction is formally described using abLFE, but this turns out to be almost equivalent, as discussed later in this section. We can instantiate the abLFE by the recent construction by [122] or blind garbled circuit [58]. The former instantiation leads to more efficient construction than the latter, but it introduces an additional assumption of circular LWE in addition to LWE and evasive LWE. Compared with Section 5.6, the constructions obtained here are simpler, though their parameter sizes are sub-optimal. Importantly, we do not rely on the result regarding pPRIO from

our companion paper [14] in this section.

5.B.1 Attribute Based Laconic Functional Encryption

Syntax. An attribute based laconic function evaluation (abLFE) scheme for a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$ for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$ and a message space $\mathcal{M}$ consists of four algorithms ($\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec}$) defined as follows.

$\mathsf{crsGen}(1^\lambda, \mathsf{prm}) \to \mathsf{crs}$. The generation algorithm takes as input the security parameter 1^λ and circuit parameters prm and outputs a uniformly sampled common reference string crs.

$\mathsf{Compress}(\mathsf{crs}, C) \to \mathsf{digest}$. The compress algorithm takes as input the common random string crs and a circuit $C \in \mathcal{C}$ and outputs a digest digest.

$\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) \to \mathsf{ct}$. The encryption algorithm takes as input the common random string crs, a digest digest, an attribute $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and a message $\mu \in \mathcal{M}$ and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \to \mu/\perp$. The decryption algorithm takes as input the common random string crs, a circuit C, digest and a ciphertext ct and outputs a message $\mu \in \mathcal{M}$ or $\perp$.

Definition 5.22 (Correctness)**.** An abLFE scheme for circuit family $\mathcal{C}_{\mathsf{prm}}$ is correct if for all prm, $C \in \mathcal{C}_{\mathsf{prm}}, \mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ such that $C(\mathbf{x}) = 1$, and for all messages $\mu \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l} \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm}), \\ \mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, C), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) : \\ \mathsf{Dec}(\mathsf{crs}, C, \mathsf{ct}) \neq \mu \end{array}\right] = \mathsf{negl}(\lambda)$$

where the probability is taken over the coins of Setup, KeyGen, and Enc.

Definition 5.23 (Pseudorandom Ciphertext Security)**.** For a abLFE scheme and an adversary $\mathcal{A}$, we define the experiment for security $\mathsf{Expt}^{\mathsf{abLFE}}_{\beta,\mathcal{A}}(1^\lambda)$ as follows.

1. Run $\mathcal{A}$ to receive circuit parameters prm. Run $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and send crs to $\mathcal{A}$.

2. $\mathcal{A}$ chooses $C \in \mathcal{C}_{\mathsf{prm}}$, $\mathbf{x} \in \mathcal{X}_{\mathsf{prm}}$ and $\mu \in \mathcal{M}$. Run $\mathsf{digest} = \mathsf{Compress}(\mathsf{crs}, C)$, sample $\beta \leftarrow \{0,1\}$. If $\beta = 0$, it computes $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu))$ else if $\beta = 1$, it computes $\mathsf{ct}_1 \leftarrow \mathcal{CT}_{\mathsf{abLFE}}$, where $\mathcal{CT}_{\mathsf{abLFE}}$ is the ciphertext space of abLFE. It sends $\mathsf{digest}, \mathsf{ct}_\beta$ to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

We define the advantage $\mathsf{Adv}^{\mathsf{abLFE}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in the above game as

$$\mathsf{Adv}^{\mathsf{abLFE}}_{\mathcal{A}}(\lambda) \coloneqq \left|\mathsf{Pr}[\mathsf{Expt}^{\mathsf{abLFE}}_{0,\mathcal{A}}(1^\lambda) = 1] - \mathsf{Pr}[\mathsf{Expt}^{\mathsf{abLFE}}_{1,\mathcal{A}}(1^\lambda) = 1]\right|.$$

We say that a abLFE scheme is *adaptive* pseudorandom ciphertext secure if for every *admissible* PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{abLFE}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$, where $\mathcal{A}$ is said to be admissible if $C(\mathbf{x}) = 0$.

The *selective* (resp. *very selective*) notion of the security requires the adversary $\mathcal{A}$ to choose $\mathbf{x}$ (resp. $\mathbf{x}, C$) along with prm before it receives crs.

Definition 5.24 (Decomposability)**.** We say that a abLFE scheme for a circuit class $\{\mathcal{C}_{\mathsf{prm}} = \{C : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$ satisfies decomposability if for any $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda, \mathsf{prm})$ and $\mathsf{digest} \leftarrow \mathsf{Compress}(\mathsf{crs}, C)$, we have $\mathsf{digest} = \{\mathsf{digest}_i\}_{i\in[q_C]}$ for some polynomial q_C, which may depend on C, and $\mathsf{size}(\mathsf{digest}_i) \leq \mathrm{poly}(\lambda)$. Furthermore, we have that $\mathsf{Enc}(\mathsf{crs}, \mathsf{digest}, (\mathbf{x}, \mu)) = \{\mathsf{Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu))\}_{i\in[q_C]}$ where size of the encryption circuit $\mathsf{size}(\mathsf{Enc}_i(\cdot,\cdot,(\cdot,\cdot))) \leq \mathrm{poly}(\lambda, |\mathbf{x}|)$.

Remark 30. Here, we do not require the digest to be much smaller than the circuit description C, unlike the usual convention in the context of abLFE. This relaxation allows us to instantiate abLFE using blind garbled circuits, which do not have compact digests.

5.B.2 Construction of kpABE with Unbounded Depth

Building Blocks. We require the following building blocks for our construction.

1. An attribute-based laconic function evaluation scheme $\mathsf{abLFE} = \mathsf{abLFE}.(\mathsf{crsGen}, \mathsf{Compress}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $C_{\mathrm{L}(\lambda)}$, consisting of circuits with input length $\mathrm{L}(\lambda)$ and with unbounded depth and size. We let $\ell_{\mathsf{ct}}^{\mathsf{abLFE}}$ and $\mathcal{CT}_{\mathsf{abLFE}} = \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{abLFE}}}$ denote the ciphertext length and the ciphertext space of the scheme, respectively. We assume that the abLFE scheme is decomposable (Definition 5.24), i.e., we have $\mathsf{abLFE.Enc} = \{\mathsf{abLFE.Enc}_i\}_{i\in[q_C]}$ for some q_C and use d_{abLFE} to denote the maximum depth of a circuit required to compute $\{\mathsf{abLFE.Enc}_i\}_{i\in[q_C]}$.

2. A FE scheme for pseudorandom functionality $\mathsf{prFE} = (\mathsf{prFE.Setup}, \mathsf{prFE.KeyGen}, \mathsf{prFE.Enc}, \mathsf{prFE.Dec})$ for circuit class $C_{\mathrm{L}(\lambda), d_{\mathsf{prFE}}(\lambda), \ell_{\mathsf{ct}}^{\mathsf{abLFE}}}$ consisting of circuits with input length $\mathrm{L}(\ell, \lambda)$, maximum depth $d_{\mathsf{prFE}}(\lambda)$ and output length $\ell_{\mathsf{ct}}^{\mathsf{abLFE}}$. We denote by prm the parameters $(1^{\mathrm{L}(\lambda)}, 1^{d_{\mathsf{prFE}}(\lambda)}, 1^{\ell_{\mathsf{ct}}^{\mathsf{abLFE}}})$ that specifies the function class being supported. We also denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{prFE}}$.

3. A PRF function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\mathsf{R}_{\mathsf{len}}}$ where $\mathsf{R}_{\mathsf{len}}$ is the length of randomness used in $\mathsf{abLFE.Enc}$. We assume that PRF can be computed by a circuit of depth at most d_{prFE}.

We assume that uniform sampling from the ciphertext space is possible without any parameter other than the security parameter λ.

Now, we describe our compiler for constructing a key-policy ABE scheme $\mathsf{kpABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuits of unbounded depth with attribute length $\mathrm{L}(\lambda)$. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{kpABE}}$. For our construction, we have $\mathcal{CT}_{\mathsf{kpABE}} = \mathcal{CT}_{\mathsf{prFE}}$.

$\mathsf{Setup}(1^{\lambda}, 1^{\mathrm{L}}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Run $(\mathsf{prFE.msk}, \mathsf{prFE.mpk}) \leftarrow \mathsf{prFE.Setup}(1^{\lambda}, \mathsf{prm})$ and $\mathsf{crs} \leftarrow \mathsf{abLFE.crsGen}(1^{\lambda})$.
- Set $\mathsf{msk} = \mathsf{prFE.msk}$[11] and $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$. Output $(\mathsf{msk}, \mathsf{mpk})$.

[11] W.L.O.G we assume that msk contains mpk.

$\mathsf{KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$. The key generation algorithm does the following.

- Parse $\mathsf{msk} = \mathsf{prFE.msk}$ and sample $\mathbf{r} \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{digest} = \mathsf{abLFE.Compress}(\mathsf{crs}, C)$. Parse $\mathsf{digest} = \{\mathsf{digest}_i\}_{i\in[q_C]}$.
- For $i \in [q_C]$, define circuit $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}]$, with $\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}$ hardwired, as follows
  On input $(\mathsf{sd}, \mathbf{x}, \mu)$:
  - Compute $\mathsf{abLFE.ct}_i \coloneqq \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
  - Output $\mathsf{abLFE.ct}_i$.
- For $i \in [q_C]$ compute $\mathsf{prFE.sk}_i \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}])$.
- Output $\mathsf{sk}_C \coloneqq \left(\{\mathsf{digest}_i,\ \mathsf{prFE.sk}_i\}_{i\in[q_C]}, \mathbf{r}\right)$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu) \to \mathsf{ct}$. The encryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu))$.
- Output $\mathsf{ct} \coloneqq \mathsf{prFE.ct}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}, \mathbf{x}) \to \mathbf{y}$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$, $\mathsf{sk}_C = \left(\{\mathsf{digest}_i,\ \mathsf{prFE.sk}_i\}_{i\in[q_C]}, \mathbf{r}\right)$ and $\mathsf{ct} = \mathsf{prFE.ct}$.
- For all $i \in [q_C]$, compute $y_i = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}_i, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}], \mathsf{prFE.ct})$.
- Set $\mathbf{y} = (y_1, \ldots, y_{q_C})$ and output $\mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathbf{y})$.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 5.40. *Assume* abLFE *and* prFE *schemes are correct, and* PRF *is secure. Then the above construction of* kpABE *scheme is correct.*

Proof. From the correctness of prFE scheme we have

$$\begin{aligned} y_i =& \mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}](\mathsf{sd}, \mathbf{x}, \mu) \\ =& \mathsf{abLFE.ct}_i = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_i, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r})) \end{aligned}$$

for $i \in [q_C]$ with probability 1. Thus we have $\mathbf{y} = \{\mathsf{abLFE.ct}_i\}_{i\in[q_C]} = \mathsf{abLFE.ct}$. Next, by the correctness of abLFE scheme it follows that, if $C(\mathbf{x}) = 1$,

$$\mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathbf{y}) = \mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathsf{abLFE.ct}) = \mu$$

with all but negligible probability. ■

Security. We prove the security of our scheme via the following theorem.

Theorem 5.41. *Assume that the* prFE *scheme is secure (Definition 5.13) with respect to the sampler class containing the sampler* Samp *as defined in Eq. 5.62,* abLFE *scheme satisfies very selective pseudorandom ciphertext security (Definition 5.23). Then our construction of* kpABE *scheme satisfies* VerSel-INDr *security (Definition 2.8).*

Proof. Suppose the adversary $\mathcal{A}$ with randomness $\mathsf{coins}_{\mathcal{A}}$ queries for $\mathbf{x}, \mu, C_1, \ldots C_Q$. To prove the security of kpABE scheme as per Definition 2.8, we show

$$\begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs}), \\ \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{sk}_{C_k} = \{\mathsf{prFE.sk}_{k,i}\}_{k\in[Q],i\in[q_{C_k}]}, \\ \mathsf{ct} = \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu)) \end{pmatrix} \approx_c \begin{pmatrix} \mathsf{coins}_{\mathcal{A}},\ \mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs}), \\ \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q], i\in[q_{C_k}]}, \\ \mathsf{sk}_{C_k} = \{\mathsf{prFE.sk}_{k,i}\}_{k\in[Q],i\in[q_{C_k}]}, \\ \mathsf{ct} = \delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{pmatrix} \tag{5.61}$$

where $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]$ denotes the functions corresponding to k-th key query C_k as defined in the KeyGen algorithm and $\mathsf{prFE.sk}_{k,i} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k])$ for $k \in [Q], i \in [q_{C_k}]$. Also for all the key queries $C_1, \ldots, C_Q$ and the challenge attribute $\mathbf{x}$ issued by the adversary, we have $C_k(\mathbf{x}) = 0$.

We invoke the security of prFE with sampler Samp that outputs

$$\begin{pmatrix} \text{Functions:} & \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k]\}_{k\in[Q],i\in[q_{C_k}]}, \\ \text{Input:} & (\mathsf{sd}, \mathbf{x}, \mu), \\ \text{Auxiliary Information:} & \mathsf{aux} = (\mathsf{coins}_{\mathcal{A}}, \{\mathsf{crs}, C_k, \mathsf{digest}_{k,i}, \mathbf{r}_k\}_{k\in[Q],i\in[q_{C_k}]}) \end{pmatrix} \tag{5.62}$$

By the security guarantee of prFE with sampler Samp we have

$$\begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{prFE.sk}_{k,i}\}_{k\in[Q],i\in[q_{C_k}]} \\ \mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathbf{x}, \mu)) \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{prFE.mpk},\ \mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{prFE.sk}_{k,i}\}_{k\in[Q],i\in[q_{C_k}]} \\ \Delta \leftarrow \mathcal{CT}_{\mathsf{prFE}} \end{pmatrix}$$

if

$$\begin{aligned} &\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_{k,i}\}_{k\in[Q],i\in[q_{C_k}]}\right) \\ \approx_c &\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ (\delta_{k,1}, \ldots, \delta_{k,q_{C_k}}) \leftarrow \{0,1\}^{\ell_{\mathsf{ct}}^{\mathsf{abLFE}}}\}_{k\in[Q]}\right) \end{aligned} \tag{5.63}$$

where

$\mathsf{abLFE.ct}_{k,i} = \mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k](\mathsf{sd}, \mathbf{x}, \mu) = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_{k,i}, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k))$ for $k \in [Q], i \in [q_C]$.

Thus to prove Equation (5.61), it suffices to prove Equation (5.63). We prove Equation (5.63) via the following sequence of hybrids.

Hyb_0. This is the LHS distribution of Equation (5.63).

$$\left(\mathsf{aux}, \left\{ \begin{matrix} \mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k], \\ \mathsf{abLFE.ct}_{k,i} = \mathsf{abLFE.Enc}_i(\mathsf{crs}, \mathsf{digest}_{k,i}, (\mathbf{x}, \mu); \mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)) \end{matrix} \right\}_{k\in[Q],i\in[q_{C_k}]}\right).$$

We can rewrite the above distribution as

$$\left(\mathsf{aux}, \left\{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k\right\}_{k\in[Q],i\in[q_{C_k}]}\right).$$

where $\mathsf{digest}_k = \{\mathsf{digest}_{k,i}\}_{i\in[q_{C_k}]}$ and $\mathsf{abLFE.ct}_k = \{\mathsf{abLFE.ct}_{k,i}\}_{i\in[q_{C_k}]}$ for all $k \in [Q]$.

Hyb_1. This hybrid is same as the previous one except that we output a failure symbol if the set $\{\mathbf{r}_k\}_{k\in[Q]}$, in aux, contains a collision. We prove that the probability with which there occurs a collision is negligible in λ. To prove this it suffices to show that there is no $k, k' \in [Q]$ such that $k \neq k'$ and $\mathbf{r}_k = \mathbf{r}_{k'}$. The probability of this happening can be bounded by $Q^2/2^\lambda$ by taking the union bound with respect to all the combinations of k, k'. Thus the probability of outputting the failure symbol is $Q^2/2^\lambda$ which is $\mathsf{negl}(\lambda)$.

Hyb_2. In this hybrid we change all the PRF values computed using sd to random. Namely, we replace $\mathsf{PRF}(\mathsf{sd}, \mathbf{r}_k)$ with true randomness R_k. Since PRF is invoked for fresh input for each $k \in [Q]$, this hybrid is indistinguishable from the previous hybrid. We now consider the following distribution:

$$\left(\mathsf{aux}, \{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k = \mathsf{abLFE.Enc}(\mathsf{crs}, \mathsf{digest}_k, (\mathbf{x}, \mu))\}_{k\in[Q],i\in[q_{C_k}]}\right)$$

Hyb_3. In this hybrid we invoke the security of abLFE scheme to switch $\mathsf{abLFE.ct}_k$ to random for all $k \in [Q]$. Namely, the distribution is now:

$$\left(\{\mathrm{F}[\mathsf{crs}, \mathsf{digest}_{k,i}, \mathbf{r}_k],\ \mathsf{abLFE.ct}_k \leftarrow \mathcal{CT}_{\mathsf{abLFE}}\}_{k\in[Q],i\in[q_{C_k}]}\right)$$

By the admissibility of the adversary and very selective pseudorandom ciphertext security of abLFE, this hybrid is indistinguishable from the previous one.

This completes the proof. ■

Instantiating the abLFE scheme as above, and using a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \mathrm{L} + \lambda + 1$, we obtain the following theorem.

Theorem 5.42. *Under the LWE assumption and the Evasive LWE assumption, there*

exists a very selectively secure kpABE *scheme for circuits of unbounded depth and attribute length* ℓ *with*

$$|\mathsf{mpk}| = \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{sk}_C| = |C| \cdot \ell \cdot \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \ell \cdot \mathrm{poly}(\lambda).$$

We note that the kpABE scheme instantiated as above has longer secret keys but is not based on any circular assumptions.

5.B.3 Using the abLFE from HLL

We can directly instantiate the abLFE in Section 5.B.2 with the construction shown by [122] (HLL henceforth). For this instantiation, we do not assume decomposability of abLFE since the HLL construction has succinct digest, i.e. $q_C = 1$ for any C and $|\mathsf{digest}| = O(1)$. This instantiation leads to the parameter size of $|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda)$, $|\mathsf{sk}_C| = \mathrm{poly}(\ell, \lambda)$, and $|\mathsf{ct}| = \mathrm{poly}(\ell, \lambda)$, which is already better than the previous instantiation.

By adding a twist to the construction in Section 5.B.2 exploiting the structural property of HLL, we can improve the secret key size so that its dependency on ℓ can be removed. To do so, we exploit the online-offline structure of the abLFE.Enc algorithm which can be split as $\mathsf{abLFE.Enc} = (\mathsf{abLFE.EncOff}, \mathsf{abLFE.EncOn})$. Here, abLFE.EncOff takes as input crs and $\mathbf{x}$ and outputs the offline part of the ciphertext $\mathsf{abLFE.ct}_{\mathsf{off}}$ and short state st and abLFE.EncOn takes as input digest and st and outputs online part of the ciphertext $\mathsf{abLFE.ct}_{\mathsf{on}}$. Formally, HLL proved the following theorem:

Theorem 5.43 ([122])**.** *Under the circular LWE assumption, there exists a very selectively secure* abLFE *scheme for circuit class* $C = \{C : \{0,1\}^{\ell} \to \{0,1\}^{\ell'}\}$ *satisfying*

$$|\mathsf{crs}| = O(\ell, \lambda), \ |\mathsf{digest}| = O(\lambda), \ |\mathsf{st}| = O(\lambda), \ |\mathsf{ct}_{\mathsf{off}}| = O(\ell, \lambda), \ |\mathsf{ct}_{\mathsf{on}}| = O(\ell', \lambda)$$

We make slight modifications to our construction of kpABE scheme to optimize the secret key size. The high level idea is very simple. Instead of letting the prFE decryption recover the entire abLFE ciphertext, we recover only the online part of it. We then put the

offline part of abLFE ciphertext into the ciphertext of kpABE so that the decryptor can recover the entire abLFE ciphertext during the decryption. This eliminates the necessity of hardwiring crs to the prFE secret key, since the online part can be computed only from the short state and digest. This leads to the improvement on the efficiency, since the state and digest are of fixed polynomial size, while crs is of size $O(\ell)$. Concretely, we modify the $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ algorithm of Section 5.B.2 as follows.

$\mathsf{KeyGen}(\mathsf{msk}, C)$. This is same as the KeyGen algorithm in Section 5.B.2 except the following.

- Define circuit $\mathrm{F}[\mathsf{digest}, \mathbf{r}]$ (instead of $\mathrm{F}[\mathsf{crs}, \mathsf{digest}_i, \mathbf{r}]$) , with $\mathsf{digest}, \mathbf{r}$ hardwired, as follows
  On input $(\mathsf{sd}, \mathsf{st})$:
  - Compute $\mathsf{abLFE.ct}_{\mathsf{on}} := \mathsf{abLFE.EncOn}(\mathsf{st}, \mathsf{digest}; \mathsf{PRF}(\mathsf{sd}, \mathbf{r}))$.
  - Output $\mathsf{abLFE.ct}_{\mathsf{on}}$.
- Compute $\mathsf{prFE.sk} \leftarrow \mathsf{prFE.KeyGen}(\mathsf{prFE.msk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}])$.
- Output $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$. The encryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$ and sample a PRF key $\mathsf{sd} \leftarrow \{0,1\}^\lambda$.
- Compute $(\mathsf{abLFE.ct}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{abLFE.EncOff}(\mathsf{crs}, (\mathbf{x}, \mu))$.
- Compute $\mathsf{prFE.ct} \leftarrow \mathsf{prFE.Enc}(\mathsf{prFE.mpk}, (\mathsf{sd}, \mathsf{st}))$.
- Output $\mathsf{ct} := (\mathsf{abLFE.ct}_{\mathsf{off}}, \mathsf{prFE.ct})$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_C, C, \mathsf{ct}, \mathbf{x})$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{prFE.mpk}, \mathsf{crs})$, $\mathsf{sk}_C = (\mathsf{digest}, \mathsf{prFE.sk}, \mathbf{r})$ and $\mathsf{ct} = (\mathsf{abLFE.ct}_{\mathsf{off}}, \mathsf{prFE.ct})$.
- Compute
  $\mathsf{abLFE.ct}_{\mathsf{on}} = \mathsf{prFE.Dec}(\mathsf{prFE.mpk}, \mathsf{prFE.sk}, \mathrm{F}[\mathsf{digest}, \mathbf{r}], \mathsf{prFE.ct})$.

– Set $\mathbf{y} = (\mathsf{abLFE.ct}_{\mathsf{off}}, \mathsf{abLFE.ct}_{\mathsf{on}})$ and output $\mathsf{abLFE.Dec}(\mathsf{crs}, C, \mathbf{y})$.

We note that even with the above changes the correctness and security arguments are same as that of Section 5.B.2.

For this instantiation, we have $d^{\mathsf{EncOn}}_{\mathsf{abLFE}} = \mathrm{poly}(\lambda)$ where $d^{\mathsf{EncOn}}_{\mathsf{abLFE}}$ is the maximum depth of a circuit required to compute $\mathsf{abLFE.EncOn}$ and hence we use a prFE scheme supporting $d_{\mathsf{prFE}} = \mathrm{poly}(\lambda)$ depth circuits with input length $\mathrm{L} = \mathrm{poly}(\lambda)$. We formalise this using the following theorem.

Theorem 5.44. *Under the circular LWE assumption and the Evasive LWE assumption, there exists a very selectively secure* kpABE *scheme for circuits of unbounded depth and attribute length* ℓ *with*

$$|\mathsf{mpk}| = \mathrm{poly}(\ell, \lambda), \quad |\mathsf{sk}_C| = \mathrm{poly}(\lambda), \quad |\mathsf{ct}| = \mathrm{poly}(\ell, \lambda).$$

We note that the kpABE scheme instantiated as above has succinct keys and ciphertexts. Our scheme achieves the same parameters as the unbounded depth KP-ABE scheme by [122] but does not make use of the circular evasive LWE assumption as they do.

CHAPTER 6

PSEUDORANDOM MULTI-INPUT FUNCTIONAL ENCRYPTION AND APPLICATIONS

6.1 INTRODUCTION

Multi-Input FE and $i\mathcal{O}$. A compelling extension of functional encryption, introduced by Goldwasser et al. is multi-input FE (MIFE) [101] where multiple parties can independently encrypt their data and the key generator can provide a function key that jointly decrypts all the ciphertexts. In more detail, now we have n parties, each of who independently computes the ciphertext for its data $\mathbf{x}_i$, for $i \in [n]$, the key generator provides a key for an n-ary function f and decryption allows to recover $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Evidently for real world applications of computing on encrypted data, it is highly desirable to be able to compute on data generated by different parties independently – this can enable important statistical functionalities such as running medical research algorithms on encrypted genomic or medical data. As discussed in the original work of MIFE, the notion is more meaningful in the symmetric rather than public key setting, since the latter allows for too much leakage on the challenge message by dint of legitimate combinations with messages chosen by the adversary. In two concurrent, influential works [29, 41] it was shown that single input FE, if it supports sufficiently expressive functionality and satisfies an efficiency property called *compactness*, is powerful enough to generically imply multi-input FE. We remark that for restricted functionalities, the compiler does not apply and multi-input FE for interesting restricted classes have been studied extensively [4, 81, 3, 75, 161, 2, 1, 141, 9, 8, 10, 23].

Aside from the real world applications of MIFE, the work of [29, 41] showed that sufficiently expressive MIFE can be used to construct the powerful notion of *Indistinguishability Obfuscation* ($i\mathcal{O}$), which seeks to garble circuits while preserving

their input-output behaviour. In more detail, given a circuit C, an obfuscation $\tilde{C}$ preserves the correctness of C so that $C(\mathbf{x}) = \tilde{C}(\mathbf{x})$ for *every* input $\mathbf{x}$, but hides everything else about C. This security property can be formalized in various ways, and one popular formalization asks that an adversary, given the obfuscation of C_b where b is a random bit and C_0 is functionally *equivalent* to C_1, cannot predict the value of b with non-negligible advantage.

While non-obvious in the beginning what such an object is useful for, a series of works has shown that $i\mathcal{O}$ can be used to instantiate a large number of advanced cryptographic primitives [90, 158, 67, 101, 118, 140, 39, 78, 92, 117, 125, 138]. Following this, there ensued a quest for constructing $i\mathcal{O}$ from reasonable cryptographic assumptions, with a large number of works coming closer and closer to the goal [91, 5, 20, 128, 95, 96, 171, 54, 53], until the beautiful work of Jain, Lin and Sahai [129] finally accomplished the goal.

Constructions from Lattice Assumptions. So far, the only constructions of compact FE/MIFE/$i\mathcal{O}$ from standard assumptions rely on pairings (together with LPN variants and/or low depth PRGs) [129, 130, 154] and are hence quantum insecure. Perhaps even more importantly, it is dissatisfying to have only a single pathway to constructing such a central object in the theory of cryptography. Casting around for other assumptions, hopefully with (conjectured) quantum security – the most obvious candidate is some assumption based on lattices. In particular, the Learning With Errors (LWE) assumption has proven to be amazingly versatile and provides elegant solutions to encrypted computation primitives such as fully homomorphic encryption or attribute based encryption [63, 52, 55, 99, 107, 43]. Thus the hope basing $i\mathcal{O}$ on LWE is natural, and unsurprisingly, has received a lot of attention. Several exciting candidates for compact FE and $i\mathcal{O}$ have been constructed from strengthenings of the LWE assumption [5, 20, 171, 96, 82] – unfortunately, these either rely on some heuristic, or their

underlying assumptions have been broken [120, 126].

Our Approach. Since full fledged FE/iO from lattice based assumptions has been elusive despite significant effort, a principled approach for making progress would be to restrict the functionality supported by FE/iO so that it is still meaningful for applications while also admitting a construction from some reasonable lattice assumption. A promising candidate assumption that interpolates the safety of LWE and the insecurity of iO or multilinear map assumptions is the recently introduced *evasive* LWE [169, 163]. At a very high level, evasive LWE can be seen as a lattice analog of the generic group model, which is popular in the pairings world, in that it restricts the class of attacks that an adversary can mount. Below, we let $\underline{X}$ denote a noisy version of X where the exact value of noise is not important and $\mathbf{B}^{-1}(\mathbf{P})$ denote a short preimage $\mathbf{K}$ (say) such that $\mathbf{BK} = \mathbf{P} \bmod q$. The evasive LWE assumption roughly says that if

$$(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{A}},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{B}},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{P}},\ \mathsf{aux}\ ) \approx_c (\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \$,\ \$,\ \$,\ \mathsf{aux})$$

where $\mathbf{A}, \mathbf{B}, \mathbf{P}$ are matrices of appropriate dimensions, $\mathbf{s}$ is a secret vector, aux is some auxiliary information and \$ represents random, then

$$(\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{A}},\ \underline{\mathbf{s}^\mathsf{T}\mathbf{B}},\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux}) \approx_c (\mathbf{A},\ \mathbf{B},\ \mathbf{P},\ \$,\ \$,\ \mathbf{B}^{-1}(\mathbf{P}),\ \mathsf{aux})$$

Evasive LWE has proven to be a very meaningful strengthening of LWE in that it has provided several strong new applications that had been elusive from plain LWE, despite significant research effort over decades – optimal broadcast encryption [169], witness encryption [164], multi-input attribute based encryption [22], optimal broadcast and trace [11], attribute based encryption (ABE) for unbounded depth circuits [122] and ABE for Turing machines [13], to name a few. Evasive LWE is considered plausible in "safe" regimes (discussed in detail in Section 5.1.3). Continuing this agenda, we ask:

Can we construct MIFE/iO *for a nontrivial functionality from evasive* LWE*?*

6.1.1 Our Results

We answer the above question in the affirmative and construct the first MIFE and $i\mathcal{O}$ for pseudorandom functionalities, namely functions where the output is pseudorandom for every input *seen by the adversary*. In the context of $i\mathcal{O}$ this means that the entire truth table must be pseudorandom, since the adversary can compute the functionality for any input. But for MIFE, it suffices to only restrict the output for the functions queried by the adversary. We denote such an MIFE by mi-prFE.

In more detail, we obtain the following results:

Theorem 6.1 (mi-prFE for constant arity)**.** *Assume evasive* LWE *and* LWE*. Then there exists a* mi-prFE *scheme for arity* $n = O(1)$*, supporting functions with (fixed) input length* L *and bounded polynomial depth.*[1]

For polynomial arity, we require a strengthening of evasive LWE, which we call non-uniform κ-evasive LWE. This is necessary to handle some delicate technical issues, which also appear in the proof of witness encryption from evasive LWE by [164], to the best of our understanding. We believe that a similar strengthening of evasive LWE should also be required for their proof, for the same reasons as ours. We then show:

Theorem 6.2 (mi-prFE for polynomial arity)**.** *Assume non-uniform* κ*-evasive* LWE*, non-uniform sub-exponential* PRF*, and non-uniform sub-exponential* LWE*. Then there exists a* mi-prFE *scheme for arity* $n = \mathrm{poly}(\lambda)$*, supporting functions with bounded polynomial depth.*

We bootstrap our mi-prFE to obtain the first $i\mathcal{O}$ for pseudorandom functionalities, similar to [29, 41], albeit via a different proof of security. We denote this by prIO. In more detail:

Theorem 6.3 (prIO)**.** *Assuming non-uniform* κ*-evasive* LWE*, non-uniform sub-exponential* PRF*, and non-uniform sub-exponential* LWE*, there exists a* prIO

[1] As is shown in [19], evasive LWE does not hold for the general samplers. When we invoke the assumption, we always invoke it with respect to a specific sampler class induced by the respective application. However, for ease of exposition, we will sometimes omit the reference to the specific sampler in the overview. Similar remarks can be applied to prFE and mi-prFE.

scheme for all polynomial sized circuits.

We also define a variant of prIO which only supports polynomial sized domains – we denote this variant by pPRIO. The advantage of considering this restricted variant is that we can base its security on (plain) evasive LWE, rather than it's strengthening.

Theorem 6.4. *Assuming evasive* LWE *and* LWE*, there exists a secure* pPRIO *scheme supporting circuits of bounded size.*

Applications.

We obtain several applications from our new tools of mi-prFE and prIO:

1. *Multi Input Predicate Encryption (*miPE*) for Constant Arity.* Assuming evasive LWE and LWE, there exists a mi-prFE scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth. The only prior work to support miPE with constant arity for function class P is by Agrawal et al. [22] and uses a strengthening of (non-standard) tensor LWE [168] together with evasive LWE.

2. *Multi Input Predicate Encryption for Polynomial Arity.* Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE, there exists a miPE scheme for arity $n = \text{poly}(\lambda)$, supporting functions of bounded polynomial depth. MIPE for polynomial arity supporting P was not known before, to the best of our knowledge.

3. *Two Party ID Based Key Exchange.* Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure two-party ID-NIKE scheme. This leads to the first ID-NIKE in the standard model without using multilinear maps or indistinguishability obfuscation.

4. *Instantiating the Random Oracle.* Hohenberger, Sahai and Waters [119] used iO to instantiate the random oracle in several applications. In more detail, they showed selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP) [37], the adaptive security of RSA FDH signatures [79], the selective security of BLS signatures, and the adaptive security of BLS signatures [45] in the standard model. Our prIO can be used to instantiate all these applications.

Additional Prior Work. The notion of iO for pseudorandom functionalities was considered implicitly by the work of Mathialagan, Peters and Vaikuntanathan [146] where they used subexponential LWE and evasive LWE to construct adaptively sound

zero-knowledge SNARKs for UP. In a previous version of their work [147], which was in private circulation and shared with us, this notion was defined explicitly and leveraged to obtain unlevelled fully homomorphic encryption. However, an explicit construction of iO for pseudorandom functionalities was not provided.

6.1.2 Technical Overview

Compact FE for Pseudorandom Functionalities. Our starting point is our companion work which provides FE for pseudorandom functionalities in the single input setting [12]. We recall the syntax here: The setup algorithm takes as input the security parameter λ and parameter prm, specifying the parameters of the function class, and outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function $f : \{0,1\}^{\mathrm{L}} \to \{0,1\}^{\ell}$ outputs a functional secret key sk_f. The encryption algorithm on input mpk and an input message $\mathbf{x} \in \{0,1\}^{\mathrm{L}}$ outputs a ciphertext ct. The decryption algorithm takes the functional secret key sk_f and ciphertext ct as input and outputs some $\mathbf{y} \in \{0,1\}^{\ell}$. Correctness requires that decryption should output $f(x)$ if the setup, encrypt and key generation algorithms were run honestly.

The definition of security, termed prCT security, says that so long as the *output* of the functionality is pseudorandom, the *ciphertext* is pseudorandom. In more detail, let Samp be a PPT algorithm that on input 1^{λ}, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries.

We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{f_i,\ f_i(x_j)\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\right) = 1] \\ - \mathsf{Pr}[\mathcal{A}_0(\ \mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1]\end{aligned} \tag{6.1}$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1] \\ - \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1] \tag{6.2}$$

where $(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathrm{Samp}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ satisfies prCT security if for every PPT Samp and $\mathcal{A}_1$, there exists another PPT $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \tag{6.3}$$

and $\mathsf{Time}(\mathcal{A}_0) \leq \mathsf{Time}(\mathcal{A}_1) \cdot Q(\lambda)$ for some polynomial $Q(\cdot)$. The work of [12] obtains the following result:

Theorem 6.5. *Assuming the LWE and evasive LWE assumptions, there exists a secure* prFE *scheme for function class* $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$ *satisfying* $|\mathsf{mpk}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda)$ *where* $\mathsf{dep} = \mathrm{poly}(\lambda)$ *is the depth bound on the functions supported by the scheme.*

We remark that their result is actually significantly more general – they construct the broader notion of *partially hiding* FE, where the ciphertext can additionally have a public attribute in addition to a private payload[2], with optimal parameters and can support circuits of unbounded depth. But bounded depth prFE is sufficient for the applications considered in the present work, so we restrict our attention to this.

Extending to the Multi-Input Setting. We extend the notion of prFE to multi-input setting and define a *secret-key* multi-input FE for pseudorandom functionalities $\mathsf{mi\text{-}prFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ for n-ary functions as follows: The setup algorithm on input 1^λ, arity 1^n and parameter prm, specifying the parameters

[2]The astute reader may wonder why regular FE does not imply partially hiding FE (PHFE) just by outputting the public attribute as part of the ciphertext – such an approach would make the ciphertext size grow with the length of public attribute which can be avoided in PHFE.

of the function class, outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function $f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}$ outputs a functional secret key sk_f. The i-th encryption algorithm on input msk and an input message $x_i \in \mathcal{X}_{\mathsf{prm}}$ outputs a ciphertext ct_i. The decryption algorithm on input secret key sk_f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ (corresponding to inputs $x_1, \ldots, x_n$ respectively) outputs some $y \in \mathcal{Y}_{\mathsf{prm}}$. The security has a similar flavor to the single-input setting, where we require that the ct is pseudorandom given the output of the function of encrypted input is pseudorandom– however it requires much care to accommodate the fact that there are exponentially many function evaluations even if only polynomially many input queries per slot are issued. We define the security as follows.

Let $\kappa = \kappa(\lambda)$ be a function in λ and Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(\{f_k\}_{k\in[q_0]}, \{x_1^{j_1}\}_{j_1\in[q_1]}, \ldots, \{x_n^{j_n}\}_{j_n\in[q_n]}, \mathsf{aux} \in \{0,1\}^*\right)$$

where q_0 is the number of key queries and q_i is the number of encryption queries for the i-th slot. We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\mathrm{def}}{=} &\mathsf{Pr}[\mathcal{A}_0(1^\kappa, f_1, \ldots, f_{q_0}, \{f_k(x_1^{j_1}, \ldots, x_n^{j_n})\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}, \mathsf{aux}) = 1] \\ &- \mathsf{Pr}[\mathcal{A}_0(1^\kappa, f_1, \ldots, f_{q_0},\ \{\Delta_{k,j_1,\ldots,j_n} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}, \mathsf{aux}) = 1]\end{aligned}$$

$$\begin{aligned}\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\mathrm{def}}{=} &\mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ f_1, \ldots, f_{q_0}, \{\mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i})\}_{i\in[n], j_i\in[q_i]}, \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{q_0}},\ \mathsf{aux}) = 1] \\ &- \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk}, f_1, \ldots, f_{q_0}, \{\delta_i^{j_i} \leftarrow \mathsf{Sim}(\mathsf{msk})\}_{i\in[n], j_i\in[q_i]}, \mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{q_0}}, \mathsf{aux}) = 1]\end{aligned}$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$ for $k \in [q_0]$.

We say that a mi-prFE scheme is secure if for every PPT Samp, $\mathcal{A}_1$, and Sim there exists another PPT $\mathcal{A}_0$ and a polynomial $p(\cdot)$ such that

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/p(\kappa) - \mathsf{negl}(\kappa), \quad \mathsf{Time}(\mathcal{A}_0) \leq p(\kappa) \cdot \mathsf{Time}(\mathcal{A}_1).$$

The parameter κ above is introduced to adjust the strength of the requirement for the

precondition. By default, we require $\kappa \geq \lambda^n$ since the input length to the distinguisher is polynomial in λ^n anyway and this condition should be fulfilled for the above equations to to make sense. If we need κ to be larger, this strengthens the requirement for the precondition, as it means we want the distributions in the pre-condition to be indistinguishable against an adversary with a longer running time.[3] Ideally, we want κ to be as small as λ^n to make the requirement weaker. Looking ahead, for our construction of $\mathsf{mi\text{-}prFE}$ scheme supporting polynomial arity $n = \mathrm{poly}(\lambda)$, we require large κ as an artifact of the security proof techniques. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

Construction. Next, we describe our construction for bounded depth $\mathsf{mi\text{-}prFE}$ using a bounded depth prFE and a secret-key encryption scheme. Our construction adapts the key idea from [29], of "unrolling" ciphertexts on the fly via recursive decryption. However, since our security notion is quite different, our proof departs significantly from theirs, as we will discuss below.

Specifically, we use n instances of a single-input prFE scheme $\{\mathsf{prFE}_i\}_{i\in[n]}$, with appropriate input lengths, to build a n arity $\mathsf{mi\text{-}prFE}$ scheme where the i-th encryption algorithm Enc_i outputs the prFE_{i+1} functional secret-key, $\mathsf{prFE}_{i+1}.\mathsf{sk}$, for $i \in [n-1]$ and Enc_n outputs a ciphertext corresponding to prFE_n scheme. Here the $\mathsf{prFE}_{i+1}.\mathsf{sk}$ contains the input $\mathbf{x}_i$ hardcoded within itself, wrapped in an SKE scheme, since prFE does not support function hiding. It computes the ciphertext $\mathsf{prFE}_i.\mathsf{ct}$ for the input $(\mathsf{SKE.sk}, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ decryptable by $\mathsf{prFE}_i.\mathsf{sk}$ which in turn computes the ciphertext $\mathsf{prFE}_{i-1}.\mathsf{ct}$ decryptable by $\mathsf{prFE}_{i-1}.\mathsf{sk}$ and so on. Now, note that the decryption of slot n ciphertext with slot $n-1$ functional secret-key will give us a ciphertext decryptable by functional key at slot $n-2$. Unrolling upto slot 1, we get a ciphertext, $\mathsf{prFE}_1.\mathsf{ct}$, corresponding to prFE_1 scheme. Finally, the key generation algorithm outputs a functional secret-key for prFE_1 which together with $\mathsf{prFE}_1.\mathsf{ct}$ will give us the desired

[3] Recall that $\mathcal{A}_1$ is a PPT algorithm. This means that it runs in polynomial time in its input length. Here, κ serves as a "padding", which artificially makes the input longer and allows $\mathcal{A}_1$ to run in longer time.

output.

In more detail[4],

1. The setup algorithm generates n instances of prFE scheme $\{\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{msk}\}$ for appropriate input lengths and a secret key corresponding to SKE scheme $\mathsf{SKE.sk}$. It outputs $\mathsf{mpk} = (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and
$\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.

2. The key generation algorithm on input msk and a function $f : (\{0,1\}^{\mathrm{L}})^n \to \{0,1\}$, computes $\mathsf{prFE}_1.\mathsf{sk}_f \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f)$ and outputs $\mathsf{sk}_f \coloneqq \mathsf{prFE}_1.\mathsf{sk}_f$.

3. The i-th encryption algorithm on input $(\mathsf{msk}, \mathbf{x}_i \in \{0,1\}^{\mathrm{L}})$, parses $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and does as follows. If $i \in [n-1]$

   - Compute $\mathsf{SKE.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i)$.

   - Define function $\mathrm{F}_i \coloneqq \mathrm{F}_i[\mathsf{SKE.ct}_i, \mathsf{prFE}_i.\mathsf{mpk}]$ which on input $(\mathsf{SKE.sk}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ first computes $\mathbf{x}_i = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i)$ and then computes a $\mathsf{prFE}_i.\mathsf{ct}$ encoding $(\mathsf{SKE.sk}, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ if $i \neq 1$ else it computes $\mathsf{prFE}_1.\mathsf{ct}$ encoding $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$[5]. It outputs $\mathsf{prFE}_i.\mathsf{ct}$.

   - It computes a functional key for F_i using the $i+1$-th instance of prFE and outputs it as the i-th ciphertext, i.e., $\mathsf{ct}_i \coloneqq \mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i)$.

   If $i = n$, it outputs $\mathsf{ct}_n \coloneqq \mathsf{prFE}_n.\mathsf{ct} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n))$.

4. The decryption algorithm on input $\mathsf{sk}_f = \mathsf{prFE}_1.\mathsf{sk}_f$, and ciphertexts $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk}$ for $i \in [n-1]$, and $\mathsf{ct}_n = \mathsf{prFE}_n.\mathsf{ct}$ does the following: (a) Iteratively compute $\mathsf{prFE}_{i-1}.\mathsf{ct}$ for $i \in [2, n]$ by decrypting $\mathsf{prFE}_i.\mathsf{ct}$ with $\mathsf{prFE}_i.\mathsf{sk}$ starting with $i = n$. (b) Compute and output $\mathbf{y} \leftarrow \mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct})$.

Correctness follows from the correctness of underlying ingredients. To see this, note that by the correctness of prFE_n and the definition of F_{n-1}, we have $\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct})$ will output $\mathrm{F}_{n-1}(\mathsf{SKE.sk}, \mathbf{x}_n)$ correctly. Next, by the correctness of the SKE scheme, we have $\mathbf{x}_{n-1} = \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE}, \mathsf{ct}_{n-1})$ thus $\mathrm{F}_{n-1}(\mathsf{SKE.sk}, \mathbf{x}_n) = \mathsf{prFE}_{n-1}.\mathsf{ct}$ where

[4] We omit substantial notation here for the ease of readability.

[5] The randomness for computing the ciphertexts comes from a PRF, which we omit here in the overview.

$\mathsf{prFE}_{n-1}.\mathsf{ct}$ encodes

$(\mathsf{SKE.sk}, \mathbf{x}_{n-1}, \mathbf{x}_n)$. Unrolling as in decryption step (a), we get $\mathsf{prFE}_1.\mathsf{ct}$ which encodes $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$. Now, from step (b) of decryption and correctness of prFE_1 scheme we have $\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Security. While the key-idea of our construction is adapted from [29], our security proof differs significantly from theirs as we elaborate next. Consider the initial view of an adversary $\mathcal{A}$ which outputs q_0 key queries $\{f_1, \ldots, f_{q_0}\}$, q_i input queries for i-th slot $\{x_1^{j_1}\}_{j_1 \in [q_1]}, \ldots, \{x_n^{j_n}\}_{j_n \in [q_n]}$ and auxiliary information $\mathsf{aux}_{\mathcal{A}}$

$$\mathcal{D}_{0,0} : \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{ct}_i^{j_i} = \mathsf{SKE.ct}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \\ \left\{\mathsf{ct}_n^{j_n} = \mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]} \end{array} \right)$$

To prove security we design a simulator Sim as follows: On input $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$

- for $i \in [n-1]$, it first samples a random SKE ciphertext γ_i from the ciphertext space of SKE scheme, defines $\mathrm{F}_i[\gamma_i, \mathsf{prFE}_i.\mathsf{mpk}]$ as in the construction and outputs $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i, \mathsf{prFE}_i.\mathsf{mpk}])$.
- for $i = n$, it outputs a randomly sampled ct_n from the ciphertext space of prFE_n scheme.

Given the above simulator it suffices to show that Section 6.1.2 is indistinguishable from the following distribution

$$\mathcal{D}_{0,1} : \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{ct}_i^{j_i} = \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \\ \left\{\delta^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n}\right\}_{j_n\in[q_n]} \end{array} \right)$$

At a high level, the security proof proceeds as follows. To prove the pseudorandomness of $\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\}_{j_n}$, we show that the decryption results of these ciphertexts using the secret keys $\{\mathsf{prFE}_n.\mathsf{sk}^{j_{n-1}}\}_{j_{n-1}}$ are all pseudorandom. This allows us to invoke the security of

prFE_n. The decryption results of the above ciphertexts using the secret keys are ciphertexts $\{\mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}\}_{j_{n-1},j_n}$ of prFE_{n-1} and we would like to prove the pseudorandomness of them. We again consider the decryption results of the ciphertexts using the secret keys $\{\mathsf{prFE}_{n-1}.\mathsf{sk}^{j_{n-2}}\}_{j_{n-2}}$. This process continues until we reach the point where we have to prove the pseudorandomness of $\{\mathsf{prFE}_1.\mathsf{ct}^{j_1,\cdots,j_n}\}_{j_1,\ldots,j_n}$, where each ciphertext encodes $(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})$. By invoking the security of prFE_1 once again, we can conclude that it suffices to show that $\{f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\}_{k,j_1,\ldots,j_n}$ are pseudorandom even given SKE ciphertexts encrypting each $\mathbf{x}_i^{j_i}$, where the latter is dealt as auxiliary information throughout the process of the above recursive invocations of prFE security. We then invoke the security of SKE to erase the information of $\mathbf{x}_i^{j_i}$ from the SKE ciphertexts. This allows us to conclude, since the pseudorandomness of $\{f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\}_{k,j_1,\ldots,j_n}$ directly follows from the precondition.

A bit more formally, to prove Section 6.1.2$\approx$ Section 6.1.2 we begin by invoking the security of prFE_n with sampler that provides inputs $\{(\mathsf{SKE.sk},\mathbf{x}_n^{j_n})\}_{j_n\in[q_n]}$, functions $\{\mathrm{F}_{n-1}^{j_{n-1}}[\mathsf{SKE.ct}_{n-1}^{j_{n-1}},\mathsf{prFE}_{n-1}.\mathsf{mpk}]\}_{j_{n-1}\in[q_{n-1}]}$, and all the remaining components of Section 6.1.2 as auxiliary information. Now, from the security guarantee of prFE_n with the sampler, we know that to prove $\mathsf{prFE.ct}_n^{j_n}$ is pseudorandom it suffices to show function output

$$\begin{aligned}&\mathrm{F}_{n-1}^{j_{n-1}}[\mathsf{SKE.ct}_{n-1}^{j_{n-1}},\mathsf{prFE}_{n-1}.\mathsf{mpk}](\mathsf{SKE.sk},\mathbf{x}_n^{j_n})\\&=\mathsf{prFE}_{n-1}.\mathsf{Enc}(\mathsf{prFE}_{n-1}.\mathsf{mpk},(\mathsf{SKE.sk},\mathbf{x}_{n-1}^{j_{n-1}},\mathbf{x}_n^{j_n}))=\mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}\end{aligned}$$

is pseudorandom for all $j_{n-1}\in[q_{n-1}]$, $j_n\in[q_n]$. Thus it suffices to show

$$\mathcal{D}_{1,0}:\left(\begin{array}{c}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n-1]},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\mathsf{SKE.ct}_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-2],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{n-1}.\mathsf{ct}^{j_{n-1},j_n}\right\}_{\substack{j_{n-1}\in[q_{n-1}],\\ j_n\in[q_n]}}\end{array}\right)$$

$$\approx$$

$$\mathcal{D}_{1,1}:\left(\begin{array}{c}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n-1]},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\gamma_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-2],\\ j_i\in[q_i]}}\left\{\delta^{j_{n-1},j_n}\right\}_{\substack{j_{n-1}\in[q_{n-1}],\\ j_n\in[q_n]}}\end{array}\right)$$

where $\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}$, $\delta^{j_{n-1},j_n}\leftarrow\mathcal{CT}_{\mathsf{prFE}_{n-1}}$, and $\mathcal{CT}_{\mathsf{SKE}}$ and $\mathcal{CT}_{\mathsf{prFE}_{n-1}}$ denotes the ciphertext space of the SKE scheme and prFE_{n-1} scheme, respectively. Recursively invoking the security of prFE_i for $i=n-1,\ldots,2$, it suffices to show the following:

$$\mathcal{D}_{n-1,0}:\left(\begin{array}{c}1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{prFE}_1.\mathsf{mpk},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\mathsf{prFE}_1.\mathsf{ct}^{j_1,\ldots,j_n}\right\}_{j_1\in[q_1],\ldots,j_n\in[q_n]}\end{array}\right)$$

$$\approx$$

$$\mathcal{D}_{n-1,1}:\left(\begin{array}{c}1^\kappa,\mathsf{aux}_{\mathcal{A}},\ \mathsf{prFE}_1.\mathsf{mpk},\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]},\\ \left\{\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\left\{\delta^{j_1,\ldots,j_n}\leftarrow\mathcal{CT}_{\mathsf{prFE}_1}\right\}_{j_1\in[q_1],\ldots,j_n\in[q_n]}\end{array}\right).$$

Finally, applying the security of prFE_1 once again, we can see that it suffices to show the following:

$$\mathcal{D}_{n,0}:\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n},\left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i}\leftarrow\mathsf{SKE}.\mathsf{Enc}(\mathsf{SKE}.\mathsf{sk},\mathbf{x}_i^{j_i})\right\}_{i,j_i}\right)$$

$$\approx_c$$

$$\mathcal{D}_{n,1}:\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{\ f_k,\ \Delta_k^{j_1,\ldots,j_n}\leftarrow\{0,1\}\right\}_{k,j_1,\ldots,j_n},\ \left\{\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right)\tag{6.4}$$

Here, 1^κ appearing in the above distributions is introduced for compensating the blow up of the size of the adversary caused by the multiple invocations of the prFE security. The detail is not important here and we refer to Section 6.3 for the detail.

To prove Equation (6.4), we first invoke the security of SKE scheme to show the pseudorandomness of SKE ciphertexts. This erases the information of $\mathbf{x}_i^{j_i}$ from $\mathsf{SKE}.\mathsf{ct}_i^{j_i}$. Next, we use the fact that the functionality supported by our scheme is pseudorandom to argue that the function values $f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})$ are pseudorandom.

Subtleties in the proof. The high level overview presented above hides many important details, and indeed, as stated is not secure. There are two important subtleties that arise when making the formal argument, and these are so significant that they require us to strengthen the underlying evasive LWE assumption. Moreover, to the best of our understanding, these issues also arise in prior work [164] and fixing them there also requires to strengthen the evasive LWE assumption as we do below.

We note that at a high level, the overall structure of our security proof above is similar to that of witness encryption in [164]. In both proofs, the main step considers parameterized distributions $\{\mathcal{D}_{h,b}\}_{h\in[n,0],b\in\{0,1\}}$ and shows that $\mathcal{D}_{0,0}$ and $\mathcal{D}_{0,1}$ are indistinguishable if $\mathcal{D}_{n,0} \approx_c \mathcal{D}_{n,1}$ even with subexponentially small advantage against subexponential time adversary. To show this claim, [164] uses the evasive LWE assumption, while we use the security of the prFE instances, which in turn is reduced to the evasive LWE assumption. While this difference stems simply from the fact that we introduce the intermediate primitive of prFE to construct $\mathsf{mi\text{-}prFE}$ instead of directly constructing it from evasive LWE, there are more fundamental differences as well. In particular, we identify certain subtle issues in the proof by [164] and fix these by strengthening the assumptions. We elaborate on this below. To begin, we formally define the evasive LWE assumption.

Evasive LWE. Let Samp be a PPT algorithm that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux})$ on input 1^λ. For PPT adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$, we define the following advantage functions:

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathsf{Adv}_0(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) = 1] - \mathsf{Pr}[\mathsf{Adv}_0(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}) = 1] \tag{6.5}$$

$$\mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Pr}[\mathsf{Adv}_1(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1] - \mathsf{Pr}[\mathsf{Adv}_1(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}) = 1] \tag{6.6}$$

where $(\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathsf{Samp}(1^\lambda)$, $\mathbf{B}, \mathbf{C}_0, \mathbf{C}'$ are uniform matrices of appropriate dimensions, $\mathbf{E}, \mathbf{E}'$ are low norm Gaussian error matrices, and $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P})$.

We say that the *evasive* LWE (EvLWE) assumption holds if for every PPT Samp and Adv_1, there exists another PPT Adv_0 and a polynomial $Q(\cdot)$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda) - \mathsf{negl}(\lambda) \text{ and } \mathsf{Time}(\mathcal{A}_0) \leq Q(\lambda) \cdot \mathsf{Time}(\mathcal{A}_1). \quad (6.7)$$

Issue 1: On the multiplicative invocation of evasive LWE. To prove $\mathcal{D}_{0,0} \approx_c \mathcal{D}_{0,1}$, [164] assumes that there exists an adversary $\mathcal{A}_0$ that distinguishes them with non-negligible advantage ϵ and polynomial time t for the sake of contradiction. They then invoke the evasive LWE assumption with respect to an appropriately defined sampler Samp_1 to conclude that there exists a distinguishing adversary $\mathcal{A}_1$ against $\mathcal{D}_{1,0}$ and $\mathcal{D}_{1,1}$. This process continues multiple times, where they invoke evasive LWE with respect to the security parameter $\lambda_j := 2^j\lambda$ and an adversary $\mathcal{A}_j$ for the j-th invocation to obtain another adversary $\mathcal{A}_{j+1}$, where $\mathcal{A}_k$ is a distinguisher against $\mathcal{D}_{k,0}$ and $\mathcal{D}_{k,1}$. Denoting the distinguishing advantage against $\mathcal{D}_{j,0}$ and $\mathcal{D}_{j,1}$ of $\mathcal{A}_j$ by ϵ_j, we have $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}_j(\lambda_j)$, where poly_j is a polynomial that is determined by the sampler Samp_j. Finally, they obtain a distinguisher $\mathcal{A}_n$ against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$, where $\epsilon_n = \epsilon/\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$ and the running time being $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$. They derive the conclusion by saying

$$\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n) = \mathrm{poly}(2^{n^2}\lambda^n) \quad (6.8)$$

and setting the parameters so that there is no adversary of this running time and distinguishing advantage against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$. However, a subtlety is that $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n)$ is not necessarily true. For example, one can consider the setting where we have $\mathrm{poly}_j(\lambda) = \lambda^{2^j}$. This example may look a bit artificial, but it does not contradict the evasive LWE assumption, since j is treated as a constant asymptotically. In words, the issue arises from the fact that even if each

polynomial has a constant exponent, the maximum of the exponents can be arbitrarily large function in λ, when we consider non-constant number of polynomials. In this setting, $\mathcal{A}_n$'s distinguishing advantage is too small to derive the contradiction.

The above issue occurs due to the invocations of evasive LWE super-constant times. To resolve the problem, we consider non-uniform sampler $\{\mathrm{Samp}_{h^*}\}_{h^*}$ that hardwires the "best" index h^* and invoke the evasive LWE[6] only with respect to this sampler. In more detail, to argue that the final distributions $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$ are indistinguishable, it is required that *all* pairs of distributions $\mathcal{D}_{j,0}$ and $\mathcal{D}_{j,1}$ be indistinguishable. Then, to arrive at a contradiction, it suffices to find even one intermediate pair (for index h^*) which is distinguishable – we invoke evasive LWE with respect to that. We avoid the problem of incurring security loss of polynomial with arbitrarily large exponent, since we invoke the evasive LWE only once w.r.t a single sampler in the proof. However, this solution entails the strengthening of the assumption where we consider non-uniform samplers. We believe that the same strengthening of the assumption is required for the proof in [164] as well.

Issue 2: On the additive term of evasive LWE. We believe there is another subtle issue that arises in the proof by [164]. To focus on the issue, we ignore the first issue discussed above and assume $\mathrm{poly}_j(\lambda) = \lambda^c$ holds for some fixed $c \in \mathbb{N}$ that does not depend on j, which makes Equation (6.8) correct. In the proof of [164] (and in our explanation above), we implicitly ignore the negligible additive term when applying evasive LWE. Namely, when we apply the assumption with respect to $\mathcal{A}_j$, the lower bound for the advantage ϵ_{j+1} of $\mathcal{A}_{j+1}$ should be $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j) - \mathsf{negl}(\lambda_j)$ rather than $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j)$. This does not cause any difference when we consider the setting where ϵ_j is non-negligible in λ_j. However, for larger j, ϵ_j is negligible function in the security parameter λ_j. Concretely, the lower bound on ϵ_{n-1} obtained by ignoring the

[6]To be precise, in our context, we invoke the security of prFE rather than evasive LWE. However, since the same issue arises both in our context and the context of witness encryption [164] and the security of prFE is eventually reduced to the evasive LWE in our context, we intentionally do not distinguish the invocation of evasive LWE and prFE security here.

additive term is $\epsilon/\mathrm{poly}(2^{(n-1)^2}\lambda^{n-1})$. If we apply evasive LWE once more with respect to $\lambda_n = 2^n\lambda^n$ to complete the proof, we have $\epsilon_n \geq \epsilon_{n-1}/\mathrm{poly}(\lambda_n) - \mathsf{negl}(\lambda_n)$. The RHS of the inequality may be negative, since $\epsilon_{n-1}/\mathrm{poly}(\lambda_n)$ is some specific negligible function in λ_n and this may be smaller than the second term $\mathsf{negl}(\lambda_n)$. Therefore, what we can derive here is the trivial bound $\epsilon_n \geq 0$, which is not enough for our purpose. To fix this issue, we introduce additional parameter κ and then modify the assumption so that the additive term is negligibly small in κ, which is set much larger than λ. Again, we believe that the same strengthening of the assumption is required for the proof in [164] as well. We define the strengthened version of evasive LWE next.

Non-Uniform κ-Evasive LWE. Let $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux})$ as above. For non-uniform adversaries $\mathcal{A}_0 = \{\mathcal{A}_{0,\lambda}\}_\lambda$ and $\mathcal{A}_1 = \{\mathcal{A}_{1,\lambda}\}_\lambda$, we define the advantage functions $\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda)$ and $\mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)$ as in Equation (6.5) and Equation (6.6), respectively. For a function $\kappa := \kappa(\lambda)$ of the security parameter λ, we say that the non-uniform κ-evasive LWE assumption holds if for every non-uniform sampler Samp and a non-uniform adversary Adv_1 whose sizes are polynomial in λ' and κ respectively for $\lambda' := \lambda'(\lambda) < \kappa(\lambda)$, there exists another non-uniform adversary Adv_0 and a polynomial $Q(\cdot)$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1). \tag{6.9}$$

There are two main differences from the standard EvLWE assumption: (i) we consider non-uniform sampler and adversaries, (ii) it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ that appears in Equation (6.7) is replaced by $\mathsf{negl}(\kappa)$ in Equation (6.9). These changes are introduced so that our prFE can satisfy a stronger notion of security, as discussed below. Note that in the case λ' is superpolynomial in λ, $\mathsf{Samp}(1^\lambda)$ outputs $\mathbf{S}$, $\mathbf{P}$, and aux whose sizes are polynomial in λ' and thus superpolynomial in λ. We require the above assumption with $\kappa = 2^{\mathrm{poly}(\lambda)}$ in our construction.

prFE with Non-Uniform κ Security. We now describe a stronger notion of security from prFE, which we need for our mi-prFE compiler.

We say that a prFE scheme is secure in the non-uniform κ setting if for every Samp and an adversary $\mathcal{A}_1$ whose sizes are polynomial in λ' and κ respectively for $\lambda' := \lambda'(\lambda) < \kappa(\lambda)$, there exists another adversary $\mathcal{A}_0$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \tag{6.10}$$

where the advantage functions $\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda)$ and $\mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)$ are defined as in Equation (6.1) and Equation (6.2) respectively, and $\mathrm{Size}(\mathcal{A}_0) \leq \mathrm{Size}(\mathcal{A}_1) \cdot Q(\lambda')$ for some polynomial $Q(\cdot)$.

The new security notion differs from the previous one in two ways: (i) it considers non-uniform adversaries instead of uniform adversaries, (ii) it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ that appears in Equation (6.3) is replaced by $\mathsf{negl}(\kappa)$ in Equation (6.10). By taking κ asymptotically larger than λ (e.g., $\kappa := \lambda^\lambda$), we can make the additive term $\mathsf{negl}(\kappa)$ much smaller than $\mathsf{negl}(\lambda)$.

Fortunately, we can show, with some careful adjustments, that the prFE scheme from [12] also satisfies this stronger notion of security. In Section 6.A.1, we prove the following theorem.

Theorem 6.6. *Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE, subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE, there exists a* prFE *scheme for function class $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$ satisfying κ-*prCT *security as per Definition 6.3 with efficiency*

$$|\mathsf{mpk}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ is the depth bound on the functions supported by the scheme.

With the above modifications, we are ready to state our final theorem:

Theorem 6.7 (mi-prFE for poly arity)**.** *Let* $\kappa = \lambda^{n^2 \log \lambda}$. *Assume non-uniform* κ*-evasive LWE, non-uniform sub-exponential* PRF*, and non-uniform sub-exponential LWE. Then there exists a* mi-prFE *scheme for arity* $n = \mathrm{poly}(\lambda)$*, supporting functions with input length* L *and bounded polynomial depth* $d = d(\lambda)$*, and satisfying* κ*-security (Definition 6.10) with efficiency* $|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$, $|\mathsf{ct}_1| = nL\mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}_i| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$ *for* $i \in [2, n]$ *where* $\Lambda := (n^2\lambda)^{1/\delta}$.

Since the aforementioned issues in the proof only arise when evasive LWE is applied super-constant number of times, we do not need the stronger version of evasive LWE to support constant arity. Hence we get:

Theorem 6.8 (mi-prFE for constant arity)**.** *Let* $\kappa = \lambda^n$. *Assume evasive LWE and LWE. Then there exists a* mi-prFE *scheme for arity* $n = O(1)$*, supporting functions with input length* L *and bounded polynomial depth* $d = d(\lambda)$*, and satisfying* κ*-security (Definition 6.10) with efficiency* $|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \lambda)$, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$, $|\mathsf{ct}_1| = nL \cdot \mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}_i| = \mathrm{poly}(n, L, d, \lambda)$ *for* $i \in [2, n]$.

***iO* for Pseudorandom Functionalities.** We now use our multi-input FE to construct *iO*, à la AJ/BV [29, 41] for the same functionality. We begin with the definition of *iO* for pseudorandom functionalities, which we refer to as prIO. Recall that this notion was first defined by Mathialagan, Peters and Vaikuntanathan [146]. The syntax of prIO is as in regular *iO*, where we have (i) an obfuscation algorithm which takes as input the security parameter λ and a circuit C and outputs an obfuscated circuit $\tilde{C}$, (ii) an evaluation algorithm takes as input an obfuscated circuit $\tilde{C}$ and an input x. It outputs $y = C(x)$. We also require that the evaluation time of the obfuscated circuit be only polynomially slower than the run time of the circuit C on x.

Our notion of security however, differs from the standard notion considered in the literature of *iO* and is specified as follows. For the security parameter $\lambda = \lambda(\lambda)$, let Samp

be a PPT algorithm that on input 1^λ, outputs

$$(C_0, C_1,\ \mathsf{aux} \in \{0,1\}^*)$$

where $C_0 : \{0,1\}^n \to \{0,1\}^m$ and $C_1 : \{0,1\}^n \to \{0,1\}^m$ have the same description size. We then require that

$$\begin{aligned} \text{If } \left(1^\kappa, \{C_0(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) &\approx_c \left(1^\kappa, \{\Delta_x \leftarrow \{0,1\}^m\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \\ &\approx_c \left(1^\kappa, \{C_1(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \qquad (6.11) \\ \text{then } \left(i\mathcal{O}(1^\lambda, C_0), \mathsf{aux}\right) &\approx_c \left(i\mathcal{O}(1^\lambda, C_1), \mathsf{aux}\right) \qquad (6.12) \end{aligned}$$

where the parameter κ above is introduced to adjust the strength of the requirement for the precondition, similarly to the case of mi-prFE. Roughly speaking, the above security definition says that the obfuscations of two circuits with pseudorandom truth tables are indistinguishable.

Our construction follows the blueprint of the multi-input FE to $i\mathcal{O}$ conversion by Ananth and Jain [29]. Briefly, the obfuscation of a circuit $C : \{0,1\}^n \to \{0,1\}^m$ using a $(n+1)$ input mi-prFE scheme is

$$\{\mathsf{mi\text{-}prFE.sk}_U, \mathsf{mi\text{-}prFE.ct}_{1,0}, \mathsf{mi\text{-}prFE.ct}_{1,1}, \ldots, \mathsf{mi\text{-}prFE.ct}_{n,0}, \mathsf{mi\text{-}prFE.ct}_{n,1}, \mathsf{mi\text{-}prFE.ct}_{n+1,C}\}$$

where $\mathsf{mi\text{-}prFE.sk}_U$ is the mi-prFE functional key corresponding to the universal circuit U such that $U(x_1, \ldots, x_n, C) = C(x_1, \ldots, x_n)$, $\mathsf{mi\text{-}prFE.ct}_{i,b}$ for $i \in [n], b \in \{0,1\}$ denotes the i-th slot mi-prFE ciphertext encoding bit b, and $\mathsf{mi\text{-}prFE.ct}_{n+1,C}$ denotes the $(n+1)$-th slot mi-prFE ciphertext encoding the circuit C. The evaluation algorithm on input $\mathbf{x} = (x_1, \ldots, x_n)$ runs $\mathsf{mi\text{-}prFE.Dec}(\mathsf{mi\text{-}prFE.sk}_U, \mathsf{mi\text{-}prFE.ct}_{1,x_1}, \ldots, \mathsf{mi\text{-}prFE.ct}_{n,x_n}, \mathsf{mi\text{-}prFE.ct}_{n+1,C})$.

Correctness as well as security follow from those of mi-prFE. This leads to the following theorem, shown in Section 6.5.

Theorem 6.9. *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE, non-uniform*

sub-exponential PRF*, and non-uniform sub-exponential LWE, there exists a* prIO *scheme for circuits.*

***iO* for Pseudorandom Functionalities with Polynomial Domains.** We also define a variant of prIO which only supports polynomial sized domains. We denote this variant by pPRIO. The advantage of considering this restricted variant is that we can base the security of the construction on (plain) evasive LWE, rather than non-uniform κ-variant of it. For applications we consider stronger variant of the primitive where we decompose the obfuscation algorithm into online-offline parts and consequently define a reusable security for this variant. In more detail, we show the following:

Theorem 6.10. *Assuming evasive LWE and LWE, there exists a secure* pPRIO *scheme supporting circuits of bounded size* $L = \mathrm{poly}(\lambda)$ *with* $|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), |\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda)$*, where* $\mathsf{Obf}_{\mathsf{off}}$ *and* $\mathsf{Obf}_{\mathsf{on}}$ *refer to the offline and online part of the obfuscated program, respectively.*

We refer the reader to Section 6.6 for details. As discussed in Section 6.1, we use this construction in our companion paper [12] to construct ABE for optimal parameters.

Applications. We show that our new tool of MIFE/*iO* for pseudorandom functionalities is quite powerful and yields several interesting applications.

Multi-Input Predicate Encryption for Circuits. A multi-input predicate encryption (miPE) scheme [25] for n-ary functions $\mathsf{miPE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ generalizes single input predicate encryption [109] to support multiple encryptors, who each encrypt their data with independently chosen randomness. In miPE, the setup algorithm on input 1^λ, arity 1^n and parameter prm, specifying the parameters of the function class, outputs $(\mathsf{mpk}, \mathsf{msk})$. The key generation algorithm on input msk and a function

$f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}$ outputs a functional secret key sk_f. The i-th encryption algorithm on input msk, an attribute $x_i \in \mathcal{X}_{\mathsf{prm}}$ and a message $\mu_i \in \{0,1\}$ outputs a ciphertext ct_i. The decryption algorithm on input secret key sk_f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ (corresponding to inputs $(x_1, \mu_1) \ldots, (x_n, \mu_n)$ respectively) outputs a string $\mu' \in \{0,1\}^n \cup \perp$.

We prove the following security guarantee for miPE: Consider an adversary $\mathcal{A}$ which outputs q_0 key queries $\{f_0, \ldots, f_{q_0}\}$, q_i pairs of attribute-message queries for i-th slot $\{(x_1^{j_1}, \mu_1^{j_1})\}_{j_1 \in [q_1]}, \ldots, \{(x_n^{j_n}, \mu_n^{j_n})\}_{j_n \in [q_n]}$ and auxiliary information $\mathsf{aux}_{\mathcal{A}}$. We say that a miPE scheme is secure if the following holds for the adversary $\mathcal{A}$

$$\left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i}, \mu_i^{j_i})\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}_{\mathcal{A}}\right)$$
$$\approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k \in [q_0]}, \left\{\delta_i^{j_i} \leftarrow \mathsf{Sim}(\mathsf{msk})\right\}_{i \in [n], j_i \in [q_i]}, \mathsf{aux}_{\mathcal{A}}\right)$$

given $f_k(x_1^{j_1}, \ldots, x_n^{j_n}) = 0$ for every $i \in [n], j_i \in [q_i]$, and $k \in [q_0]$, where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$.

Previously, the works of [25, 87] defined the notion of multi-input predicate encryption and provided the first constructions for specific functionalities. The follow-up work of Agrawal et al. [22] supported the most general functionality – it allowed to compute arbitrary predicates in P on vector $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i$ is encrypted by party $i \in [n]$ and k is a constant. The miPE of this work is obtained by first building a multi-input ABE, which is then compiled into miPE using a generic compiler by [25]. Their multi-input ABE for constant arity is quite complex and leverages intricate algebraic properties of the underlying building blocks. Moreover, the limitation for a constant k seems inherent to their techniques, since the parameters grow exponentially in n. In contrast, our construction is extremely simple and can bootstrap a single-input PE scheme to a polynomial-input one generically by simply using an $\mathsf{mi\text{-}prFE}$ to generate PE ciphertext using randomness jointly chosen by the encryptors. The PE must have pseudorandom ciphertext so as to be suitable for the compiler but this is a relatively mild property and

readily satisfied by known constructions [109]. In more detail, our construction works as follows.

- The setup generates a $\mathsf{mi\text{-}prFE}$ instance $(\mathsf{mi\text{-}prFE.msk}, \mathsf{mi\text{-}prFE.mpk})$ and a single-input PE instance $(\mathsf{PE.msk}, \mathsf{PE.mpk})$. It outputs $\mathsf{msk} = (\mathsf{mi\text{-}prFE.msk}, \mathsf{PE.msk})$ and $\mathsf{mpk} = (\mathsf{mi\text{-}prFE.mpk}, \mathsf{PE.mpk})$.

- The i-th slot encryption algorithm on input $(\mathsf{msk}, \mathbf{x}_i, \mu_i)$ generates an i-th slot $\mathsf{mi\text{-}prFE}$ ciphertext $\mathsf{mi\text{-}prFE.ct}_i \leftarrow \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{mi\text{-}prFE.msk}, (\mathbf{x}_i, \mu_i))$. It outputs $\mathsf{ct}_i = \mathsf{mi\text{-}prFE.ct}_i$

- The key generator on input msk and a function f generates a single-input PE functional secret key $\mathsf{PE.sk}_f \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f)$. It also generates a $\mathsf{mi\text{-}prFE}$ key, $\mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}$, for function $\mathrm{F}[\mathsf{PE.mpk}]$ that, on input n attribute-message pairs $(\mathbf{x}_1, \mu_1), \ldots, (\mathbf{x}_n, \mu_n)$, generates a single-input PE ciphertext w.r.t. attribute $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and message $\mu = (\mu_1, \ldots, \mu_n)$. It outputs $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}})$.

- The decryption algorithm first runs the $\mathsf{mi\text{-}prFE}$ decryption using $\mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}$ and $\{\mathsf{ct}_i = \mathsf{mi\text{-}prFE.ct}_i\}_{i\in[n]}$ to compute the single-input PE ciphertext, $\mathsf{PE.ct}$, encoding message $\mu = (\mu_1, \ldots, \mu_n)$ w.r.t attribute $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Finally it performs PE decryption using $\mathsf{PE.sk}_f$ and $\mathsf{PE.ct}$.

Correctness and security follow readily from those of the underlying building blocks. Please see Section 6.4 for details. In summary, we obtain the following theorems.

Theorem 6.11 (miPE for poly arity)**.** *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE, non-uniform sub-exponential PRF, and non-uniform sub-exponential LWE, there exists a miPE scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions of bounded polynomial depth.*

Using $\mathsf{mi\text{-}prFE}$ scheme for constant arity allows to relax the assumption to normal evasive LWE.

Theorem 6.12 (miPE for constant arity)**.** *Assuming evasive LWE and LWE, there exists a $\mathsf{mi\text{-}prFE}$ scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth.*

Two Party ID based Non-Interactive Key Exchange. Next, we provide a construction of two party ID based non-interactive key exchange (ID-NIKE) scheme. The construction is the same as the ID-based NIKE system by Sakai, Ohgishi, and Kasahara [159] except

that the hash function is replaced with an obfuscation of a PRF, which can be supported by our prIO. In more detail, we show:

Theorem 6.13. *Let* $\kappa = \lambda^{n^2 \log \lambda}$. *Assuming non-uniform* κ*-evasive LWE (Assumption 6.14), non-uniform sub-exponential* PRF*, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure ID-NIKE scheme.*

This leads to the first construction of ID-NIKE without multilinear maps [88] or indistinguishability obfuscation in the standard model. Please see Section 6.8 for details.

Instantiating the Random Oracle. In an elegant work [119], Hohenberger, Sahai and Waters posed the following question: "Can we instantiate the random oracle with an actual family of hash functions for existing cryptographic schemes in the random oracle model, such as Full Domain Hash signatures?" They then demonstrated that the selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP) [37], the adaptive security of RSA FDH signatures [79], the selective security of BLS signatures, and the adaptive security of BLS signatures [45] can be proven in the standard model by carefully instantiating the underlying hash function by IO for each application.

We show in Section 6.7.1, that the random oracle in the FDH signature can be instantiated using prIO instead of full-fledged $i\mathcal{O}$. Similarly, we can instantiate the random oracle in selectively secure BLS signatures with prIO, following a strategy similar to that in [119]. At a high level, these proofs follow those in the random oracle model (ROM), where we use $i\mathcal{O}$ to obfuscate a derandomized version of the simulator for the hash function in ROM-based proofs. In these settings, the truth table of the simulated hash function is pseudorandom, allowing us to follow the same proof strategy using prIO.

For adaptively secure RSA FDH and BLS signatures, the situation is different. In these cases, Hohenberger et al. adopt an alternative proof strategy that deviates from the high level strategy of obfuscating the simulator for the proof in the ROM. This is due to the fact that the original proofs [45, 79] are incompatible with the conditions required for

using iO, where the truth table of the hash functions must remain unchanged across game hops. To be compatible with iO, they introduce a structure for the hash function, making its truth table no longer pseudorandom. This prevents us from replacing the hash function with prIO following their approach.

To instantiate the hash function with prIO, we revert to the original ROM-based proof strategy [45, 79]. Unlike the iO-based approach, prIO-based proof does not require the truth table of the hash function to remain unchanged across game hops; it only requires the truth table to be pseudorandom. This relaxed condition enables the use of the original ROM security proofs. Please see Section 6.7 for details.

Organization of the Chapter We provide the preliminaries used in this work in Section 6.2. In Section 6.3, we define the notion of multi-input FE for pseudorandom functionalities (mi-prFE) and construct a (bounded-depth) mi-prFE using a single-input FE scheme for pseudorandom functionalities (prFE) – a tool from our companion paper [12]. In Section 6.A, we recall the construction of prFE from [12] and prove that it achieves strengthened security notion of non-uniform κ-prCT security which is required for the mi-prFE compiler in Section 6.3. In Section 6.4, we give the construction for multi-input predicate encryption scheme for polynomial arity. In Section 6.5, we give the definitions for indistinguishability obfuscation for pseudorandom functionalities (prIO) for circuits and construct this using a mi-prFE scheme supporting polynomial arity. In Section 6.6, we define and construct indistinguishability obfuscation for pseudorandom functionalities with polynomial size domain (pPRIO) using a mi-prFE scheme supporting constant arity. In Section 6.7 and 6.8 we show how to instantiate the random oracle using a prIO scheme via various applications.

6.2 PRELIMINARIES

In this section we introduce some additional preliminaries for this chapter. We refer to Chapter 2 for preliminaries on lattices, Section 4.2.7 for properties of GSW Homomorphic encryption and evaluation and Section 4.2.8 for properties of BGG^+ homomorphic evaluation procedures.

6.2.1 Hardness Assumptions

The following variant of the evasive LWE assumption will be used in Section 6.3. This strengthens Assumption 2.6 in that it considers non-uniform samplers and replaces the negligible term in Equation (2.3) with negligible function in another parameter κ, which can be much larger than λ. The reason why we need this strengthened version of the assumption is that we need prFE to satisfy stronger security notion than prCT security that we call non-uniform κ-prCT security for the application to $\mathsf{mi\text{-}prFE}$. We refer to Remark 35 for the detailed discussion on why we need the stronger security definition for prFE for the application to $\mathsf{mi\text{-}prFE}$.

Assumption 6.14 (Non-Uniform κ-Evasive LWE)**.** Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 2.6 and $\mathsf{Samp} = \{\mathsf{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $\mathbf{S}, \mathbf{P}, \mathsf{aux}$ as in Assumption 2.6. For non-uniform adversaries $\mathcal{A}_0 = \{\mathcal{A}_{0,\lambda}\}_\lambda$ and $\mathcal{A}_1 = \{\mathcal{A}_{1,\lambda}\}_\lambda$, we define the advantage functions $\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda)$ and $\mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)$ as in Equation (2.1) and Equation (2.2), respectively. For a function $\kappa := \kappa(\lambda)$ of the security parameter λ, we say that the non-uniform κ-evasive LWE assumption with respect to the sampler class $\mathcal{SC}$ holds if for every non-uniform sampler $\mathsf{Samp} \in \mathcal{SC}$ and a non-uniform adversary Adv_1 such that $\mathrm{Size}(\mathsf{Samp}) \le \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathsf{Adv}_1) \le \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \le \kappa(\lambda)$, there exists another non-uniform adversary Adv_0 and a polynomial $Q(\cdot)$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \ge \mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \text{and} \quad \mathrm{Size}(\mathcal{A}_0) \le Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1). \tag{6.13}$$

Note that in the case λ' is superpolynomial in λ, $\mathsf{Samp}(1^\lambda)$ outputs $\mathbf{S}$, $\mathbf{P}$, and aux

whose sizes are polynomial in λ' and thus superpolynomial in λ. We require the above assumption with $\kappa = 2^{\mathrm{poly}(\lambda)}$ in our construction.

The following lemma is an adaptation of Theorem 2.7 for the stronger version of evasive LWE assumption defined in Assumption 6.14. The proof is almost the same as that for Theorem 2.7. We provide it here for completeness.

Lemma 6.15. *Let $n, m, t, m', q, \lambda \in \mathbb{N}$ be parameters defined as in Assumption 2.6 and $\mathrm{Samp} = \{\mathrm{Samp}_\lambda\}_\lambda$ be a non-uniform sampler that takes as input 1^λ and outputs $\mathbf{S}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2)$, and $\mathbf{P}$ as in Theorem 2.7. For a non-uniform adversaries $\mathcal{A}$, we define the advantage functions $\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda)$ and $\mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}}(\lambda)$ as in Equation (2.4) and Equation (2.5), respectively. Then, for a function $\kappa := \kappa(\lambda)$ of the security parameter λ, under the non-uniform κ-evasive LWE assumption (Assumption 6.14), if $\mathrm{Size}(\mathrm{Samp}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathsf{Adv}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda'(\lambda) \leq \kappa(\lambda)$, there exists another non-uniform adversary Adv_0 and a polynomial $Q(\cdot)$ such that*

$$\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}'}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \quad \textit{and} \quad \mathrm{Size}(\mathcal{A}_0) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1).$$

Proof. Let us consider an adversary $\mathcal{A}_1$ and a sampler Samp with size being polynomial in κ and λ' respectively and $\epsilon = \mathcal{A}^{\mathrm{POST}'}_{\mathcal{A}_1}$. Then, the same adversary is able to distinguish either (1) $(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ with advantage at least $\epsilon/2$ or (2) $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2)$ with advantage at least $\epsilon/2$. If the latter is the case, then we can obtain an adversary $\mathcal{A}_0$ that distinguishes $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ with advantage $\epsilon/2$. This can be seen by observing that $\mathcal{A}_1$ can be turned into an adversary that distinguishes $(\mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{c}, \mathsf{aux}_2)$ and then turned into an adversary that distinguishes $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2)$ by sampling $(\mathbf{B}, \mathbf{C}_0, \mathbf{K})$ by itself, where we sample $\mathbf{B}$ with the corresponding trapdoor and then sample $\mathbf{K}$ using it. We therefore assume that the former is the case. Then, by invoking the non-uniform κ-evasive LWE with respect to the sampler Samp, we obtain another adversary $\mathcal{A}_0$ whose size is bounded by

$Q(\lambda') \cdot \mathrm{Size}(\mathcal{A}_1)$ and distinguishing advantage against $(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ and $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}_1, \mathsf{aux}_2)$ is at least $\epsilon/2Q(\lambda')$. Then, $\mathcal{A}_0$ is able to distinguish either (1) $(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ with advantage at least $\epsilon/4Q(\lambda')$ or (2) $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ from $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}_1, \mathsf{aux}_2)$ with advantage at least $\epsilon/4Q(\lambda')$. If the former is the case, we are done. If the latter is the case, we are still able to convert it into a distinguisher against $(\mathbf{B}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2)$ and $(\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$ by the similar argument to the above. ■

Assumption 6.16 (Decisional Bilinear Diffie-Hellman)**.** Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, where $e(g^a, g^b) = e(g, g)^{ab}$ and $a, b \in \mathbb{Z}_p$, for a cyclic group $\mathbb{G}$ of order p with a generator g. The Decisional Bilinear Diffie-Hellman assumption (DBDH) states that for any PPT adversary

$$\left(g^\alpha, g^\beta, g^\gamma, e(g,g)^{\alpha\beta\gamma} \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p\right) \approx_c \left(g^\alpha, g^\beta, g^\gamma, T \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p, T \leftarrow \mathbb{G}_T\right)$$

6.2.2 Puncturable Pseudorandom Functions

Syntax. A puncturable pseudorandom function (PPRF) $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key space $\mathcal{K}$, input space $\mathcal{X}$ and output space $\mathcal{Y}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to K$. The setup algorithm takes as input the security parameter λ and outputs a key $K \in \mathcal{K}$.

$\mathsf{puncture}(K, x) \to K_x$. The puncture algorithm takes as input a PRF key $K \in \mathcal{K}$ and an input $x \in \mathcal{X}$, and outputs a punctured key key K_x.

$\mathsf{Eval}(K_x, x') \to y$. The evaluation algorithm takes as input a punctured key K_x an input $x' \in \mathcal{X}$, such that $x \neq x'$ and outputs $y \in \mathcal{Y}$. It outputs $\perp$ if $x = x'$.

Definition 6.1. (Correctness) A PPRF scheme is said to be correct if for any $K \in \mathcal{K}$, $x, x' \in \mathcal{X}$ such that $x \neq x'$, we have

$$\mathsf{Pr}[\mathsf{Eval}(K_x, x') = F(K, x') \mid K_x \leftarrow \mathsf{puncture}(K, x)] = 1.$$

Definition 6.2. (Security) A PPRF scheme is said to be (selectively) secure if the advantage of a PPT adversary $\mathcal{A}$ in the following experiment is negligible.

1. $\mathcal{A}$ on input 1^λ outputs the challenge input $x^\star$.
2. The challenger samples a random key $K \leftarrow \mathcal{K}$ and a bit $\beta \leftarrow \{0,1\}$. Then, it computes $y = F(K,x)$ if $\beta = 0$, else it sample $y \leftarrow \mathcal{Y}$ uniformly at random. It also computes $K_{x^\star} \leftarrow \mathsf{puncture}(K, x^\star)$ and sends $K_{x^\star}, y$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a guess bit β'.

$\mathcal{A}$ wins if $\beta = \beta'$.

We know that PPRF with the security defined above from one-way functions [100, 47, 51, 134].

6.2.3 Pseudorandom Functional Encryption

In this section we define an additional security notion for functional encryption for pseudorandom functionalities. We refer to Section 5.3 for definitions of a prFE scheme.

Definition 6.3 (Non-uniform κ-prCT Security)**.** For a prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, and function $\kappa \stackrel{\text{def}}{=} \kappa(\lambda)$ of λ, let $\mathrm{Samp} = \{\mathrm{Samp}_\lambda\}_\lambda$ be a non-uniform polynomial-time algorithm that on input 1^λ, outputs

$$(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*)$$

where Q_{key} is the number of key queries, Q_{msg} is the number of message queries, and $f_i \in \mathcal{F}_{\mathsf{prm}}$ $x_j \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [Q_{\mathsf{key}}], j \in [Q_{\mathsf{msg}}]$.

For non-uniform adversaries $\mathcal{A}_0 := \{\mathcal{A}_{0,\lambda}\}_\lambda$ and $\mathcal{A}_1 := \{\mathcal{A}_{1,\lambda}\}_\lambda$, we define the following advantage functions:

$$\mathsf{Adv}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \stackrel{\text{def}}{=} \mathsf{Pr}[\mathcal{A}_0\left(\ \mathsf{aux},\ \{f_i,\ f_i(x_j)\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}\right) = 1]$$
$$- \mathsf{Pr}[\mathcal{A}_0(\ \mathsf{aux},\ \{f_i,\ \Delta_{i,j} \leftarrow \mathcal{Y}_{\mathsf{prm}}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1]$$

$$\mathsf{Adv}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda) \stackrel{\text{def}}{=} \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_j),\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1]$$
$$- \mathsf{Pr}[\mathcal{A}_1(\mathsf{mpk},\ \mathsf{aux},\ \{f_i,\ \delta_j \leftarrow \mathcal{CT},\ \mathsf{sk}_{f_i}\}_{i\in[Q_{\mathsf{key}}],j\in[Q_{\mathsf{msg}}]}) = 1]$$

where $(f_1, \ldots, f_{Q_{\mathsf{key}}}, x_1, \ldots, x_{Q_{\mathsf{msg}}}, \mathsf{aux} \in \{0,1\}^*) \leftarrow \mathsf{Samp}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$ and $\mathcal{CT}$ is the ciphertext space. We say that a prFE scheme for function family $\mathcal{F}_{\mathsf{prm}}$ is secure in the non-uniform κ setting with respect to the sampler class $\mathcal{SC}$ if for every sampler $\mathsf{Samp} \in \mathcal{SC}$ and an adversary $\mathcal{A}_1$ such that $\mathrm{Size}(\mathsf{Samp}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda' \leq \kappa$, there exists another adversary $\mathcal{A}_0$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathcal{A}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathcal{A}_1}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \tag{6.14}$$

and $\mathrm{Size}(\mathcal{A}_0) \leq \mathrm{Size}(\mathcal{A}_1) \cdot Q(\lambda')$ for some polynomial $Q(\cdot)$.

In Section 6.A.1, we prove the following theorem.

Theorem 6.17. *Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 2.5), there exists a* prFE *scheme for function class $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$ satisfying κ-*prCT *security as per Definition 6.3, with respect to a specific sampler class, with efficiency*

$$|\mathsf{mpk}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ is the depth bound on the functions supported by the scheme.

Remark 31 (Comparison between Definition 6.3 and Definition 5.13)**.** We remark that Definition 6.3 strengthens Definition 5.13 in two aspects. First of all, it considers non-uniform adversaries instead of uniform adversaries. Secondly, it is parameterized by κ and the additive term $\mathsf{negl}(\lambda)$ is replaced by $\mathsf{negl}(\kappa)$ in Equation (6.14). By taking κ asymptotically larger than λ (e.g., $\kappa := \lambda^\lambda$), we can make the additive term $\mathsf{negl}(\kappa)$ much smaller than $\mathsf{negl}(\lambda)$. We note that these changes are introduced to prove the security of our mi-prFE in Section 6.3. We refer to Remark 35 for the discussion on why these changes are necessary for the security proof there.

Theorem 6.18 ([12])**.** *Assuming LWE and evasive LWE assumptions, there exists a secure (Definition 5.13) prFE scheme, with respect to a specific sampler class, for*

function class $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$ *satisfying*

$$|\mathsf{mpk}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ *is the depth bound on the functions supported by the scheme.*

6.2.4 Predicate Encryption

Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : \mathcal{X}_{\mathsf{prm}} \to \{0,1\}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$.

Syntax. A PE scheme PE for function family $\mathcal{F}_{\mathsf{prm}}$ consists of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and a parameter prm, and outputs a master public key mpk and master secret key msk[7].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$.The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$. The encryption algorithm takes as input a master secret key msk, an attribute $x \in \mathcal{X}_{\mathsf{prm}}$, and message $\mu \in \{0,1\}$, and outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}) \to \{0,1\} \cup \perp$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and ciphertext ct, and outputs a string $\mu' \in \{0,1\} \cup \perp$.

Definition 6.4 (Correctness.)**.** For every $\lambda \in \mathbb{N}, \mu \in \{0,1\}$, $x \in \mathcal{X}_{\mathsf{prm}}$, $f \in \mathcal{F}_{\mathsf{prm}}$, if

[7]We assume w.l.o.g that msk includes mpk.

$f(x) = 1$, then

$$\mathsf{Pr}\left[\mathsf{Dec}\left(\mathsf{mpk}, \mathsf{KeyGen}(\mathsf{msk}, f), f, \mathsf{Enc}(\mathsf{mpk}, x, \mu)\right) = \mu\right] = 1 - \mathsf{negl}(\lambda)$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm})$.

Definition 6.5 (Selective INDr Security)**.** A PE scheme is said to satisfy selective INDr security if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have

$$\left|\mathsf{Pr}\left[\beta' = \beta : \begin{array}{l} (\mathbf{x}, \mathsf{prm}) \leftarrow \mathcal{A}(1^\lambda); \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{prm}); \\ (\mu, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}); \\ \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu), \mathsf{ct}_1 \leftarrow \mathcal{CT}; \\ \beta \leftarrow \{0,1\}, \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{st}, \mathsf{ct}_\beta) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda), \quad (6.15)$$

where $\mathcal{CT}$ is the ciphertext space of the scheme and the adversary $\mathcal{A}$ is admissible in the sense that for all key query f made by $\mathcal{A}$, it holds that $f(\mathbf{x}) = 0$.

Predicate encryption schemes for (bounded depth) circuits satisfying the above security notion are known from LWE [108, 111, 172].

6.2.5 Multi-Input Predicate Encryption

In this section we define multi-input Predicate Encryption (mi-PE), adapting the syntax from [25]. Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \rightarrow \{0,1\}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, where each $\mathcal{F}_{\mathsf{prm}}$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$.

Syntax. A mi-PE scheme miPE_n for n-ary function family $\mathcal{F}_{\mathsf{prm}}$ consists of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \rightarrow (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, the function arity n and a parameter prm and outputs a master public

key mpk and master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$.The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}_i(\mathsf{msk}, x_i, \mu_i) \to \mathsf{ct}_i$ for $i \in [n]$. The encryption algorithm for the i^{th} slot takes as input a master secret key msk, an attribute $x_i \in \mathcal{X}_{\mathsf{prm}}$, and message $\mu_i \in \{0,1\}$, and outputs a ciphertext ct_i.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_n) \to \{0,1\}^n \cup \perp$. The decryption algorithm takes as input the master public key mpk, secret key sk_f, function f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and outputs a string $\mu' \in \{0,1\}^n \cup \perp$.

Next, we define correctness and security.

Correctness: For every $\lambda \in \mathbb{N}, \mu_1, \ldots, \mu_n \in \{0,1\}, x_1, \ldots, x_n \in \mathcal{X}_{\mathsf{prm}}$, $f \in \mathcal{F}_{\mathsf{prm}}$, it holds that if $f(x_1, \ldots, x_n) = 1$, then

$$\mathsf{Pr}\left[\mathsf{Dec}\left(\begin{matrix}\mathsf{mpk}, \mathsf{KeyGen}(\mathsf{msk}, f), f, \\ \mathsf{Enc}_1(\mathsf{msk}, x_1, \mu_1), \ldots, \mathsf{Enc}_n(\mathsf{msk}, x_n, \mu_n)\end{matrix}\right) = (\mu_1, \ldots, \mu_n)\right] = 1 - \mathsf{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$ and over the internal randomness of KeyGen and $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_n$.

Definition 6.6 (Sim-Security.)**.** For a miPE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \{0,1\}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, a stateful adversary $\mathcal{A}$, and a simulator algorithm $\{\mathsf{Sim}_i\}_{i\in[n]}$, we define the Sim-security game, $\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}$, as follows.

1. **Query phase:** On input $1^\lambda, 1^n, \mathsf{prm}$, $\mathcal{A}$ outputs the following in an arbitrary order.

   a) **Key Queries:** $\mathcal{A}$ issues polynomial number of key queries, say $q_0 = q_0(\lambda)$. For each key query $k \in [q_0]$, $\mathcal{A}$ chooses a function $f_k \in \mathcal{F}_{\mathsf{prm}}$.

b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $q_i = q_i(\lambda)$ for the i^{th} slot. We use $(x_i^{j_i}, \mu_i^{j_i})$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.

2. **Setup phase:** On input $1^\lambda, 1^n, \mathsf{prm}, \{f_k\}_{k\in[q_0]}$, the challenger samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, a bit $\beta \leftarrow \{0, 1\}$ and does the following.

   a) It computes $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$.

   b) If $\beta = 0$, it computes $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}_i(\mathsf{msk}, x_i^{j_i}, \mu_i^{j_i})$, else if $\beta = 1$, it computes $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Sim}_i(\mathsf{msk})$ for $i \in [n]$, $j_i \in [q_i]$.

   It returns $(\mathsf{mpk}, \{\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \{\mathsf{ct}_i^{j_i}\}_{i\in[n],j_i\in[q_i]})$ to $\mathcal{A}$.

3. **Output phase:** $\mathcal{A}$ outputs a guess bit β' as the output of the experiment.

For the adversary to be *admissible*, we require that it holds that $f_k(x_1^{j_1}, \ldots, x_n^{j_n}) = 0$ for every $i \in [n]$, $j_i \in [q_i]$, and $k \in [q_0]$. We define the advantage $\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{miPE},\mathcal{A}}$ of $\mathcal{A}$ in the security game as

$$\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) := \left|\mathsf{Pr}[\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = 1|\beta = 0] - \mathsf{Pr}[\mathsf{Exp}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = 1|\beta = 1]\right|.$$

The miPE scheme is said to satisfy Sim-security if for any stateful PPT adversary $\mathcal{A}$, there exists a PPT simulator algorithm $\{\mathsf{Sim}_i\}_{i\in[n]}$ such that $\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{miPE},\mathcal{A}}(1^\lambda) = \mathsf{negl}(\lambda)$.

6.2.6 ID-Based Non-Interactive Key Exchange

In this section, we give the definitions for an identity-based non-interactive key exchange scheme, for two parties, adapted from [88].

Syntax. An identity-based non-interactive key exchange (ID-NIKE) scheme for identity space $\mathcal{ID}$ has the following syntax.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$ The setup algorithm takes as input the security parameter 1^λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}) \to \mathsf{usk}_{\mathsf{id}}$**.** The key extraction algorithm takes as input the master public key mpk, the master secret key msk and an identity $\mathsf{id} \in \mathcal{ID}$. It outputs a

user secret key $\mathsf{usk}_{\mathsf{id}}$ for id.

$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2) \to K$**.** The share algorithm takes as input the master public key mpk, a user secret key $\mathsf{usk}_{\mathsf{id}_1}$ for an identity $\mathsf{id}_1 \in \mathcal{ID}$ and an identity $\mathsf{id}_2 \in \mathcal{ID}$. It outputs a shared key K.

Definition 6.7 (Correctness)**.** An ID-NIKE scheme for an identity space $\mathcal{ID}$ is correct if for any $\lambda \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, any $\mathsf{id}_1, \mathsf{id}_2 \in \mathcal{ID}$ we have

$$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2) = \mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_2}, \mathsf{id}_1)$$

where $\mathsf{usk}_{\mathsf{id}_1} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_1)$ and $\mathsf{usk}_{\mathsf{id}_2} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_2)$.

Definition 6.8 (Security)**.** We say that an ID-NIKE scheme for an identity space $\mathcal{ID}$ is secure if the advantage function

$$\mathrm{Adv}_{\text{ID-NIKE},\mathcal{A}}(\lambda) = \mathsf{Pr}\left[\mathrm{Exp}_{\text{ID-NIKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}$$

is negligible for all PPT adversaries $\mathcal{A}$. Here, experiment $\mathrm{Exp}_{\text{ID-NIKE},\mathcal{A}}$ is defined as follows:

1. **Setup Phase.** The experiment samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and gives $\mathcal{A}$ the mpk.

2. **Query Phase.** The adversary can make the following queries in an arbitrary order.

   - Extraction Query: $\mathcal{A}$ sends an extraction query for an identity $\mathsf{id} \in \mathcal{ID}$. The experiment returns $\mathsf{usk}_{\mathsf{id}}$ to the adversary, where $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$.

   - Challenge Query : $\mathcal{A}$ sends a pair $(\mathsf{id}_1^\star, \mathsf{id}_2^\star) \in \mathcal{ID} \times \mathcal{ID}$ as the challenge query. The experiment samples a bit $\beta \leftarrow \{0, 1\}$ and returns $K^\star \leftarrow \mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1^\star}, \mathsf{id}_2^\star)$, where $\mathsf{usk}_{\mathsf{id}_1^\star} \leftarrow \mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id}_1^\star)$, if $\beta = 0$ and returns $K^\star \leftarrow \mathbb{G}_T$ if $\beta = 1$.

3. **Output Phase.** $\mathcal{A}$ outputs a guess bit β', and the experiment outputs 1 if $\beta = \beta'$.

We only quantify over $\mathcal{A}$ that guarantees that does not make extraction queries for $\mathsf{id}_1^\star$ and $\mathsf{id}_2^\star$.

6.3 MULTI-INPUT FE FOR PSEUDORANDOM FUNCTIONALITIES

In this section, we construct our main tool – multi-input functional encryption for pseudorandom functionalities.

6.3.1 Definition

In this section we give the definitions for multi-input functional encryption for pseudorandom functionalities (mi-prFE).

Consider a function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, for a parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, where each $\mathcal{F}_{\mathsf{prm}}$ is a finite collection of n-ary functions. Each function $f \in \mathcal{F}_{\mathsf{prm}}$ takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \mathcal{X}_{\mathsf{prm}}$ and outputs $f(x_1, \ldots, x_n) \in \mathcal{Y}_{\mathsf{prm}}$.

Syntax. A miprfe scheme $\mathsf{mi\text{-}prFE}_n$ for n-ary function family $\mathcal{F}_{\mathsf{prm}}$ consists of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ defined as follows.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, the function arity n and a parameter prm and outputs a master public key mpk and master secret key msk[8].

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_{\mathsf{prm}}$ and it outputs a functional secret key sk_f.

$\mathsf{Enc}_i(\mathsf{msk}, x) \to \mathsf{ct}$. The encryption algorithm for the i-th slot takes as input the master secret key msk and an input $x \in \mathcal{X}_{\mathsf{prm}}$ and outputs a ciphertext $\mathsf{ct}_i \in \mathcal{CT}$, where $\mathcal{CT}$ is the ciphertext space.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y$. The decryption algorithm takes as input the

[8] We assume w.l.o.g that msk includes mpk.

master public key mpk, secret key sk_f, function f and n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$, and outputs $y \in \mathcal{Y}_{\mathsf{prm}}$.

Definition 6.9 (Correctness)**.** A mi-prFE scheme is said to be correct if for every prm, n-ary function $f \in \mathcal{F}_{\mathsf{prm}}$ and input tuple $(x_1, \ldots, x_n) \in \mathcal{X}^n_{\mathsf{prm}}$ we have

$$\Pr\left[\begin{array}{c}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \ , \ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), \\ \mathsf{Dec}\big(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{Enc}_1(\mathsf{msk}, x_1), \ldots \mathsf{Enc}_n(\mathsf{msk}, x_n)\big) = f(x_1, \ldots, x_n)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda). \tag{6.16}$$

Definition 6.10 (κ-Security)**.** Let $\kappa = \kappa(\lambda)$ be a function in λ. For a mi-prFE scheme for function family $\{\mathcal{F}_{\mathsf{prm}} = \{f : (\mathcal{X}_{\mathsf{prm}})^n \to \mathcal{Y}_{\mathsf{prm}}\}\}_{\mathsf{prm}}$, parameter $\mathsf{prm} = \mathsf{prm}(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(\{f_k\}_{k\in[q_0]}, \{x_1^{j_1}\}_{j_1\in[q_1]}, \ldots, \{x_n^{j_n}\}_{j_n\in[q_n]}, \mathsf{aux} \in \{0,1\}^*\right)$$

where q_0 is the number of key queries, q_i is the number of encryption queries for the i-th slot, $f_1, \ldots, f_{q_0} \in \mathcal{F}_{\mathsf{prm}}$ and $x_i^{j_i} \in \mathcal{X}_{\mathsf{prm}}$ for all $i \in [n]$, $j_i \in [q_i]$. We say that the mi-prFE scheme satisfies κ-security with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ there exists a PPT simulator algorithm $\{\mathsf{Sim}_i\}_{i\in[n]}$ such that

$$\begin{aligned}\text{If } &\left(1^\kappa, \left\{f_k, f_k(x_1^{j_1}, \ldots, x_n^{j_n})\right\}_{k\in[q_0], j_1\in[q_1]\ldots, j_n\in[q_n]}, \mathsf{aux}\right) \\ &\approx_c \left(1^\kappa, \left\{f_k, \Delta_{k,j_1,\ldots,j_n}\right\}_{k\in[q_0], j_1\in[q_1],\ldots,j_n\in[q_n]}, \mathsf{aux}\right)\end{aligned} \tag{6.17}$$

$$\begin{aligned}\text{then } &\left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{ct}_i^{j_i}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right) \\ &\approx_c \left(\mathsf{mpk}, \left\{f_k, \mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\delta_i^{j_i}\right\}_{i\in[n], j_i\in[q_i]}, \mathsf{aux}\right),\end{aligned} \tag{6.18}$$

where $\kappa \geq \lambda^n$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$, $\mathsf{sk}_{f_k} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_k)$, $\mathsf{ct}_i^{j_i} \leftarrow \mathsf{Enc}(\mathsf{msk}, x_i^{j_i})$, $\delta_i^{j_i} \leftarrow \mathsf{Sim}_i(\mathsf{msk})$, and $\Delta_{k,j_1,\ldots,j_n} \leftarrow \mathcal{Y}_{\mathsf{prm}}$ for $i \in [n]$, $j_i \in [q_i]$, and $k \in [q_0]$.

Remark 32. Note that 1^κ in Equation (6.17) is introduced for the purpose of padding, allowing the distinguisher for the distributions to run in time polynomial in κ and requiring

the distinguishing advantage to be negligible in κ.[9] The reason why we require $\kappa \geq \lambda^n$ is that the input length to the distinguisher is polynomial in λ^n anyway and in order for the padding to make sense, κ should satisfy this condition. If we need κ to be larger, this doubly strengthens the requirement for the precondition, as it means we want the distributions in Equation (6.17) to be indistinguishable against an adversary with a longer running time and smaller advantage. Ideally, we want κ to be as small as λ^n to make the requirement weaker. However, the security proof for our construction in Section 6.3.2 for general n requires large κ as an artifact of the proof technique. In the special case of n being constant, we can achieve $\kappa = \lambda^n$.

The following variant of security will be necessary in Section 6.6.

Definition 6.11 (Pseudorandomness of the Last Slot Ciphertext)**.** We say that a $\mathsf{mi\text{-}prFE}$ scheme satisfies κ-pseudorandomness of the last slot ciphertext property if it satisfies κ-security as per defined in Definition 6.10 where the simulator Sim_n corresponding to the last slot ciphertext outputs a random string of the same length as $\mathsf{ct}_n^{j_n}$.

Remark 33. It is shown in [19, 65] that there is no $\mathsf{mi\text{-}prFE}$ that satisfies the above style security for all general samplers. Therefore, when we use the security of $\mathsf{mi\text{-}prFE}$, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications. For simplicity, we sometimes will treat as if there was $\mathsf{mi\text{-}prFE}$ that is secure for all the samplers.

6.3.2 Construction for n-input prFE

In this section we provide our construction of a multi-input functional encryption scheme for pseudorandom functionalities for function family $\mathcal{F}_{nL(\lambda),d(\lambda)} = \{f : \{\{0,1\}^L\}^n \to \{0,1\}\}$, where the depth of a function $f \in \mathcal{F}$ is at most $d(\lambda) = \mathrm{poly}(\lambda)$. Each function $f \in \mathcal{F}$ takes as input strings $x_1, \ldots, x_n \in \{0,1\}^L$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$. We consider the case of arity n being constant and the general case of n being arbitrary polynomial in λ. While we provide separate security proofs for these cases, we have

[9] This is due to our convention, where the running time of the distinguisher should be polynomial in its input length and the distinguishing advantage should be negligible in its input length. Please refer to Section 6.2 for the details.

unified description of the construction. The reason why we consider the proofs separately is that we can base the security of the scheme on a weaker assumption when n is constant than the general case.

Building Blocks. Our construction uses the following building blocks.

1. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$. When we set up the scheme SKE, we run it on a scaled version of the security parameter Λ, instead of the usual security parameter λ. We will explain how to set Λ in Remark 34. We denote the ciphertext space of the scheme by $\mathcal{CT}_{\mathsf{SKE},\Lambda}$ and the key space of the scheme by $\mathcal{K}_{\mathsf{SKE},\Lambda}$. In the following, we drop Λ and denote them as $\mathcal{CT}_{\mathsf{SKE}}$ and $\mathcal{K}_{\mathsf{SKE}}$ respectively.

2. n single-input FE scheme for pseudorandom functionality $\mathsf{prFE}_1, \ldots, \mathsf{prFE}_n$. For $i \in [n]$, $\mathsf{prFE}_i = (\mathsf{prFE}_i.\mathsf{Setup}, \mathsf{prFE}_i.\mathsf{KeyGen}, \mathsf{prFE}_i.\mathsf{Enc}, \mathsf{prFE}_i.\mathsf{Dec})$ for circuit class $\mathcal{C}_{\mathsf{inp}_i(\lambda),\mathsf{dep}_i(\lambda),\mathsf{out}_i(\lambda)}$ consisting of circuits with input length $\mathsf{inp}_i(\lambda)$, maximum depth $\mathsf{dep}_i(\lambda)$ and output length $\mathsf{out}_i(\lambda)$. We denote the ciphertext space of the prFE_i scheme by $\mathcal{CT}_{\mathsf{prFE}_i}$.
   For our construction, we set the following parameters

   - $\mathsf{inp}_1 = n \cdot L$, $\mathsf{dep}_1 = d$, and $\mathsf{out}_1 = 1$.

   - $\mathsf{inp}_i = |\mathsf{SKE.key}| + (n-i)L + n\Lambda$, $\mathsf{dep}_i = \mathrm{poly}(d, \lambda)$, and $\mathsf{out}_i = |\mathsf{prFE}_{i-1}.\mathsf{ct}|$ for $i \in [2, n]$, where $\mathsf{SKE.key} \in \mathcal{CT}_{\mathsf{SKE}}$ and $\mathsf{prFE}_{i-1}.\mathsf{ct} \in \mathcal{CT}_{\mathsf{prFE}_{i-1}}$.

3. We also use $n-1$ pseudorandom functions $\mathsf{PRF}_1, \ldots, \mathsf{PRF}_{n-1}$. Similarly to the case of SKE, we use Λ to setup these instances of PRF. We specify the domain and codomain of the functions as $\mathsf{PRF}_i : \{0,1\}^\Lambda \times \{\{0,1\}^\Lambda\}^{n-i} \to \{0,1\}^{\mathsf{len}_i}$ where len_i is the length of randomness used in $\mathsf{prFE}_i.\mathsf{Enc}$ for $i \in [n-1]$.

We describe our construction of $\mathsf{mi\text{-}prFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ in the following.

$\mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Generate $(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{msk}) \leftarrow \mathsf{prFE}_i.\mathsf{Setup}(1^\lambda, 1^{\mathsf{prm}_i})$ for all $i \in [n]$,

- Generate $\mathsf{SKE.sk} \leftarrow \mathsf{SKE.Setup}(1^\Lambda)$.

- Output $\mathsf{mpk} := (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$ and $\mathsf{msk} := (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. The key generation algorithm does the following.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Compute $\mathsf{prFE}_1.\mathsf{sk}_f \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f)$.
- Output $\mathsf{sk}_f \coloneqq \mathsf{prFE}_1.\mathsf{sk}_f$.

$\mathsf{Enc}_i(\mathsf{msk}, \mathbf{x}_i) \to \mathsf{ct}_i$. For $i \in [n-1]$, the Enc_i algorithm outputs a function secret key corresponding to prFE_{i+1}-th instance in the following way.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Sample $\mathbf{r}_i \leftarrow \{0,1\}^\Lambda$.
- Compute $\mathsf{SKE.ct}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.sk}, \mathbf{x}_i)$.
- Define $\mathrm{F}_i \coloneqq \mathrm{F}_i[\mathsf{SKE.ct}_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}]$ as in Figure 6.1.[10]
- Compute $\mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i)$.
- Output $\mathsf{ct}_i \coloneqq \mathsf{prFE}_{i+1}.\mathsf{sk}$.

$\mathsf{Enc}_n(\mathsf{msk}, \mathbf{x}_n) \to \mathsf{ct}_n$. The Enc_n algorithm does the following.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- For $i \in [n-1]$, sample $K_i \leftarrow \{0,1\}^\Lambda$.
- Compute
  $\mathsf{prFE}_n.\mathsf{ct} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE.sk}, \mathbf{x}_n, K_1, \ldots, K_{n-1}))$.
- Output $\mathsf{ct}_n \coloneqq \mathsf{prFE}_n.\mathsf{ct}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y \in \{0,1\}$. The decryption algorithm does the following.

- Parse $\mathsf{mpk} = (\{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$, $\mathsf{sk}_f = \mathsf{prFE}_1.\mathsf{sk}_f$, $\mathsf{ct}_i = \mathsf{prFE}_{i+1}.\mathsf{sk}$ for $i \in [n-1]$, and $\mathsf{ct}_n = \mathsf{prFE}_n.\mathsf{ct}$.

[10] The hardwired values are not hidden, even if we don't output them explicitly.

Function $\mathrm{F}_i[\mathsf{SKE.ct}_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}]$

Hardwired constants: A SKE ciphertext $\mathsf{SKE.ct}_i$, $\mathbf{r}_i \in \{0,1\}^\Lambda$, and a prFE master public key $\mathsf{prFE}_i.\mathsf{mpk}$.
On input $(\mathsf{SKE.sk}, (\mathbf{x}_{i+1}, \mathbf{r}_{i+1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_i)$, proceed as follows:

1. Compute $\mathbf{x}_i \coloneqq \mathsf{SKE.Dec}(\mathsf{SKE.sk}, \mathsf{SKE.ct}_i)$.
2. Compute $\mathsf{prFE}_i.\mathsf{ct}$ as
   - $\mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1})))$ if $i = 1$.
   - $\mathsf{prFE}_i.\mathsf{Enc}\left(\begin{matrix}\mathsf{prFE}_i.\mathsf{mpk}, (\mathsf{SKE.sk}, (\mathbf{x}_i, \mathbf{r}_i), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-1}); \\ \mathsf{PRF}_i(K_i, (\mathbf{r}_i, \ldots, \mathbf{r}_{n-1}))\end{matrix}\right)$, if $i \neq 1$.
3. Output $\mathsf{prFE}_i.\mathsf{ct}$.

Figure 6.1: Function F_i

- For $i = n, \ldots, 2$ and do the following.
  1. Compute $\mathsf{prFE}_{i-1}.\mathsf{ct} \coloneqq \mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct})$.
  2. If $i = 2$ output $\mathsf{prFE}_1.\mathsf{ct}$, else set $i \coloneqq i - 1$ and go to Step 1.
- Output $y \coloneqq \mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk}, \mathsf{prFE}_1.\mathsf{sk}_f, f, \mathsf{prFE}_1.\mathsf{ct})$.

Remark 34. We consider two cases of parameter settings for the construction. One is the case of n being constant. In this case, we simply set $\Lambda = \lambda$. In the general case of $n = \mathrm{poly}(\lambda)$, we do something more complex. In this case, we assume that PRF and SKE have subexponential security. This means that there exists $0 < \delta < 1$ such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$ against SKE and PRF for all sufficiently large λ. In the security proof, we require PRF and SKE to be secure even against an adversary that takes 1^κ as an input and thus runs in polynomial time in κ. To satisfy this requirement, we run SKE and PRF with respect to a larger security parameter Λ that satisfies $2^{\Lambda^\delta} \geq \kappa^{\omega(1)}$. An example choice would be to take $\Lambda \coloneqq (n^2\lambda)^{1/\delta}$.

Efficiency. The scheme has the following parameters : $|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$, $|\mathsf{ct}_1| = nL\mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}_i| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$ for $i \in [2, n]$.

Correctness. We prove the correctness of our scheme via the following theorem.

Theorem 6.19. *Suppose* prFE_i *for* $i \in [n]$ *and* SKE *are correct, then the above construction of* mi-prFE *satisfies correctness as defined in Definition 6.9.*

Proof. To prove the theorem, we first prove the following statement.

Claim 6.20. For $i = n, \ldots, 2$, we have

$$\Pr[\mathsf{prFE}_i.\mathsf{Dec}(\mathsf{prFE}_i.\mathsf{mpk}, \mathsf{prFE}_i.\mathsf{sk}, \mathrm{F}_{i-1}, \mathsf{prFE}_i.\mathsf{ct}) = \mathsf{prFE}_{i-1}.\mathsf{ct}] = 1 \qquad (6.19)$$

where

$$\mathsf{prFE}_{i-1}.\mathsf{ct} = \begin{cases} \mathsf{prFE}_{i-1}.\mathsf{Enc}\left(\mathsf{prFE}_{i-1}.\mathsf{mpk}, (\mathsf{SKE}.\mathsf{sk}, (\mathbf{x}_{i-1}, \mathbf{r}_{i-1}), \ldots, (\mathbf{x}_{n-1}, \mathbf{r}_{n-1}), \mathbf{x}_n, K_1, \ldots K_{i-2}) \right. \\ \qquad\qquad \left. ; \mathsf{PRF}_{i-1}(K_{i-1}, (\mathbf{r}_{i-1}, \ldots, \mathbf{r}_{n-1}))\right) & \text{if } i \neq 2 \\ \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n); \mathsf{PRF}_1(K_1, (\mathbf{r}_1, \ldots, \mathbf{r}_{n-1}))) & \text{if } i = 2. \end{cases}$$

Proof. We prove this by induction.

Base Case: For $i = n$, we show that

$$\Pr[\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct}) = \mathsf{prFE}_{n-1}.\mathsf{ct}] = 1.$$

From the correctness of prFE_n scheme, we have with probability 1

$$\begin{aligned} &\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct}) \\ &\qquad = \mathrm{F}_{n-1}[\mathsf{SKE}.\mathsf{ct}_{n-1}, \mathbf{r}_{n-1}, \mathsf{prFE}_{n-1}.\mathsf{mpk}](\mathsf{SKE}.\mathsf{sk}, \mathbf{x}_n, K_1, \ldots K_{n-1}). \end{aligned}$$

Next, by the definition of F_{n-1} and the correctness of the SKE scheme, we have

$$\mathsf{prFE}_n.\mathsf{Dec}(\mathsf{prFE}_n.\mathsf{mpk}, \mathsf{prFE}_n.\mathsf{sk}, \mathrm{F}_{n-1}, \mathsf{prFE}_n.\mathsf{ct})$$

$$=\mathsf{prFE}_{n-1}.\mathsf{Enc}\left(\begin{array}{r}\mathsf{prFE}_{n-1}.\mathsf{mpk},(\mathsf{SKE.sk},(\mathbf{x}_{n-1},\mathbf{r}_{n-1}),\mathbf{x}_n,K_1,\ldots K_{n-2});\\ \mathsf{PRF}_{n-1}(K_{n-1},\mathbf{r}_{n-1})\end{array}\right) \tag{6.20}$$

which proves the base case.

Inductive Step: For the inductive step, suppose Equation (6.19) holds for some $i \in [3,n]$ then we prove the same statement for $i-1$. Consider

$$\begin{aligned}
&\mathsf{prFE}_{i-1}.\mathsf{Dec}(\mathsf{prFE}_{i-1}.\mathsf{mpk},\mathsf{prFE}_{i-1}.\mathsf{sk},\mathrm{F}_{i-2},\mathsf{prFE}_{i-1}.\mathsf{ct})\\
&=\mathrm{F}_{i-2}[\mathsf{SKE.ct}_{i-2},\mathbf{r}_{i-2},\mathsf{prFE}_{i-2}.\mathsf{mpk}](\mathsf{SKE.sk},(\mathbf{x}_{i-1},\mathbf{r}_{i-1}),\ldots,(\mathbf{x}_{n-1},\mathbf{r}_{n-1}),\mathbf{x}_n,K_1,\ldots K_{i-2})\\
&=\begin{cases}\mathsf{prFE}_{i-2}.\mathsf{Enc}\left(\begin{array}{r}\mathsf{prFE}_{i-2}.\mathsf{mpk},(\mathsf{SKE.sk},(\mathbf{x}_{i-2},\mathbf{r}_{i-2}),\ldots,\mathbf{x}_n,K_1,\ldots K_{i-3});\\ \mathsf{PRF}_{i-2}(K_{i-2},(\mathbf{r}_{i-2},\ldots,\mathbf{r}_{n-1}))\end{array}\right) & \text{if } i\neq 3\\ \mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk},(\mathbf{x}_1,\ldots,\mathbf{x}_n);\mathsf{PRF}_1(K_1,(\mathbf{r}_1,\ldots,\mathbf{r}_{n-1}))) & \text{if } i=3.\end{cases}
\end{aligned} \tag{6.21}$$

where in the first equality we use $\mathsf{prFE}_{i-1}.\mathsf{sk}=\mathsf{prFE}_{i-1}.\mathsf{KeyGen}(\mathsf{prFE}_{i-1}.\mathsf{msk},\mathrm{F}_{i-2})$ and $\mathsf{prFE}_{i-1}.\mathsf{ct}=\mathsf{prFE}_{i-1}.\mathsf{Enc}(\mathsf{prFE}_{i-1}.\mathsf{mpk},(\mathsf{SKE.sk},(\mathbf{x}_{i-1},\mathbf{r}_{i-1}),\ldots,(\mathbf{x}_{n-1},\mathbf{r}_{n-1}),\mathbf{x}_n,K_1,\ldots K_{i-2});\mathsf{PRF}_{i-1}(K_{i-1},(\mathbf{r}_{i-1},\ldots,\mathbf{r}_{n-1})))$ which follows from the assumption for i. The second equality follows from the definition of F_{i-2} and the correctness of the SKE scheme.

This completes the proof of the inductive step. ■

Using the above claim we get $\mathsf{prFE.ct}_1=\mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk},(\mathbf{x}_1,\ldots,\mathbf{x}_n);\mathsf{PRF}_1(K_1,(\mathbf{r}_1,\ldots,\mathbf{r}_{n-1})))$ from Step 6.3.2 of the decryption algorithm with probability 1. From the correctness of prFE_1 scheme, the decryption Step 6.3.2 outputs

$$\begin{aligned}
y=&\mathsf{prFE}_1.\mathsf{Dec}(\mathsf{prFE}_1.\mathsf{mpk},\mathsf{prFE}_1.\mathsf{sk}_f,f,\mathsf{prFE}_1.\mathsf{ct})\\
=&\mathsf{prFE}_1.\mathsf{Dec}(\begin{array}{r}\mathsf{prFE}_1.\mathsf{mpk},\mathsf{prFE}_1.\mathsf{sk}_f,f,\mathsf{prFE}_1.\mathsf{Enc}(\mathsf{prFE}_1.\mathsf{mpk},(\mathbf{x}_1,\ldots,\mathbf{x}_n);\\ \mathsf{PRF}_1(K_1,(\mathbf{r}_1,\ldots,\mathbf{r}_{n-1}))\end{array})
\end{aligned}$$

$$= f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$$

with probability 1. ■

6.3.3 Security Proof for General n

Theorem 6.21. *Let $\mathcal{SC}_{\mathsf{mi\text{-}prFE}}$ be a sampler class for* mi-prFE. *Suppose* prFE_i *scheme satisfies non-uniform* κ-prCT *security as per Definition 6.3 for $\kappa = \lambda^{n^2 \log \lambda}$ with respect to the sampler class that contains all $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$, induced by $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}} \in \mathcal{SC}_{\mathsf{mi\text{-}prFE}}$, as in Section 6.3.3, SKE satisfies sub-exponential INDr security and PRF_i is sub-exponentially secure, then* mi-prFE *constructed above satisfies security for $\kappa = \lambda^{n^2 \log \lambda}$ as in Definition 6.11. Note that this in particular implies the κ-security defined in Definition 6.10.*

Proof. Consider a sampler $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$ that generates the following:

1. **Key Queries.** It issues q_0 number of functions $f_1, \ldots, f_{q_0}$ for key queries.
2. **Ciphertext Queries.** It issues q_i number of messages for ciphertext queries for slot i. We use $\mathbf{x}_i^{j_i}$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.
3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security of mi-prFE as per Definition 6.11, we first define $\{\mathsf{Sim}_i\}_{i\in[n]}$ as follows. Observe that Sim_n outputs random string as is required by Definition 6.11.

$\mathsf{Sim}_i(\mathsf{msk}) \to \mathsf{ct}_i$ for $i \in [n-1]$.

- Parse $\mathsf{msk} = (\mathsf{SKE.sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]})$.
- Sample $\mathbf{r}_i \leftarrow \{0,1\}^\Lambda$ and $\gamma_i \leftarrow \mathcal{CT}_{\mathsf{SKE}}$.
- Compute $\mathsf{prFE}_{i+1}.\mathsf{sk} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i, \mathbf{r}_i, \mathsf{prFE}_i.\mathsf{mpk}])$.
- Output $\mathsf{ct}_i \coloneqq \mathsf{prFE}_{i+1}.\mathsf{sk}$.

$\mathsf{Sim}_n(\mathsf{msk}) \to \mathsf{ct}_n$. Sample $\delta_n \leftarrow \mathcal{CT}_{\mathsf{prFE}_n}$ and output $\mathsf{ct}_n := \delta_n$.
Then, it suffices to show

$$\left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]}, \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}}, \end{array}\right)$$
$$\approx_c \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]}, \left\{\gamma_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}} \end{array}\right) \tag{6.22}$$

where $(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{\mathbf{x}_i^{j_i}\}_{i,j_i}) \leftarrow \mathrm{Samp}_{\mathsf{mi\text{-}prFE}}(1^\lambda)$,

$$\left(\mathsf{mpk} = \{\mathsf{prFE}_i.\mathsf{mpk}\}_i, \mathsf{msk} = (\mathsf{SKE}.\mathsf{sk}, \{\mathsf{prFE}_i.\mathsf{msk}, \mathsf{prFE}_i.\mathsf{mpk}\}_i)\right) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm}),$$

$$\mathsf{prFE}_1.\mathsf{sk}_{f_k} \leftarrow \mathsf{prFE}_1.\mathsf{KeyGen}(\mathsf{prFE}_1.\mathsf{msk}, f_k) \quad \text{for } k \in [q_0],$$

$$\mathsf{SKE}.\mathsf{ct}_i^{j_i} \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathsf{SKE}.\mathsf{sk}, \mathbf{x}_i^{j_i}), \quad \gamma_i^{j_i} \leftarrow \mathcal{CT}_{\mathsf{SKE}}, \quad \mathbf{r}_i^{j_i} \leftarrow \{0,1\}^\Lambda,$$

$$\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \leftarrow \begin{cases} \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\mathsf{SKE}.\mathsf{ct}_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in LHS of Eq. (6.22)} \\ \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, \mathrm{F}_i[\gamma_i^{j_i}, \mathbf{r}_i^{j_i}, \mathsf{prFE}_i.\mathsf{mpk}]) & \text{in RHS of Eq. (6.22)} \end{cases}$$

$$\mathsf{prFE}_n.\mathsf{ct}^{j_n} \leftarrow \mathsf{prFE}_n.\mathsf{Enc}(\mathsf{prFE}_n.\mathsf{mpk}, (\mathsf{SKE}.\mathsf{sk}, \mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{n-1}^{j_n})),\ \delta_n^{j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_n},$$

$$K_i^{j_n} \leftarrow \{0,1\}^\Lambda, \quad \text{for } i \in [n-1],\ j_i \in [q_i],\ \text{and}\ \ j_n \in [q_n]$$

assuming we have

$$\left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{f_k,\ f_k(\mathbf{x}_1^{j_1}, \ldots \mathbf{x}_n^{j_n})\right\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}\right)$$
$$\approx_c \left(1^\kappa, \mathsf{aux}_{\mathcal{A}}, \left\{f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k\in[q_0], j_1\in[q_1], \ldots, j_n\in[q_n]}\right) \tag{6.23}$$

where $(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{\mathbf{x}_i^{j_i}\}_{i,j_i}) \leftarrow \mathrm{Samp}_{\mathsf{mi\text{-}prFE}}(1^\lambda)$, and $\Delta_k^{j_1,\ldots,j_n} \leftarrow \{0,1\}$. We prove this in the following two steps.

- **Step 1.** We first show that Equation (6.23) implies

$$\left(\begin{array}{c} 1^\kappa,\quad \mathsf{aux}_{\mathcal{A}},\quad \mathsf{prFE}.\mathsf{mpk}_1 \\ \{\mathsf{SKE}.\mathsf{ct}_i^{j_i}\}_{i\in[n-1], j_i\in[q_i]} \quad \left\{\mathsf{prFE}_1.\mathsf{ct}^{\mathbf{j}}\right\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{array}\right)$$

$$\approx_c \left(\begin{array}{c} 1^\kappa,\quad \mathsf{aux}_{\mathcal{A}},\quad \mathsf{prFE.mpk}_1 \\ \{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_{\mathsf{msg}}]},\quad \left\{\Delta^{\mathbf{j}}\right\}_{\mathbf{j}\in[q_1]\times\cdots\times[q_n]} \\ \left\{f_k,\ \mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \end{array}\right) \tag{6.24}$$

where $\mathbf{j}=(j_1,\ldots,j_n)\in[q_1]\times\cdots\times[q_n]$, $\Delta^{\mathbf{j}}\leftarrow\mathcal{CT}_{\mathsf{prFE}_1}$, and

$$\mathsf{prFE}_1.\mathsf{ct}^{\mathbf{j}}\leftarrow\mathsf{prFE}_1.\mathsf{Enc}\left(\mathsf{prFE}_1.\mathsf{mpk},(\mathbf{x}_1^{j_1},\ldots\mathbf{x}_n^{j_n})\right).$$

- **Step 2.** We prove that Equation (6.24) implies Equation (6.22).

Step 1. We show the following lemma.

Lemma 6.22. *If* SKE *satisfies subexponential* INDr *security, Equation* (6.23) *implies Equation* (6.24).

Proof. We first prove the following:

$$\begin{aligned}&\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ f_k(\mathbf{x}_1^{j_1},\ldots\mathbf{x}_n^{j_n})\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]},\left\{\mathsf{SKE.ct}_i^{j_i}\leftarrow\mathsf{SKE.Enc}(\mathsf{SKE.sk},\mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right)\\ &\approx_c\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k\in[q_0],j_1\in[q_1],\ldots,j_n\in[q_n]},\left\{\mathsf{SKE.ct}_i^{j_i}\leftarrow\mathsf{SKE.Enc}(\mathsf{SKE.sk},\mathbf{x}_i^{j_i})\right\}_{i\in[n-1],j_i\in[q_i]}\right),\end{aligned} \tag{6.25}$$

where $(\mathsf{aux}_{\mathcal{A}},\{f_k\}_k,\{\mathbf{x}_i^{j_i}\}_{i,j_i})\leftarrow\mathsf{Samp}(1^\lambda)$, $\mathsf{SKE.sk}\leftarrow\mathsf{SKE.Setup}(1^\Lambda)$, and $\Delta_k^{j_1,\ldots,j_n}\leftarrow\{0,1\}$. To prove this, we observe

$$\begin{aligned}&\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n},\left\{\mathsf{SKE.ct}_i^{j_i}\leftarrow\mathsf{SKE.Enc}(\mathsf{SKE.sk},\mathbf{x}_i^{j_i})\right\}_{i,j_i}\right)\\ &\approx_c\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ f_k(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\right\}_{k,j_1,\ldots,j_n},\left\{\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) &(6.26)\\ &\approx_c\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n},\left\{\gamma_i^{j_i}\leftarrow\mathcal{CT}_{\mathsf{SKE}}\right\}_{i,j_i}\right) &(6.27)\\ &\approx_c\left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\left\{f_k,\ \Delta_k^{j_1,\ldots,j_n}\right\}_{k,j_1,\ldots,j_n},\left\{\mathsf{SKE.ct}_i^{j_i}\leftarrow\mathsf{SKE.Enc}(\mathsf{SKE.sk},\mathbf{x}_i^{j_i})\right\}_{i,j_i}\right). &(6.28)\end{aligned}$$

Here, we justify each step of the equations above. We can see that Equation (6.26) follows from subexponential INDr security of SKE, since SKE.sk is used only for computing $\{\mathsf{SKE.ct}_i^{j_i}\}_{i,j_i}$ and not used anywhere else. Note that by our choice of the parameter $2^{\Lambda^\delta}>\kappa^{\omega(1)}$, we can use the security of SKE even for an adversary who runs in time

polynomial in κ. We can see that Equation (6.27) follows from Equation (6.23) by noting that adding random strings does not make the task of disginguishing the two distributions any easier. Finally, Equation (6.28) follows from INDr security of SKE again.

We then consider a sampler Samp_1 that on input 1^κ outputs

$$\left(f_1,\ldots,f_{q_0},\ \{(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n})\}_{j_1\in[q_1],\ldots,j_n\in[q_n]},\ \mathsf{aux}_1 \stackrel{\text{def}}{=} \left(1^\kappa,\mathsf{aux}_{\mathcal{A}},\{\mathsf{SKE.ct}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}\right)\right).$$

By the security guarantee of prFE_1 with sampler Samp_1 and Equation (6.25), we obtain Equation (6.24). ■

Step 2. To prove that Equation (6.24) implies Equation (6.22), we prove the following statement.

Lemma 6.23. *For $h \in [n]$ and an adversary $\mathcal{A}$, let us consider the following distinguishing advantage:*

$$\mathsf{Adv}^h_{\mathcal{A}}(\lambda) \stackrel{def}{=} \left|\mathcal{A}\left(\begin{array}{r}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i},\mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}\\ \{f_k,\mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]},\\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\mathsf{prFE}_h.\mathsf{ct}^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]},\end{array}\right)\right.$$
$$\left. - \mathcal{A}\left(\begin{array}{r}\mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE.mpk}_i\}_{i\in[h]},\ \{\mathsf{SKE.ct}_i^{j_i},\mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}\\ \{f_k,\mathsf{sk}_{f_k}=\mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]},\\ \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h-1],j_i\in[q_i]},\ \{\Delta^{\mathbf{j}}\}_{\mathbf{j}\in[q_h]\times\cdots\times[q_n]}\end{array}\right)\right| \quad (6.29)$$

where $\mathbf{j} = (j_h,\ldots,j_n)$, $\Delta^{\mathbf{j}} \leftarrow \mathcal{CT}_{\mathsf{prFE}_h}$, $\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i} \leftarrow \mathsf{prFE}_{i+1}.\mathsf{KeyGen}(\mathsf{prFE}_{i+1}.\mathsf{msk}, F_i[\mathsf{SKE.ct}_i^{j_i},\mathbf{r}_i,\mathsf{prFE}_i.\mathsf{mpk}])$, *and*

$$\mathsf{prFE}_h.\mathsf{ct}^{\mathbf{j}} \leftarrow \mathsf{prFE}_h.\mathsf{Enc}\left(\mathsf{prFE}_h.\mathsf{mpk},\left(\mathsf{SKE.sk},(\mathbf{x}_h^{j_h},\mathbf{r}_h^{j_h}),\ldots,(\mathbf{x}_{n-1}^{j_{n-1}},\mathbf{r}_{n-1}^{j_{n-1}}),(\mathbf{x}_n^{j_n},K_1^{j_n},\ldots,K_{h-1}^{j_n})\right)\right).$$

Then, for every $h^* := \{h^*_\lambda \in [2,n(\lambda)]\}_\lambda$ and every non-uniform adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ such that $\mathrm{Size}(\mathcal{A}) < \mathrm{poly}(\kappa)$,[11] *there exists another non-uniform adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$*

[11] Here, we deviate from our convention that the adversary runs in polynomial time in its input length. Note that κ here may be super-polynomial in the input length to $\mathcal{A}$.

and a polynomial Q such that

$$\mathsf{Adv}_{\mathcal{B}}^{h^*-1}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}}^{h^*}(\lambda)/Q(\lambda') - \mathsf{negl}(\kappa) \qquad \textit{and} \qquad \mathrm{Size}(\mathcal{B}) \leq Q(\lambda') \cdot \mathrm{Size}(\mathcal{A})$$

assuming the security of prFE *as per Definition 6.3 with respect to* κ *and the subexponential security of* PRF*, where* $\lambda' := \lambda^n$.

Proof. We invoke the security of prFE_{h^*} with non-uniform sampler Samp_{h^*} that takes as input the security parameter 1^λ and outputs

$$\left(\begin{array}{rc} \text{Functions:} & \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}} = \mathrm{F}_{h^*-1}^{j_{h^*-1}}[\mathsf{SKE.ct}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*-1}.\mathsf{mpk}]\right\}_{j_{h^*-1}\in[q_{h^*-1}]}, \\ \text{Inputs:} & \left\{\mathbf{x}^{j_{h^*},\ldots,j_n} \stackrel{\text{def}}{=} \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*}^{j_{h^*}}, \mathbf{r}_{h^*}^{j_{h^*}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-1}^{j_n})\right)\right\}_{j_{h^*}\in[q_{h^*}],\ldots,j_n\in[q_n]}, \\ \text{Auxiliary Information:} & \mathsf{aux}_{h^*} \stackrel{\text{def}}{=} \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}}, \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[h^*-1]}, \{f_k, \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\}_{k\in[q_0]}, \\ \{\mathsf{SKE.ct}_i^{j_i}, \mathbf{r}_i^{j_i}\}_{i\in[n-1],j_i\in[q_i]}, \{\mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\}_{i\in[h^*-2],j_i\in[q_i]} \end{array}\right) \end{array}\right).$$

We can see that the size of Samp is $\mathrm{poly}(\lambda^n) = \mathrm{poly}(\lambda')$, since $q_0 q_1 \ldots q_n = \mathrm{poly}(\lambda^n)$.

We also consider the following distributions:

$$\left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathsf{prFE}_{h^*}.\mathsf{Enc}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$

$$\text{and} \quad \left(\mathsf{prFE}_{h^*}.\mathsf{mpk}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}, \mathsf{prFE}_{h^*}.\mathsf{sk}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\Delta^{j_{h^*},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}}\right\}_{j_{h^*},\ldots,j_n}, \mathsf{aux}_{h^*}\right). \tag{6.30}$$

By the security guarantee of prFE_{h^*} with respect to Samp_{h^*}, we can see that an adversary $\mathcal{A}$ that can distinguish the distributions in Equation (6.30) with advantage more than ϵ can be converted into another adversary $\mathcal{B}$ that can distinguish the following distributions with advantage more than $\epsilon' \stackrel{\text{def}}{=} \epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ satisfying $\mathrm{Size}(\mathcal{B}) \leq Q(\lambda')\mathrm{Size}(\mathcal{A})$ for some polynomial Q:

$$\left(\left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\right\}_{j_{h^*-1}}, \left\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n})\right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*}\right)$$

$$\text{and} \quad \left(\left\{ \mathrm{F}_{h^*-1}^{j_{h^*-1}} \right\}_{j_{h^*-1}}, \left\{ \Delta^{j_{h^*-1},\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{prFE}_{h^*}} \right\}_{j_{h^*-1},\ldots,j_n}, \mathsf{aux}_{h^*} \right). \tag{6.31}$$

By inspection, one can see that the distributions in Equation (6.30) are equivalent to those in Equation (6.29) with $h = h^*$.[12] Therefore, to complete the proof, it suffices to show that $\mathcal{B}$ can be used as a distinguisher against the distributions in Equation (6.29) with $h = h^* - 1$ whose advantage is at least $\epsilon' - \mathsf{negl}(\kappa)$. To show this, we first observe that the second distribution in Equation (6.31) is equivalent to that in Equation (6.29) with $h = h^* - 1$. Therefore, it suffices to prove that $\mathcal{A}$ cannot distinguish the first distribution in Equation (6.31) from the first distribution in Equation (6.29) with $h = h^* - 1$ with more than negligible advantage in κ. To show this, we consider the following sequence of hybrids.

Hyb_1. This is the first distribution of Equation (6.31). Recall that we have

$$\mathrm{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*},\ldots,j_n}) = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\begin{array}{r} \mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-2}^{j_n}) \right); \\ \mathsf{PRF}_{h^*-1}(K_{h^*-1}^{j_n}, (\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}})) \end{array} \right).$$

Hyb_2. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}_{h^*-1}(K_{h^*-1}^{j_n}, \cdot)$ with the real random function $\mathsf{R}^{j_n}(\cdot)$ for each $j_n \in [q_{\mathsf{msg}}]$. Since $K_{h^*-1}^{j_n}$ is not used anywhere else, we can use the subexponential security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\mathsf{negl}(\kappa)$.

Hyb_3. This hybrid is the same as the previous one except that we output a failure symbol when there exist $(j_{h^*-1}, \ldots, j_{n-1}) \neq (j'_{h^*-1}, \ldots, j'_{n-1})$ such that

[12] Equation (6.30) includes additional terms $\{\mathrm{F}_{h^*-1}^{j_{h^*-1}}\}_{j_{h^*-1}}$ while Equation (6.29) does not. We ignore this difference, since theses terms can be efficiently computed from aux_{h^*} and does not affect the indistinguishability.

$(\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}}) = (\mathbf{r}_{h^*-1}^{j'_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j'_{n-1}})$. We show that the probability of this happening is negligible in κ. To prove this, it suffices to show that there are no $i \in [h^* - 1, n-1]$, $j, j' \in [q_i]$ satisfying $j \neq j'$ and $\mathbf{r}_i^j = \mathbf{r}_i^{j'}$. The probability of this happening can be bounded by $(q_1^2 + \cdots q_{n-1}^2)/2^{2\Lambda}$ by taking the union bound with respect to all the combinations of i, j, j'. By our choice of Λ, this is bounded by $\mathsf{negl}(\kappa)$.

Hyb_4. In this hybrid, we replace $\mathsf{F}_{h^*-1}^{j_{h^*-1}}(\mathbf{x}^{j_{h^*}, \ldots, j_n})$ with

$$\mathsf{prFE}_{h^*-1}.\mathsf{ct}^{j_{h^*-1}, \ldots, j_n} = \mathsf{prFE}_{h^*-1}.\mathsf{Enc}\left(\begin{array}{r} \mathsf{prFE}_{h^*-1}.\mathsf{mpk}, \\ \left(\mathsf{SKE.sk}, (\mathbf{x}_{h^*-1}^{j_{h^*-1}}, \mathbf{r}_{h^*-1}^{j_{h^*-1}}), \ldots (\mathbf{x}_n^{j_n}, K_1^{j_n}, \ldots, K_{h^*-2}^{j_n})\right) \end{array}\right).$$

Namely, we use fresh randomness for each encryption instead of deriving the randomness by $\mathsf{R}^{j_n}(\mathbf{r}_{h^*-1}^{j_{h^*-1}}, \ldots, \mathbf{r}_{n-1}^{j_{n-1}})$. We claim that this change is only conceptual. To see this, we observe that unless the failure condition introduced in Hyb_3 is satisfied, every invocation of the function R^{j_n} is with respect to a fresh input and thus the output can be replaced with a fresh randomness.

Hyb_5. In this hybrid, we remove the failure event. Namely, we always outputs $\mathsf{prFE}_{h^*-1}.\mathsf{ct}_{j_{h^*-1}, \ldots, j_n}$ regardless of whether the failure event happens or not. Since the failure event happens with probability only $\mathsf{negl}(\kappa)$ probability, we conclude that the adversary is not able to distinguish this hybrid from the previous one with more than $\mathsf{negl}(\kappa)$ probability.

Noting that the final hybrid is equivalent to the first distribution in Equation (6.29) with $h = h^* - 1$, we complete the proof. ∎

Lemma 6.24. *Assuming that* $\mathsf{Adv}_{\mathcal{A}}^1(\lambda)$ *(defined in Equation* (6.29)*) is negligible in* κ *for all non-uniform adversary* $\mathcal{A}$ *such that* $\mathrm{Size}(\mathcal{A}) = \mathrm{poly}(\kappa)$*, we have* $\mathsf{Adv}_{\mathcal{B}}^n(\lambda) = \mathsf{negl}(\lambda)$ *for all non-uniform adversary* $\mathcal{B}$ *such that* $\mathrm{Size}(\mathcal{B}) = \mathrm{poly}(\lambda)$.

Proof. For the sake of contradiction, suppose that there exists an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$

such that $\epsilon_n(\lambda) \stackrel{\mathrm{def}}{=} \mathsf{Adv}^n_{\mathcal{B}}(\lambda)$ is non-negligible and $t_n(\lambda) \stackrel{\mathrm{def}}{=} \mathrm{Size}(\mathcal{B})$ is polynomial in λ. In particular, this implies that there exists an infinite set $\mathcal{L} \subseteq \mathbb{N}$ and some polynomial p such that $\epsilon_n(\lambda) \geq 1/p(\lambda)$ and $t_n(\lambda) \leq p(\lambda)$ for all $\lambda \in \mathcal{L}$. We then define $t_i(\lambda) \stackrel{\mathrm{def}}{=} p(\lambda)\lambda^{n(n-i)\log\lambda}$ and $\epsilon_i(\lambda) \stackrel{\mathrm{def}}{=} 1/p(\lambda)\lambda^{n(n-i)\log\lambda}$ for $i = 1, \ldots, n-1$. We can also see that $t_1(\lambda) < \kappa$ and $\epsilon_1(\lambda) > 1/\kappa$ for sufficiently large λ. This implies that there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$, there is no adversary $\mathcal{A}_\lambda$ such that $\mathrm{Size}(\mathcal{A}_\lambda) \leq t_1(\lambda)$ and $\mathsf{Adv}^1_{\mathcal{A}_\lambda}(\lambda) \geq \epsilon_1(\lambda)$. We then consider the following statement that is parameterized by λ and $h \in [n]$:

$\mathrm{Statement}_{\lambda,h}$: There exists an adversary $\mathcal{A}_\lambda$ such that $\mathrm{Size}(\mathcal{A}_\lambda) \leq t_h(\lambda)$ and $\mathsf{Adv}^h_{\mathcal{A}_\lambda}(\lambda) \geq \epsilon_h(\lambda)$.

For each $\lambda \in \mathcal{L} \cap \mathbb{N}_{>\lambda_0}$, there exists $h^*_\lambda \in [2, n]$ such that $\mathrm{Statement}_{\lambda,h^*_\lambda-1}$ is false and $\mathrm{Statement}_{\lambda,h^*_\lambda}$ is true, since $\mathrm{Statement}_{\lambda,1}$ is false and $\mathrm{Statement}_{\lambda,n}$ is true. However, by applying Theorem 6.23 to the adversary guaranteed by $\mathrm{Statement}_{\lambda,h^*_\lambda}$ being true for the sequence $\{h^*_\lambda\}_\lambda$, we obtain another adversary $\mathcal{A}' = \{\mathcal{A}'_\lambda\}_\lambda$ such that $\mathrm{Size}(\mathcal{A}'_\lambda) \leq t_{h^*}Q(\lambda^n)$ and $\mathsf{Adv}^{h^*-1}_{\mathcal{A}'_\lambda}(\lambda) \geq \epsilon_{h^*}/Q(\lambda^n) - \mathsf{negl}(\kappa) \geq \epsilon_{h^*}/2Q(\lambda^n)$ for some polynomial Q. In particular, $\mathrm{Size}(\mathcal{A}'_\lambda) \leq t_{h^*-1}(\lambda)$ and $\mathsf{Adv}^{h^*-1}_{\mathcal{A}'_\lambda}(\lambda) \geq \epsilon_{h^*-1}(\lambda)$ for all sufficiently large λ, since we have $\lambda^{n\log\lambda} > Q(\lambda^n)$ for all sufficiently large λ. However, this contradicts the above assertion that $\mathrm{Statement}_{\lambda,h^*_\lambda-1}$ is false. This concludes the proof. ∎

The following lemma completes Step 2 of the proof of Theorem 6.21.

Lemma 6.25. *If* SKE *is* INDr *secure, Equation* (6.24) *implies Equation* (6.22).

Proof. We first observe that Equation (6.24) is equivalent to saying $\mathsf{Adv}^1_{\mathcal{A}}(\lambda) = \mathsf{negl}(\kappa)$ for all $\mathcal{A}$ with $\mathrm{Size}(\mathcal{A}) = \mathrm{poly}(\kappa)$. By Theorem 6.24, this implies that $\mathsf{Adv}^n_{\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ for all $\mathcal{A}$ with $\mathrm{Size}(\mathcal{A}) = \mathrm{poly}(\lambda)$. Namely, we have

$$\left(\begin{array}{l} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\mathsf{prFE}_n.\mathsf{ct}^{j_n}\right\}_{j_n\in[q_n]} \end{array}\right)$$

$$\approx_c \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \end{array} \right).$$

By INDr security of SKE, we have

$$\left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\mathsf{SKE}.\mathsf{ct}_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \end{array} \right)$$
$$\approx_c \left(\begin{array}{c} \mathsf{aux}_{\mathcal{A}},\ \{\mathsf{prFE}_i.\mathsf{mpk}\}_{i\in[n]},\ \left\{f_k,\ \mathsf{sk}_{f_k} = \mathsf{prFE}_1.\mathsf{sk}_{f_k}\right\}_{k\in[q_0]} \\ \left\{\gamma_i^{j_i},\ \mathbf{r}_i^{j_i},\ \mathsf{prFE}_{i+1}.\mathsf{sk}^{j_i}\right\}_{\substack{i\in[n-1],\\ j_i\in[q_i]}},\ \left\{\delta_n^{j_n}\right\}_{j_n\in[q_n]} \end{array} \right).$$

Combining the above equations, Equation (6.22) readily follows. ■

We conclude the proof of Theorem 6.21. ■

Remark 35 (Comparison with [164])**.** We note that in high level, overall structure of our security proof above is similar to that of witness encryption in [164]. In both proofs, the main step considers parameterized distributions $\{\mathcal{D}_{h,b}\}_{h\in[n],b\in\{0,1\}}$ and shows that $\mathcal{D}_{1,0} \approx_c \mathcal{D}_{1,1}$ holds if $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$ are indistinguishable even with subexponentially small advantage against subexponential time adversary. To show this claim, [164] uses the evasive LWE assumption, while we use the security of prFE, which in turn is reduced to the evasive LWE assumption. While this difference stems simply from the fact that we introduce the intermediate primitive of prFE to construct mi-prFE instead of directly constructing it from evasive LWE, there are more fundamental differences as well. In particular, we identify certain subtle issues in the proof by [164] and fix these by strengthening the assumptions. We elaborate on this in the following.

- **On the multiplicative invocation of evasive LWE.** To prove $\mathcal{D}_{1,0} \approx_c \mathcal{D}_{1,1}$, [164] assumes that there exists an adversary $\mathcal{A}_1$ that distinguishes them with non-negligible advantage ϵ and polynomial time t for the sake of contradiction. They then invoke the evasive LWE assumption with respect to an appropriately defined sampler Samp_1 to conclude that there exists a distinguishing adversary $\mathcal{A}_2$ against $\mathcal{D}_{2,0}$ and $\mathcal{D}_{2,1}$. This process continues multiple times, where they invoke evasive LWE with respect to the security parameter $\lambda_j := 2^j\lambda$ and an adversary $\mathcal{A}_j$ for

the j-th invocation to obtain another adversary $\mathcal{A}_{j+1}$, where $\mathcal{A}_k$ is a distinguisher against $\mathcal{D}_{k,0}$ and $\mathcal{D}_{k,1}$. Denoting the distinguishing advantage against $\mathcal{D}_{j,0}$ and $\mathcal{D}_{j,1}$ of $\mathcal{A}_j$ by ϵ_j, we have $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}_j(\lambda_j)$, where poly_j is a polynomial that is determined by the sampler Samp_j. Finally, they obtain a distinguisher $\mathcal{A}_n$ against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$, where $\epsilon_n = \epsilon/\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$ and the running time being $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n)$. They derive the conclusion by saying

$$\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n) = \mathrm{poly}(2^{n^2}\lambda^n) \tag{6.32}$$

and setting the parameter so that there is no adversary of this running time and distinguishing advantage against $\mathcal{D}_{n,0}$ and $\mathcal{D}_{n,1}$. However, a subtlety is that $\mathrm{poly}_1(\lambda_1)\mathrm{poly}_2(\lambda_2)\cdots\mathrm{poly}_n(\lambda_n) = \mathrm{poly}(\lambda_1\cdots\lambda_n)$ is not necessarily true. For example, one can consider the setting where we have $\mathrm{poly}_j(\lambda) = \lambda^{2^j}$. This example may look a bit artificial, but it does not contradict the evasive LWE assumption, since j is treated as a constant asymptotically. In words, the issue arises from the fact that even if each polynomial has a constant exponent, the maximum of the exponents can be arbitrarily large function in λ, when we consider non-constant number of polynomials. In this setting, $\mathcal{A}_n$'s distinguishing advantage is too small to derive the contradiction.

The above issue occurs due to the invocations of evasive LWE super-constant times. To resolve the problem, we consider non-uniform sampler $\{\mathrm{Samp}_{h^*}\}_{h^*}$ that hardwires the "best" index h^* and invoke the evasive LWE only with respect to this sampler (See Theorem 6.23 and 6.24). We avoid the above problem, since we invoke the evasive LWE only once in the proof. However, this solution entails the strengthening of the assumption where we consider non-uniform samplers. We believe that the same strengthening of the assumption is required for the proof in [164] as well.

- **On the additive term of evasive LWE.** Here, we also discuss the other subtlety that arises in the proof by [164]. To focus on the issue, we ignore the first issue discussed above and assume $\mathrm{poly}_j(\lambda) = \lambda^c$ holds for some fixed $c \in \mathbb{N}$ that does not depend on j, which makes Equation (6.32) correct. In the proof of [164] (and in our explanation above), we implicitly ignore the negligible additive term when applying evasive LWE. Namely, when we apply the assumption with respect to $\mathcal{A}_j$, the lower bound for the advantage ϵ_{j+1} of $\mathcal{A}_{j+1}$ should be $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j) - \mathsf{negl}(\lambda_j)$ rather than $\epsilon_{j+1} \geq \epsilon_j/\mathrm{poly}(\lambda_j)$. This does not cause any difference when we consider the setting where ϵ_j is non-negligible in λ_j. However, for larger j, ϵ_j is negligible function in the security parameter λ_j. Concretely, the lower bound on ϵ_{n-1} obtained by ignoring the additive term is $\epsilon/\mathrm{poly}(2^{(n-1)^2}\lambda^{n-1})$.[13] If we apply evasive LWE once more with respect to $\lambda_n = 2^n\lambda^n$ to complete the proof, we have $\epsilon_n \geq \epsilon_{n-1}/\mathrm{poly}(\lambda_n) - \mathsf{negl}(\lambda_n)$. The RHS of the inequality may be negative, since $\epsilon_{n-1}/\mathrm{poly}(\lambda_n)$ is some specific negligible function in λ_n and this may be smaller than the second term $\mathsf{negl}(\lambda_n)$. Therefore, what we can derive here is the trivial

[13] Namely, the actual value of ϵ_{n-1} may be even smaller.

bound $\epsilon_n \geq 0$, which is not enough for our purpose. To fix this issue, we introduce additional parameter κ and then modify the assumption so that the additive term is negligibly small in κ, which is set much larger than λ. Again, we believe that the same strengthening of the assumption is required for the proof in [164] as well. Finally, we note that the above problem does not occur when n is constant. This is because in that case, we have ϵ_n is non-negligible and thus the problem of ϵ_n may be smaller than negl_n does not occur.

6.3.4 Security Proof for Constant n (with Weaker Assumption)

Here, we prove the security of our construction in the case of n being a constant. The reason why we consider the security proof separately for this special case is that we can give a proof from better assumptions than the general case. In more detail, the security is proven assuming the standard security notion for prFE, rather than the non-uniform and κ version of it. As a result, the security of the mi-prFE is reduced to the the (plain) evasive LWE instead of non-uniform κ-evasive LWE. The reason why we can achieve this is that in the case of n being constant, we can avoid all the subtleties that arise in the general case. We refer to Remark 35 for more discussions.

Theorem 6.26. *Let $\mathcal{SC}_{\mathsf{mi\text{-}prFE}}$ be a samples class for* mi-prFE. *Suppose* prFE_i *scheme satisfies* prCT *security as per Definition 5.13 with respect to the sampler class that contains all* $\mathsf{Samp}_{\mathsf{prFE}}(1^\lambda)$*, induced by* $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}} \in \mathcal{SC}_{\mathsf{mi\text{-}prFE}}$*, as in Section 6.3.3, SKE satisfies INDr security and* PRF_i *is secure, then* mi-prFE *constructed in Section 6.3.2 for constant n satisfies security for $\kappa = \lambda^n$ as in Definition 6.11. Note that this in particular implies the κ-security defined in Definition 6.10.*

Proof. The proof of this theorem largely follows that of Theorem 6.21. The crucial difference is that to get the equivalent of Theorem 6.24, we simply invoke the (non-κ, uniform) security of prFE n-times. We sketch the proof below while highlighting the difference. We consider the same simulator as in the proof of Theorem 6.21 and divide the proof steps into Step 1 and Step 2 in the same manner.

- We start with Step 1, which consists of proving that Equation (6.23) implies Equation (6.24). This is proven in the same manner as Theorem 6.22. However here, since we set $\kappa = \lambda^n = \mathrm{poly}(\lambda)$, we do not need subexponential INDr security

and only (polynomial) INDr security suffices for SKE.

- We then move to prove Step 2. The goal here is to prove Equation (6.24) implies Equation (6.22).

  - We first observe that the uniform version of Theorem 6.23 holds by the same proof, where we only consider constant h^* rather than arbitrary sequence $h^* = \{h^*_\lambda\}_\lambda$ and uniform PPT adversaries. Notice that then the sampler is now uniform, since it no longer has to hardwire the sequence $\{h^*_\lambda\}_\lambda$. In this setting, non-uniform κ-prCT security collapses to the (plain) prCT security, since $\kappa = \lambda^n = \mathrm{poly}(\lambda)$ and the sampler is uniform. Therefore, plain prCT security is sufficient for the proof. Furthermore, since we set $\kappa = \lambda^n = \mathrm{poly}(\lambda)$, we do not need subexponential security for PRF and standard security suffices.

  - We then consider an analogue of Theorem 6.24, which asserts that if $\mathsf{Adv}^1_{\mathcal{A}}(\lambda)$ (defined in Equation (6.29)) is negligible for all (uniform) PPT adversary that runs in polynomial time in λ, then so is $\mathsf{Adv}^n_{\mathcal{A}}(\lambda)$. This is proven by observing that the indistinguishability of the distributions in Equation (6.29) for $h = h^* - 1$ implies that for h^* by the analogue of Theorem 6.23 explained in the previous item. By applying this n-times, we obtain the conclusion.

  - We finally conclude the proof by the same argument as Theorem 6.25.

This completes the proof of Theorem 6.26. ■

We encapsulate the results of this section using the following theorems.

Theorem 6.27 (mi-prFE for poly arity)**.** *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assume non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 2.5). Then there exists a* mi-prFE *scheme for arity $n = \mathrm{poly}(\lambda)$, supporting functions with input length L and bounded polynomial depth $d = d(\lambda)$, and satisfying κ-security (Definition 6.10) with efficiency $|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$, $|\mathsf{ct}_1| = nL\mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}_i| = \mathrm{poly}(n, L, d, \Lambda, \lambda)$ for $i \in [2, n]$ where $\Lambda := (n^2\lambda)^{1/\delta}$.*

Theorem 6.28 (mi-prFE for constant arity)**.** *Let $\kappa = \lambda^n$. Assume evasive LWE (Assumption 2.6) and LWE (Assumption 2.5). Then there exists a* mi-prFE *scheme for arity $n = O(1)$, supporting functions with input length L and bounded polynomial depth $d = d(\lambda)$, and satisfying κ-security (Definition 6.10) with efficiency $|\mathsf{mpk}| = \mathrm{poly}(n, L, d, \lambda)$, $|\mathsf{sk}_f| = \mathrm{poly}(d, \lambda)$, $|\mathsf{ct}_1| = nL\mathrm{poly}(\mathsf{dep}, \lambda)$, $|\mathsf{ct}_i| =$*

$\mathrm{poly}(n, L, d, \lambda)$ *for* $i \in [2, n]$.

6.4 MULTI-INPUT PREDICATE ENCRYPTION FOR POLYNOMIAL ARITY FOR P

In this section, we provide our construction of multi-input predicate encryption (miPE) for all circuits as an application of mi-prFE. Similarly to Section 6.3, we consider two settings where the arity is either constant or arbitrary polynomial. The former setting leads to a construction with weaker assumption where we only require (plain) evasive LWE and LWE. This improves the construction by [22], which additionally requires non-standard tensor LWE assumption. The latter setting leads to a construction with polynomial arity and for all circuits under the stronger non-uniform κ-evasive LWE assumption. This resolves the open problem posed by [22].

6.4.1 Construction

In this section, we give a construction of a miPE scheme using mi-prFE and PE. The construction will support functions with arity $n = n(\lambda)$ where input string for each arity is in $\{0,1\}^L$ and the output is $\{0,1\}$. We further restrict the depth of the circuits that implement the function by a parameter dep. We denote this function class by $\mathcal{F}_{\mathsf{prm}}$, where prm is the set of the parameters n, L, dep. Namely, $\mathcal{F}_{\mathsf{prm}}$ consists of n-ary functions that takes as input strings $x_1, \ldots, x_n$, where each $x_i \in \{0,1\}^L$ and outputs $f(x_1, \ldots, x_n) \in \{0,1\}$. The message space for each slot is $\{0,1\}$. Namely, we have $\mu_1, \ldots, \mu_n \in \{0,1\}$ in the following.

Building Blocks. Below, we list the building blocks required for our construction.

1. A single input predicate encryption scheme $\mathsf{PE} = \mathsf{PE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for function family supporting functions $f : \{0,1\}^{nL} \to \{0,1\}$ that can be represented as circuits with depth at most dep. We denote by this function class by $\mathcal{F}_{\mathsf{prm}_{\mathsf{PE}}}$, where $\mathsf{prm}_{\mathsf{PE}} = (1^{nL}, 1^{\mathsf{dep}})$ is the parameter that specifies the circuit class. We also assume that the scheme has message space $\{0,1\}^n$ and satisfies INDr security (Definition 6.5). We can

instantiate such a PE by using the construction by [109] or by the combination of lockable obfuscation [172] (a.k.a compute-and-compare obfuscation [172]) and ABE for circuits by [43], for example.

We use $\mathcal{CT}_{\mathsf{PE}}$ to denote the ciphertext space, $\ell_{\mathsf{ct}}^{\mathsf{PE}}$ to denote the ciphertext length and $d_{\mathsf{Enc}}^{\mathsf{PE}}$ to denote the depth of the circuit required to compute the $\mathsf{PE.Enc}$ algorithm.

2. A n-input FE for pseudorandom functionalities $\mathsf{mi\text{-}prFE} = \mathsf{mi\text{-}prFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$ for function family $\mathcal{F}_{L'(\lambda), d_{\mathsf{Enc}}^{\mathsf{PE}}}$ consisting of circuits with input space $\{0,1\}^{L'}$ and output space $\{0,1\}$ where we set $L' = L + \lambda + 1$ for our construction and with maximum depth $d_{\mathsf{Enc}}^{\mathsf{PE}}$. We denote the parameters that specify $\mathcal{F}_{L'(\lambda), d_{\mathsf{Enc}}^{\mathsf{PE}}}$ by $\mathsf{prm}_{\mathsf{mi\text{-}prFE}}$.

3. A pseudorandom function $\mathsf{PRF} : \{0,1\}^{\Lambda} \times \{0,1\}^{(n-1)\Lambda} \to \{0,1\}^{\mathsf{R}_{\mathsf{len}}}$, where $\{0,1\}^{\Lambda}$ and $\{0,1\}^{(n-1)\Lambda}$ are the key space and input space respectively and $\mathsf{R}_{\mathsf{len}}$ is the length of randomness used in the $\mathsf{PE.Enc}$ algorithm. We will discuss how to set Λ in Remark 36.

Next, we describe our construction.

$\mathsf{Setup}(1^{\lambda}, 1^{n}, \mathsf{prm}) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm does the following.

- Generate $(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.msk}) \leftarrow \mathsf{mi\text{-}prFE.Setup}(1^{\lambda}, 1^{n}, \mathsf{prm}_{\mathsf{mi\text{-}prFE}})$.
- Generate $(\mathsf{PE.mpk}, \mathsf{PE.msk}) \leftarrow \mathsf{PE.Setup}(1^{\Lambda}, \mathsf{prm}_{\mathsf{PE}})$.
- Output $\mathsf{mpk} = (\mathsf{mi\text{-}prFE.mpk}, \mathsf{PE.mpk})$ and $\mathsf{msk} = (\mathsf{mi\text{-}prFE.msk}, \mathsf{PE.msk})$.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, f) \to \mathsf{sk}_f$.The key generation algorithm does the following.

- Parse $\mathsf{mpk} = (\mathsf{mi\text{-}prFE.mpk}, \mathsf{PE.mpk})$ and $\mathsf{msk} = (\mathsf{mi\text{-}prFE.msk}, \mathsf{PE.msk})$.
- Compute $\mathsf{PE.sk}_f \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f)$.
- Compute $\mathsf{mi\text{-}prFE.sk}_{\mathrm{F}} \leftarrow \mathsf{mi\text{-}prFE.KeyGen}(\mathsf{mi\text{-}prFE.msk}, \mathrm{F}[\mathsf{PE.mpk}])$ where $\mathrm{F}[\mathsf{PE.mpk}]$ is defined as follows:

$$\mathrm{F}[\mathsf{PE.mpk}]\left((\mathbf{x}_1, \mu_1, \mathbf{r}_1), \ldots, (\mathbf{x}_{(n-1)}, \mu_{(n-1)}, \mathbf{r}_{(n-1)}), (\mathbf{x}_n, \mu_n, \mathbf{r}_n)\right)$$
$$= \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))\right)$$

– Output $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}})$[14].

$\mathsf{Enc}_i(\mathsf{msk}, \mathbf{x}_i, \mu_i) \to \mathsf{ct}_i$ for $1 \leq i \leq n$.The slot i encryption algorithm does the following.

– Parse $\mathsf{msk} = (\mathsf{mi\text{-}prFE.msk}, \mathsf{PE.msk})$.

– For $1 \leq i \leq n$, sample $\mathbf{r}_i \leftarrow \{0,1\}^\Lambda$ and compute $\mathsf{mi\text{-}prFE.ct}_i \leftarrow \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{mi\text{-}prFE.msk}, (\mathbf{x}_i, \mu_i, \mathbf{r}_i))$.

– Output $\mathsf{ct}_i \coloneqq \mathsf{mi\text{-}prFE.ct}_i$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \to y \in \{0,1\}^n \cup \{\bot\}$. The decryption algorithm does the following.

– Parse $\mathsf{mpk} = (\mathsf{mi\text{-}prFE.mpk}, \mathsf{PE.mpk})$, $\mathsf{sk}_f = (\mathsf{PE.sk}_f, \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}})$ and $\{\mathsf{ct}_i = \mathsf{mi\text{-}prFE.ct}_i\}_{i\in[n]}$.

– Compute $\mathsf{PE.ct} = \mathsf{mi\text{-}prFE.Dec}(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}, \mathrm{F}, \mathsf{mi\text{-}prFE.ct}_1, \ldots, \mathsf{mi\text{-}prFE.ct}_n)$.

– Compute $y = \mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.ct})$.

– Output y.

Remark 36. Here, we discuss how we set Λ. Similarly to the case of mi-prFE in Section 6.3, we consider two cases of parameter settings for the construction. One is the case of n being constant. In this case, we simply set $\Lambda = \lambda$. In the general case of $n = \mathrm{poly}(\lambda)$, we assume that PRF and SKE are subexponentially secure. This means that there exists $0 < \delta < 1$ such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$. Similarly to the case of mi-prFE (See Remark 34 for further discussion), we set Λ so that it satisfies $2^{\Lambda^\delta} \geq \kappa^{\omega(1)}$. An example choice would be to take $\Lambda \coloneqq (n^2\lambda)^{1/\delta}$.

[14]Note that one can compute $\mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}$ in the Setup algorithm and output it as a part of public key rather than the secret key as it does not require the knowledge of f. We output this here for notational convinience.

Correctness. We prove the correctness of our scheme using the following theorem.

Theorem 6.29. *Assume* PE *and* mi-prFE *schemes are correct, and* PRF *is secure. Then the above construction of* miPE *scheme is correct.*

Proof. From the correctness of the mi-prFE scheme and definition of function F, we have

$$\mathsf{mi\text{-}prFE.Dec}(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}, \mathrm{F}, \mathsf{mi\text{-}prFE.ct}_1, \ldots, \mathsf{mi\text{-}prFE.ct}_n) = \mathsf{PE.ct}$$

with probability 1, where

$$\mathsf{PE.ct} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))\right).$$

Next, using the security of PRF, with all but negl advantage, $\mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))$ is indistinguishable from $R \leftarrow \{0,1\}^{\mathsf{R_{len}}}$. Since

$$\mathsf{PE.Dec}(\mathsf{PE.mpk}, \mathsf{PE.sk}_f, f, \mathsf{PE.Enc}\,(\mathsf{PE.mpk}, (\mathbf{x}_1, \ldots, \mathbf{x}_n), (\mu_1, \ldots, \mu_n); R)) = (\mu_1, \ldots, \mu_n)$$

holds for randomly chosen R with all but negl probability if $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = 1$ by the correctness of PE scheme, the above holds even for $R = \mathsf{PRF}(\mathbf{r}_n, (\mathbf{r}_1, \ldots, \mathbf{r}_{(n-1)}))$ with all but negl probability. Hence, the correctness. ■

6.4.2 Security

Here, we prove the security of our scheme. The following theorem asserts the security of the scheme for the case of n being arbitrary polynomial.

Theorem 6.30. *Assume* mi-prFE *scheme is secure (as per Definition 6.10) with respect to* $\kappa = \lambda^{n^2 \log \lambda}$ *and the sampler class containing the sampler* $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$ *as defined in Eq. 6.34,* PE *scheme is sub-exponentially secure (Definition 6.5) and* PRF *is sub-exponentially secure. Then the construction of* miPE *is secure as per Definition 6.6.*

Proof. Suppose the adversary $\mathcal{A}$ outputs the following:

1. **Key Queries.** It issues q_0 number of functions $f_1, \ldots, f_{q_0}$ for key queries.

2. **Ciphertext Queries.** It issues q_i number of messages for ciphertext queries for slot i. We use $(\mathbf{x}_i^{j_i}, \mu_i^{j_i})$ to denote the j_i-th ciphertext query corresponding to the i-th slot, where $j_i \in [q_i]$ and $i \in [n]$.

3. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 6.6 we first define $\{\mathsf{Sim}_i\}_{i\in[n]}$ as follows.

$\mathsf{Sim}_i(\mathsf{msk}) \to \mathsf{ct}_i$ for $i \in [n]$. Parse $\mathsf{msk} = (\text{mi-prFE.msk}, \mathsf{PE.msk})$. Set $\mathsf{Sim}_i(\mathsf{msk}) = \text{mi-prFE.Sim}_i(\text{mi-prFE.msk})$, where $\{\text{mi-prFE.Sim}_i\}_{i\in[n]}$ is the simulator whose existence is guaranteed by the security of mi-prFE (See Definition 6.10).

Then, it suffices to show

$$\left(\begin{array}{r} \mathsf{aux}_{\mathcal{A}},\ \mathsf{mpk} = (\text{mi-prFE.mpk}, \mathsf{PE.mpk}), \\ \left\{\mathrm{F}[\mathsf{PE.mpk}], f_k, \mathsf{sk}_{f_k} = (\mathsf{PE.sk}_{f_k}, \text{mi-prFE.sk}_{\mathrm{F}})\right\}_{k\in[q_0]}, \\ \left\{\mathsf{ct}_i^{j_i} = \text{mi-prFE.ct}_i^{j_i}\right\}_{i\in[n], j_i\in[q_i]} \end{array}\right) \approx_c \left(\begin{array}{r} \mathsf{aux}_{\mathcal{A}},\ \mathsf{mpk} = (\text{mi-prFE.mpk}, \mathsf{PE.mpk}), \\ \left\{\mathrm{F}[\mathsf{PE.mpk}], f_k, \mathsf{sk}_{f_k} = (\mathsf{PE.sk}_{f_k}, \text{mi-prFE.sk}_{\mathrm{F}})\right\}_{k\in[q_0]}, \\ \left\{\delta_i^{j_i} \leftarrow \text{mi-prFE.Sim}_i(\mathsf{msk})\right\}_{i\in[n], j_i\in[q_i]} \end{array}\right) \tag{6.33}$$

where

$(\mathsf{aux}_{\mathcal{A}}, \{f_k\}_k, \{(\mathbf{x}_i^{j_i}, \mu_i^{j_i})\}_{i,j_i}) \leftarrow \mathcal{A}(1^\lambda)$,

$(\mathsf{mpk} = (\text{mi-prFE.mpk}, \mathsf{PE.mpk}),\ \mathsf{msk} = (\text{mi-prFE.msk}, \mathsf{PE.msk})) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, \mathsf{prm})$,

$\text{mi-prFE.sk}_{\mathrm{F}} \leftarrow \text{mi-prFE.KeyGen}(\text{mi-prFE.msk}, \mathrm{F}[\mathsf{PE.mpk}])$,

$\mathsf{PE.sk}_{f_k} \leftarrow \mathsf{PE.KeyGen}(\mathsf{PE.msk}, f_k)$ for $k \in [q_0]$,

$\text{mi-prFE.ct}_i^{j_i} \leftarrow \text{mi-prFE.Enc}_i(\text{mi-prFE.msk}, (\mathbf{x}_i^{j_i}, \mu_i^{j_i}, \mathbf{r}_i^{j_i}))$, $\mathbf{r}_i^{j_i} \leftarrow \{0,1\}^\Lambda$ for $i \in [n], j_i \in [q_i]$.

We invoke the security of mi-prFE with sampler $\mathsf{Samp}_{\mathsf{mi\text{-}prFE}}$ that outputs

$$\begin{pmatrix} \text{Function:} & \mathrm{F}[\mathsf{PE.mpk}], \\ \text{Inputs:} & \left\{\left((\mathbf{x}_1^{j_1}, \mu_1^{j_1}, \mathbf{r}_1^{j_1}), \ldots, (\mathbf{x}_n^{j_n}, \mu_n^{j_n}, \mathbf{r}_n^{j_n})\right)\right\}_{i\in[n], j_i\in[q_i]}, \\ \text{Auxiliary Information:} & \mathsf{aux} = \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{PE.mpk}, \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}\right) \end{pmatrix} \tag{6.34}$$

From the security guarantee of the mi-prFE scheme with sampler $\mathsf{Samp}_{\mathsf{mi\text{-}prFE}}$ and simulator $\{\mathsf{mi\text{-}prFE.Sim}_i\}_{i\in[n]}$, we have that

$$\begin{pmatrix} \mathsf{aux},\ \mathsf{mi\text{-}prFE.mpk},\ \mathrm{F}[\mathsf{PE.mpk}], \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}, \\ \{\mathsf{mi\text{-}prFE.ct}_i^{j_i} \leftarrow \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{mi\text{-}prFE.msk}, (\mathbf{x}_i^{j_i}, \mu_i^{j_i}, \mathbf{r}_i^{j_i}))\}_{i\in[n], j_i\in[q_i]} \end{pmatrix}$$
$$\approx_c \begin{pmatrix} \mathsf{aux},\ \mathsf{mi\text{-}prFE.mpk},\ \mathrm{F}[\mathsf{PE.mpk}], \mathsf{mi\text{-}prFE.sk}_{\mathrm{F}}, \\ \{\mathsf{mi\text{-}prFE.ct}_i^{j_i} \leftarrow \mathsf{mi\text{-}prFE.Sim}_i(\mathsf{mi\text{-}prFE.msk})\}_{i\in[n], j_i\in[q_i]} \end{pmatrix}$$

$$\text{if} \quad \left(1^\kappa,\ \mathsf{aux},\ \{\mathrm{F}[\mathsf{PE.mpk}], \mathsf{PE.ct}_{j_1,\ldots,j_n}\}_{i\in[n], j_i\in[q_i]}\right)$$
$$\approx_c \left(1^\kappa,\ \mathsf{aux},\ \{\mathrm{F}[\mathsf{PE.mpk}], \mathsf{PE.}\Delta_{j_1,\ldots,j_n}\}_{i\in[n], j_i\in[q_i]}\right)$$

where $\mathsf{PE.}\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for $i \in [n], j_i \in [q_i]$ and

$$\begin{aligned} \mathsf{PE.ct}_{j_1,\ldots,j_n} =& \mathrm{F}[\mathsf{PE.mpk}]\left(\left(\mathbf{x}_1^{j_1}, \mu_1^{j_1}, \mathbf{r}_1^{j_1}\right), \ldots, \left(\mathbf{x}_{(n-1)}^{j_{(n-1)}}, \mu_{(n-1)}^{j_{(n-1)}}, \mathbf{r}_{(n-1)}^{j_{(n-1)}}\right), \left(\mathbf{x}_n^{j_n}, \mu_n^{j_n}, \mathbf{r}_n^{j_n}\right)\right) \\ =& \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}); \mathsf{PRF}\left(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{(n-1)}^{j_{(n-1)}})\right)\right) \end{aligned}$$

and $\mathsf{PE.}\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for $i \in [n], j_i \in [q_i]$.

Thus to prove Equation (6.33), it suffices to show

$$\left(\begin{array}{c} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathrm{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} = \mathsf{PE.Enc}\left(\begin{array}{c}\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_n^{j_n}), (\mu_1^{j_1}, \ldots, \mu_n^{j_n}); \\ \mathsf{PRF}(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1}, \ldots, \mathbf{r}_{(n-1)}^{j_{(n-1)}}))\end{array}\right)\right\}_{i\in[n], j_i\in[q_i]} \end{array}\right) \tag{6.35}$$

$$\approx_c \left(\begin{array}{c} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE}.\Delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}\right\}_{i\in[n], j_i\in[q_i]} \end{array} \right)$$

We prove the above via the following sequence of hybrids.

Hyb_1. This is Equation (6.35).

$$\left(\begin{array}{c} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} = \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n}), (\mu_1^{j_1},\ldots,\mu_n^{j_n}); \mathsf{PRF}(\mathbf{r}_n^{j_n}, (\mathbf{r}_1^{j_1},\ldots,\mathbf{r}_{(n-1)}^{j_{(n-1)}}))\right)\right\}_{i\in[n], j_i\in[q_i]} \end{array} \right)$$

Hyb_2. This hybrid is the same as the previous one except that we replace $\mathsf{PRF}(\mathbf{r}_n^{j_n}, \cdot)$ with the real random function $\mathsf{R}^{j_n}(\cdot)$ for each $j_n \in [q_n]$. Since $\mathbf{r}_n^{j_n}$ is not used anywhere else, we can use the subexponential security of PRF to conclude that this hybrid is computationally indistinguishable from the previous one. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary cannot distinguish this hybrid from the previous one with advantage more than $\mathsf{negl}(\kappa)$.

Hyb_4. This hybrid is the same as the previous one except that we output a failure symbol when there exist $(j_1,\ldots,j_{n-1}) \neq (j'_1,\ldots,j'_{n-1})$ such that $(\mathbf{r}_1^{j_1},\ldots,\mathbf{r}_{n-1}^{j_{n-1}}) = (\mathbf{r}_1^{j'_1},\ldots,\mathbf{r}_{n-1}^{j'_{n-1}})$. We show that the probability of this happening is negligible in κ. To prove this, it suffices to show that there are no $i \in [n]$, $j, j' \in [q_i]$ satisfying $j \neq j'$ and $\mathbf{r}_i^j = \mathbf{r}_i^{j'}$. The probability of this happening can be bounded by $(q_1^2 + \cdots q_{n-1}^2)/2^{2\Lambda}$ by taking the union bound with respect to all the combinations of i, j, j'. By our choice that $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, this probability is bounded by $\mathsf{negl}(\kappa)$.

Hyb_5. This hybrid is the same as the previous one except that we compute

$$\mathsf{PE.ct}_{j_1,\ldots,j_n} \leftarrow \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n}), (\mu_1^{j_1},\ldots,\mu_n^{j_n})\right).$$

Namely, we use fresh randomness for each encryption instead of deriving the

randomness from $\mathsf{R}^{j_n}(\cdot)$. This change is only conceptual.

In this hybrid, the view of the adversary is

$$\left(\begin{array}{c} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE.ct}_{j_1,\ldots,j_n} \leftarrow \mathsf{PE.Enc}\left(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n}), (\mu_1^{j_1},\ldots,\mu_n^{j_n})\right)\right\}_{i\in[n],j_i\in[q_i]} \end{array}\right)$$

Hyb_6. This hybrid is the same as the previous one except that we use sub-exponential security of underlying PE scheme to replace $\mathsf{PE.Enc}(\mathsf{PE.mpk}, (\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n}), (\mu_1^{j_1},\ldots,\mu_n^{j_n}))$ with $\mathsf{PE}.\delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}$ for all $i \in [n]$ and $j_i \in [q_i]$. By admissibility of $\mathcal{A}$ in the miPE security, we have $f(\mathbf{x}_1^{j_1},\ldots,\mathbf{x}_n^{j_n}) = 0$ which satisfies the admissibility of the single input PE security game. We therefore replace each of $\mathsf{PE.ct}_{j_1,\ldots,j_n}$ with a random string by the PE security. By our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$ and subexponential security of PE, the distinguishing advantage of the adversary when we replace single ciphertext is $\mathsf{negl}(\kappa)$. Since there are $q_1 \cdots q_n = \mathrm{poly}(\kappa)$ ciphertexts, we can conclude that the distinguishing advantage between this hybrid from the previous one is at most $\mathrm{poly}(\kappa)\mathsf{negl}(\kappa) = \mathsf{negl}(\kappa)$.

In this hybrid, the view of the adversary is

$$\left(\begin{array}{c} 1^\kappa,\ \mathsf{aux}_{\mathcal{A}},\ \mathsf{F}[\mathsf{PE.mpk}],\ \mathsf{PE.mpk},\ \{f_k, \mathsf{PE.sk}_{f_k}\}_{k\in[q_0]}, \\ \left\{\mathsf{PE}.\delta_{j_1,\ldots,j_n} \leftarrow \mathcal{CT}_{\mathsf{PE}}\right\}_{i\in[n],j_i\in[q_i]} \end{array}\right)$$

which is the distribution on RHS in Equation (6.35).

Hence, the proof. ∎

Constant arity case. In the special case of n being a constant, we can base the security of the scheme on weaker security requirements for the underlying ingredients. Concretely, we have the following theorem.

Theorem 6.31. *Assume* $\mathsf{mi\text{-}prFE}$ *scheme is secure (as per Definition 6.10) with respect to* $\kappa = \lambda^n$ *and the sampler class containing the sampler* $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$ *as defined in*

Eq. 6.34, PE *scheme is secure (Definition 6.5) and* PRF *is secure. Then the construction of* miPE *is secure as per Definition 6.6.*

The proof of the above theorem is exactly the same as Theorem 6.30 except that here $\kappa = \lambda^n$. Since $\kappa = \text{poly}(\lambda)$, we do not need subexponential security for PRF and PE. In addition, since mi-prFE for constant arity with $\kappa = \lambda^n$ can be constructed from (plain) evasive LWE, which is weaker assumption than non-uniform κ-version of it that is necessary for the general case.

We encapsulate the results of this section using the following theorems.

Theorem 6.32 (miPE for poly arity)**.** *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 2.5), there exists a* miPE *scheme for arity $n = \text{poly}(\lambda)$, supporting functions of bounded polynomial depth* $\mathsf{dep} = \mathsf{dep}(\lambda)$*, and satisfying security as per Definition 6.6.*

Theorem 6.33 (miPE for constant arity)**.** *Assuming evasive LWE (Assumption 2.6) and LWE (Assumption 2.5), there exists a* mi-prFE *scheme for arity $n = O(1)$, supporting functions of bounded polynomial depth $d = d(\lambda)$, and satisfying security as per Definition 6.6.*

6.5 INDISTINGUISHABILITY OBFUSCATION FOR PSEUDORANDOM FUNCTIONALITIES

6.5.1 Definition

In this section we give the definitions for indistinguishability obfuscation for pseudorandom functionalities (prIO) for circuits.

Syntax. An indistinguishability obfuscator for pseudorandom functionalities consists of the following algorithms.

$i\mathcal{O}(1^\lambda, C) \to \tilde{C}$**.** The obfuscation algorithm takes as input the security parameter λ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$ with arbitrary n and m. It outputs an obfuscated circuit $\tilde{C}$.

$\mathsf{Eval}(\tilde{C}, x) \to y$. The evaluation algorithm takes as input an obfuscated circuit $\tilde{C}$ and an input $x \in \{0,1\}^n$. It outputs y.

A uniform PPT machine $i\mathcal{O}$ is an indistinguishability obfuscator for pseudorandom functionalities w.r.t parameter $\kappa = \kappa(\lambda)$ if it satisfies the following properties.

Definition 6.12 (Polynomial Slowdown)**.** For all security parameters $\lambda \in \mathbb{N}$, for any circuit C and every input x, the evaluation time of $i\mathcal{O}(1^\lambda, C)$ on x is at most polynomially slower than the run time of the circuit C on x.

Definition 6.13 (Correctness)**.** For all security parameters $\lambda \in \mathbb{N}$, for all integers n, m, all circuits $C : \{0,1\}^n \to \{0,1\}^m$, and all input $x \in \{0,1\}^n$, we have that:

$$\mathsf{Pr}\left[C' \leftarrow i\mathcal{O}(1^\lambda, C) : C'(x) = C(x)\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator $i\mathcal{O}$.

Definition 6.14 (Indistinguishability for Pseudorandom Functionality)**.** For the security parameter $\lambda = \lambda(\lambda)$, let Samp be a PPT algorithm that on input 1^λ, outputs

$$(C_0, C_1, \ \mathsf{aux} \in \{0,1\}^*)$$

where $C_0 : \{0,1\}^n \to \{0,1\}^m$ and $C_1 : \{0,1\}^n \to \{0,1\}^m$ have the same description size. We say that a prIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\text{If } \left(1^\kappa, \{C_0(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{\Delta_x\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \approx_c \left(1^\kappa, \{C_1(x)\}_{x\in\{0,1\}^n}, \mathsf{aux}\right) \tag{6.36}$$

$$\text{then } \left(i\mathcal{O}(1^\lambda, C_0), \mathsf{aux}\right) \approx_c \left(i\mathcal{O}(1^\lambda, C_1), \mathsf{aux}\right) \tag{6.37}$$

where $\Delta_x \leftarrow \{0,1\}^m$ and $\kappa \geq 2^n$.

Remark 37. Note that 1^κ in the precondition is introduced for the purpose of padding, allowing the distinguisher for the distributions to run in time polynomial in κ. The reason why we require $\kappa \geq 2^n$ is that the input length to the distinguisher is polynomial in 2^n anyway and in order for the padding to make sense, κ should satisfy this condition.

Remark 38. It is shown in [19, 65] that there is no prIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of prIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications.

6.5.2 Construction

Our construction follows the blueprint of the multi-input FE to iO conversion by Ananth and Jain [29]. To obfuscate a circuit with input domain $\{0,1\}^n$, we generate a mi-prFE instance for arity $n+1$. We then let C be encrypted in position $n+1$, and the two inputs 0 and 1 be encrypted in position i for $i \in [n]$. The $2n+1$ ciphertexts together with a secret key for the universal circuit and the public parameters would form the iO.

Building Blocks. We use a $(n+1)$-input prFE scheme $\mathsf{mi\text{-}prFE} = \mathsf{mi\text{-}prFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Enc}_i\}_{i\in[n+1]}, \mathsf{Dec})$ for the circuit class with fixed input length, bounded depth, and binary output. We require mi-prFE to satisfy κ-security defined as per Definition 6.10. We can instantiate the scheme by our construction in Section 6.3.2 with $\kappa = \lambda^{n^2 \log \lambda}$.

Next, we describe the construction of prIO for all circuits.

$iO(1^\lambda, C)$**.** Given as input the security parameter 1^λ and a circuit $C : \{0,1\}^n \to \{0,1\}^m$ for arbitrary input length n, output length m, and description size L, do the following:

- Run $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{mi\text{-}prFE.Setup}(1^\lambda, 1^{n+1}, \mathsf{prm})$, where prm specifies message length L and the maximum depth d of the circuits supported by the mi-prFE instance. We set d to be the depth of the universal circuit U that, upon input an n-ary circuit C and vector $\mathbf{x} \in \{0,1\}^n$, outputs $U(C, \mathbf{x}) = C(\mathbf{x})$.

– Compute $\mathsf{ct}_{n+1} \leftarrow \mathsf{mi\text{-}prFE.Enc}_{n+1}(\mathsf{msk}, C)$.

– For $i \in [n]$ and $b \in \{0,1\}$, compute $\mathsf{ct}_{i,b} = \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{msk}, b)$.

– Compute $\mathsf{sk}_U \leftarrow \mathsf{mi\text{-}prFE.KeyGen}(\mathsf{msk}, U)$, where U is defined in the first item above.

– Output $\tilde{C} = \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{n+1}, \mathsf{sk}_U, \mathsf{mpk} \right)$

$\mathsf{Eval}(\tilde{C}, \mathbf{x})$**.** Given as input an obfuscated circuit $\tilde{C}$ and an input $\mathbf{x} \in \{0,1\}^n$, do the following:

1. Parse $\tilde{C} = \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{n+1}, \mathsf{sk}_U, \mathsf{mpk} \right)$.
2. Output $\mathsf{mi\text{-}prFE.Dec}(\mathsf{mpk}, \mathsf{sk}_U, U, \mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_n}, \mathsf{ct}_{n+1})$.

Remark 39. We note that the input size of mi-prFE scheme varies for slot 0, where we encrypt C, and slot i, where we encrypt a bit b, for $i \in [n]$. To make the input size consistent throughout the slots, we can pad the bit b with $\mathbf{0}$ (say) such that $|b\mathbf{0}| = |C|$ and give a mi-prFE key for a circuit U which on input $(C, b_1\mathbf{0}, \ldots, b_n\mathbf{0})$, where $b_i \in \{0,1\}$, simply discards the padding and outputs $C(b_1, \ldots, b_n)$.

Correctness. The correctness of the scheme follows in a straightforward manner from the correctness of the underlying mi-prFE scheme and the definition of the universal circuit U.

6.5.3 Security

Theorem 6.34. *Let $\mathcal{SC}_{\mathsf{prIO}}$ be a sampler class for* prIO*. Suppose* mi-prFE *scheme is secure (Definition 6.10) for $\kappa = \kappa(\lambda)$ w.r.t sampler class that contains all $\mathrm{Samp}_{\mathsf{mi\text{-}prFE},0}$ and $\mathrm{Samp}_{\mathsf{mi\text{-}prFE},1}$, induced by $\mathrm{Samp}_{\mathsf{prIO}} \in \mathcal{SC}_{\mathsf{prIO}}$, as defined in Equation (6.41). Then the* prIO *scheme satisfies security as defined in Definition 6.14 with $\kappa = \kappa(\lambda)$.*

Proof. Consider a sampler $\mathrm{Samp}_{\mathsf{prIO}}$ that generates the following:

1. **Obfuscation Query.** It issues $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$ with the same size L as an obfuscation query.

2. **Auxiliary Information.** It outputs the auxiliary information $\mathsf{aux}_{\mathcal{A}}$.

To prove the security as per Definition 6.14, we show that

$$\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \tag{6.38}$$

$$\text{if } \left(1^\kappa, \{C_0(x_i)\}_{\forall\, x_i\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(1^\kappa, \{\Delta_i \leftarrow \mathcal{Y}_\lambda\}_i, \mathsf{aux}_{\mathcal{A}}\right) \approx_c \left(1^\kappa, \{C_1(x_i)\}_{\forall\, x_i\in\mathcal{X}_\lambda}, \mathsf{aux}_{\mathcal{A}}\right) \tag{6.39}$$

where

$$\begin{aligned}
&(C_0, C_1, \mathsf{aux}_{\mathcal{A}}) \leftarrow \mathrm{Samp}_{\mathsf{prIO}}(1^\lambda),\\
&(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{mi\text{-}prFE.Setup}(1^\lambda, 1^{n+1}, \mathsf{prm}),\\
&\mathsf{sk}_U \leftarrow \mathsf{mi\text{-}prFE.KeyGen}(\mathsf{msk}, U),\\
&\mathsf{ct}^0_{n+1} \leftarrow \mathsf{mi\text{-}prFE.Enc}_{n+1}(\mathsf{msk}, C_0), \mathsf{ct}^1_{n+1} \leftarrow \mathsf{mi\text{-}prFE.Enc}_{n+1}(\mathsf{msk}, C_1),\\
&\mathsf{ct}_{i,b} = \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{msk}, b), \text{ for } i \in [n], b \in \{0,1\}.
\end{aligned}$$

From the right hand indistinguishability of Equation (6.39), we have that

$$\begin{aligned}
&\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right)\\
\approx_c &\left(\{\gamma_{i,b} \leftarrow \mathsf{mi\text{-}prFE.Sim}_i(\mathsf{msk})\}_{i\in[n],b\in\{0,1\}}, \gamma_{n+1} \leftarrow \mathsf{mi\text{-}prFE.Sim}_{n+1}(\mathsf{msk}), \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right)
\end{aligned} \tag{6.40}$$

using the mi-prFE security with simulator $\{\mathsf{mi\text{-}prFE.Sim}_i\}_{i\in[n+1]}$ and sampler

$\mathsf{Samp}_{\mathsf{mi\text{-}prFE},0}$ that outputs

$$\begin{pmatrix} \text{Function:} & \text{Universal Circuit } U \\ \text{Inputs:} & \{x_1^{j_1} = j_1, \ldots, x_n^{j_n} = j_n\}_{j_1 \in \{0,1\}, \ldots, j_n \in \{0,1\}}, x_{n+1} = C_0 \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}} \end{pmatrix} \tag{6.41}$$

To see this we note that the pseudorandomness of $U(x_1 \in \{0,1\} \ldots, x_n \in \{0,1\}, x_{n+1} = C_{n+1} \in \{0,1\}^L) = C_0(x_1, \ldots, x_n)$ is implied from Equation (6.39).

Similarly, from the right hand indistinguishability of Equation (6.39), we have

$$\begin{aligned} &\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \\ \approx_c &\left(\{\gamma_{i,b} \leftarrow \mathsf{mi\text{-}prFE.Sim}_i(\mathsf{msk})\}_{i\in[n],b\in\{0,1\}}, \gamma_{n+1} \leftarrow \mathsf{mi\text{-}prFE.Sim}_{n+1}(\mathsf{msk}), \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \end{aligned} \tag{6.42}$$

using the $\mathsf{mi\text{-}prFE}$ security with simulator $\{\mathsf{mi\text{-}prFE.Sim}_i\}_{i\in[0,n]}$ and sampler $\mathsf{Samp}_{\mathsf{mi\text{-}prFE},1}$ whose output is same as that of Samp_0 except that $x_{n+1} = C_1$.

From Equation (6.40) and Equation (6.42), we have

$$\begin{aligned} &\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^0_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \\ \approx_c &\left(\{\mathsf{ct}_{i,b}\}_{i\in[n],b\in\{0,1\}}, \mathsf{ct}^1_{n+1}, \mathsf{sk}_U, \mathsf{mpk}, \mathsf{aux}_{\mathcal{A}}\right) \end{aligned}$$

hence the proof. ■

The following theorem holds for the specific class of samplers we described in the proof.

Theorem 6.35. *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 2.5), there exists a prIO scheme for circuits with input domain $\{0,1\}^n$ satisfying security as per Definition 6.14.*

6.6 POLYNOMIAL DOMAIN IO FOR PSEUDORANDOM FUNCTIONALITIES

In this section, we define and construct indistinguishability obfuscation for pseudorandom functionalities with polynomial size domain (pPRIO). The advantage of considering this restricted variant is that we can base the security of the construction on (plain) evasive LWE, rather than non-uniform κ-variant of it. Here, we introduce online-offline property and reusable security and provide a construction satisfying these properties. The reason why we introduce these properties is that they seem to be useful for some applications. Indeed, in our companion paper [12], we use pPRIO with these properties to construct ABE with optimized parameter size. We consider two variants of pPRIO: pPRIO with fixed input domain and pPRIO with flexible domain. As the name suggests, the latter is more flexible and desirable. To obtain the latter, we first construct the former from mi-prFE in Section 6.6.2 and then convert it into the latter in Section 6.6.3.

6.6.1 Definition

Definition 6.15 (Syntax)**.** A pPRIO scheme consists of the following algorithms.

$\mathsf{Obf}(1^\lambda, C) \to \mathsf{Obf}$. The obfuscation algorithm takes as input the security parameter λ and a circuit $C : [N] \to [M]$ with $\mathsf{size}(C) \leq L$ for some arbitrary polynomial $L = L(\lambda)$. It outputs an obfuscation of the circuit Obf.

We consider a definition where the Obf algorithm can be decomposed into the following two phases.

$\mathsf{ObfOff}(1^\lambda, 1^L) \to (\mathsf{obf}_{\mathsf{off}}, \mathsf{st})$. The offline obfuscation algorithm takes as input the security parameter λ and the circuit size bound L. It outputs $\mathsf{obf}_{\mathsf{off}}$ and st.

$\mathsf{ObfOn}(\mathsf{st}, C) \to \mathsf{obf}_{\mathsf{on}}$. The online obfuscation algorithm takes as input the security parameter st and the circuit C and outputs $\mathsf{obf}_{\mathsf{on}}$.

The final output of Obf is $\mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}})$.

$\mathsf{Eval}(\mathsf{Obf}, x) \to y$. The evaluation algorithm takes as input an obfuscated circuit Obf and an input $x \in [N]$. It outputs $y \in [M]$.

We consider the following syntax variants of a pPRIO scheme. We consider fixed input domain pPRIO and flexible input domain pPRIO. In the former, we need that ObfOff algorithm should know the input domain $[N]$ of C that is going to be input to ObfOn while in the latter, we do not.

Definition 6.16 (Fixed Input Domain pPRIO)**.** A *fixed input domain* pPRIO scheme has syntax as in Definition 6.15 except that ObfOff takes N (in binary) as an additional input.

Definition 6.17 (Flexible Input Domain pPRIO)**.** A *flexible input domain* pPRIO scheme has syntax exactly as in Definition 6.15, i.e., a variant without inputting N into ObfOff.

Next, we define the properties of a pPRIO scheme.

Definition 6.18 (Correctness for Flexible Input Domain Case)**.** For all security parameters $\lambda \in \mathbb{N}$, for any $C : [N] \to [M]$, $L = L(\lambda)$ such that $\mathsf{size}(C) \leq L$ and every input $x \in [N]$, we have that:

$$\Pr\left[\mathsf{Eval}(\mathsf{Obf}, x) = C(x) \;\middle|\; \begin{array}{c} \mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}),\ (\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L), \\ \mathsf{obf}_{\mathsf{on}} \leftarrow \mathsf{ObfOn}(\mathsf{st}, C) \end{array}\right] = 1$$

where the probability is taken over the coin-tosses of the obfuscator Obf. The correctness for the fixed input domain case is defined as above, except that $\mathsf{ObfOff}(1^\lambda, 1^L)$ is replaced with $\mathsf{ObfOff}(1^\lambda, 1^L, N)$.

We introduce a bit unusual efficiency requirement for pPRIO. Namely, we require that pPRIO can obfuscate circuits from exponentially large input domain efficiently. One may think that with this property, pPRIO is equivalent to prIO. However, for pPRIO, we require that it can securely obfuscate a circuit only when it has polynomial size domain and do not require any security guarantee when it has superpolynomial domain size. This relaxation leads to the construction with weaker assumption (i.e., uniform and non-κ variant of evasive LWE).

The following efficiency requirement is the consequence of inputting N into ObfOff in *binary* form and the requirement that ObfOff is a PPT algorithm, which implies that it runs in polynomial time in the input length. We explicitly require this for emphasizing the property.

Definition 6.19 (Efficiency for Fixed Input Domain)**.** We require that the running time of $\mathsf{ObfOff}(1^\lambda, 1^L, N)$ is bounded by $\mathrm{poly}(\lambda, L, \log N)$.

We introduce the security definition for pPRIO.

Definition 6.20 (Security for Flexible Input Domain)**.** Let Samp be a PPT algorithm that on input 1^λ, outputs

$$\left(1^{N_1+N_2+\cdots N_Q}, 1^L, \mathsf{aux}, C^1, \ldots, C^Q\right), \quad \text{where} \quad C^i : [N_i] \to [M_i], \ \ \mathsf{size}(C^i) \leq L$$

where we enforce Samp to output $1^{N_1+N_2+\cdots N_Q}$ to make sure that all N_i are bounded by $\mathrm{poly}(\lambda)$. We say that a flexible input domain pPRIO scheme is secure with respect to the sampler class $\mathcal{SC}$ if for every PPT sampler Samp $\in \mathcal{SC}$ the following holds.

$$\text{If } \left(\mathsf{aux}, 1^L, \{C^1(i)\}_{i\in[N_1]}, \ldots, \{C^Q(i)\}_{i\in[N_Q]}\right) \approx_c \left(\mathsf{aux}, 1^L, \{\Delta_i^1\}_{i\in[N_1]}, \ldots, \{\Delta_i^Q\}_{i\in[N_Q]}\right) \tag{6.43}$$

$$\text{then} \quad \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}^1, \ldots, \mathsf{obf}_{\mathsf{on}}^Q\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \delta^1 \leftarrow \mathcal{CT}^1 \ldots, \delta^Q \leftarrow \mathcal{CT}^Q\right), \tag{6.44}$$

where $\Delta_i^j \leftarrow [M_j]$ for $j \in [Q], i \in [N_j]$, $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{ObfOff}(1^\lambda, 1^L)$, $\mathsf{obf}_{\mathsf{on}}^j \leftarrow \mathsf{ObfOn}(\mathsf{st}, C^j)$ for $j \in [Q]$, and $\mathcal{CT}^j$ denotes the set of binary strings of the same length as the output of $\mathsf{obf}_{\mathsf{on}}(\mathsf{st}, C^j)$ algorithm.

Definition 6.21 (Security for Fixed Input Domain)**.** The security for flexible input domain pPRIO is defined exactly as in Definition 6.20, except that we have $N_1 = N_2 = \cdots = N_Q$.

Remark 40 (Comparison with Definition 6.14)**.** Here, we compare the above security definitions for pPRIO with that for prIO (Definition 6.14). On the one hand, the above security definition is weaker than Definition 6.14 in that the adversary is not allowed to submit a circuit with exponential size input domain. This restriction is captured by the

above definitions where we enforce the adversary to output the size of the input domain for each circuit in *unary*.

On the other hand, the above security definitions are stronger than Definition 6.14 in two folds. First, it requires the online part of the obfuscation $\mathsf{obf}_{\mathsf{on}}$ to be *pseudorandom*, whereas Definition 6.14 only requires the obfuscation of the circuits are *computationally indistinguishable* (when their truth tables are pseudorandom). Secondly, the above security definition considers a security game where the same internal state st is reused for obfuscating *multiple* circuits, whereas only a single circuit is obfuscated in Definition 6.14. The above security definition may look odd, but pPRIO with this security notion has proven useful in our companion paper [12] to obtain optimal parameters for ABE schemes.

Remark 41. Similar to Remark 38, there is no pPRIO scheme satisfying the above security for all general samplers. Therefore, when we use the security of pPRIO, we invoke the security *with respect to a specific sampler class* that is induced by the respective applications.

6.6.2 Construction for Fixed Input Domain

In this section, we provide a construction of fixed input domain pPRIO scheme $\mathsf{pPRIO} = (\mathsf{Obf} = (\mathsf{ObfOff}, \mathsf{ObfOn}), \mathsf{Eval})$ for circuit class $C = \{C : [\lambda^c] \to \{0,1\}\}$, for some constant $c \in \mathbb{N}$, from $\mathsf{mi\text{-}prFE}$ for $c+1$ arity. The construction is extended to be able to handle flexible input domain in Section 6.6.3.

Building Blocks. We use the following ingredient for our construction.

1. A mi-prFE scheme $\mathsf{mi\text{-}prFE} = \mathsf{mi\text{-}prFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \{\mathsf{Enc}_i\}_{i\in[c+1]}, \mathsf{Dec})$ for the circuit class with fixed input length $L = |C|$, bounded depth $d = d(\lambda)$, and binary output. We require $\mathsf{mi\text{-}prFE}$ to satisfy the pseudorandomness of the last slot ciphertext as per Definition 6.11. We can instantiate the scheme by our construction in Section 6.3.2.

Next, we describe our construction.

$\mathsf{ObfOff}(1^\lambda, 1^L, N = \lambda^c)$. The offline phase of obfuscation algorithm does the following.

– Run $(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.msk}) \leftarrow \mathsf{mi\text{-}prFE.Setup}(1^\lambda, 1^{c+1}, \mathsf{prm})$, where prm specifies the maximum size L of the circuit that is going to input to ObfOn and the maximum depth d of the circuit class supported by the $\mathsf{mi\text{-}prFE}$ instance. We set d to be the depth of the universal circuit U that, upon input a c-ary circuit C and vector $\mathbf{x} \in [\lambda]^c$, outputs $U(C, \mathbf{x}) = C(\mathbf{x})$, where $\mathbf{x}$ is interpreted as an integer in $[\lambda^c]$ by some efficient bijective mapping between $[\lambda^c]$ and $[\lambda]^c$.

– For $i \in [c]$ and $j \in [\lambda]$, compute $\mathsf{mi\text{-}prFE.ct}_{i,j} = \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{mi\text{-}prFE.msk}, j)$.[15]

– Compute $\mathsf{mi\text{-}prFE.sk}_U = \mathsf{mi\text{-}prFE.KeyGen}(\mathsf{mi\text{-}prFE.msk}, U)$, where U is supported by the $\mathsf{mi\text{-}prFE}$ instance because of our choice of prm.

– Output $\mathsf{obf}_{\mathsf{off}} \coloneqq \left(\mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U\right)$ and $\mathsf{st} = \mathsf{mi\text{-}prFE.msk}$.

$\mathsf{ObfOn}(\mathsf{st}, C)$. The online phase of obfuscation algorithm does the following.

– Run $\mathsf{mi\text{-}prFE.ct}_{c+1} \leftarrow \mathsf{mi\text{-}prFE.Enc}_{c+1}(\mathsf{mi\text{-}prFE.msk}, C)$.

– Output $\mathsf{obf}_{\mathsf{on}} \coloneqq \mathsf{mi\text{-}prFE.ct}_{c+1}$.

$\mathsf{Eval}(\mathsf{Obf}, \mathbf{x})$. The evaluation algorithm does the following.

1. Parse $\mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}})$ where $\mathsf{obf}_{\mathsf{off}} = (\mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U)$ and $\mathsf{obf}_{\mathsf{on}} = \mathsf{mi\text{-}prFE.ct}_{c+1}$.

2. Output $\mathsf{mi\text{-}prFE.Dec}(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.sk}_U, U, \mathsf{mi\text{-}prFE.ct}_{1,x_1}, \ldots, \mathsf{mi\text{-}prFE.ct}_{c,x_c}, \mathsf{mi\text{-}prFE.ct}_{c+1})$, where $\mathbf{x} \in [\lambda^c]$ is mapped to $(x_1, \ldots, x_c) \in [\lambda]^c$ by the bijective mapping between $[\lambda^c]$ and $[\lambda]^c$.

Correctness. The correctness of the scheme follows in a straightforward manner from the correctness of the underlying $\mathsf{mi\text{-}prFE}$ scheme and the definition of the universal circuit U.

[15] Using Remark 39, it is safe to use this instance of $\mathsf{mi\text{-}prFE}$ to encrypt messages with smaller length than L.

Efficiency. The scheme satisfies

$$|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(c, L, \lambda) = \mathrm{poly}(\lambda, L), |\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(c, L, \lambda) = \mathrm{poly}(\lambda, L),$$

where $(\mathsf{Obf}_{\mathsf{off}}, \mathsf{Obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit $C : [N] \to [M]$ whose size is bounded by $L = L(\lambda)$.

Security. We prove the security of the fixed input pPRIO using the following theorem.

Theorem 6.36. *Let $\mathcal{SC}_{\mathsf{pPRIO}}$ be a sampler class for* pPRIO. *Suppose* mi-prFE *scheme is secure (Definition 6.10) with $\kappa = \lambda^{c+1}$ and sampler class that contains all $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$ induced by $\mathrm{Samp}_{\mathsf{prIO}} \in \mathcal{SC}_{\mathsf{prIO}}$, as defined in Equation (6.47). Then the* pPRIO *scheme satisfies security as defined in Definition 6.21.*

Proof. Consider a sampler Samp that generates

$$\left(1^{Q\cdot N}, 1^L, \mathsf{aux}_{\mathcal{A}}, \{C^k\}_{k\in[Q]}\right).$$

To prove the theorem, we show that

$$\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{obf}_{\mathsf{off}} = \left(\mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U\right), \left\{\mathsf{obf}^k_{\mathsf{on}} = \mathsf{mi\text{-}prFE.ct}^k_{c+1}\right\}_{k\in[Q]}\right)$$
$$\approx_c \left(\begin{array}{c}\mathsf{aux}_{\mathcal{A}}, \mathsf{obf}_{\mathsf{off}} = \left(\mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U\right), \\ \left\{\mathsf{obf}^k_{\mathsf{on}} \leftarrow \mathcal{CT}_{\mathsf{mi\text{-}prFE},c+1}\right\}_{k\in[Q]},\end{array}\right) \tag{6.45}$$

holds assuming

$$\left(\mathsf{aux}_{\mathcal{A}}, 1^L, \{C^1(i)\}_{i\in[N]}, \ldots, \{C^Q(i)\}_{i\in[N]}\right) \approx_c \left(\mathsf{aux}_{\mathcal{A}}, 1^L, \{\Delta_i^1\}_{i\in[N]}, \ldots, \{\Delta_i^Q\}_{i\in[N]}\right) \tag{6.46}$$

where $\mathcal{CT}_{\mathsf{mi\text{-}prFE},c+1}$ denotes the set of binary strings with the same length as $\mathsf{mi\text{-}prFE.ct}^k_{c+1}$ and

$$\left(1^{Q\cdot N}, 1^L, \mathsf{aux}_{\mathcal{A}}, \{C^k\}_{k\in[Q]}\right) \leftarrow \mathrm{Samp}(1^\lambda),$$
$$(\mathsf{mi\text{-}prFE.mpk}, \mathsf{mi\text{-}prFE.msk}) \leftarrow \mathsf{mi\text{-}prFE.Setup}(1^\lambda, 1^{c+1}, \mathsf{prm}),$$

$$\mathsf{mi\text{-}prFE.sk}_U \leftarrow \mathsf{mi\text{-}prFE.KeyGen}(\mathsf{mi\text{-}prFE.msk}, U),$$

$$\mathsf{mi\text{-}prFE.ct}_{i,j} \leftarrow \mathsf{mi\text{-}prFE.Enc}_i(\mathsf{mi\text{-}prFE.msk}, j) \text{ for } i \in [c], j \in [\lambda],$$

$$\Delta_1^k, \ldots, \Delta_N^k \leftarrow \{0,1\} \text{ for } k \in [Q],$$

$$\mathsf{mi\text{-}prFE.ct}_{c+1}^k \leftarrow \mathsf{mi\text{-}prFE.Enc}_{c+1}(\mathsf{mi\text{-}prFE.msk}, C^k) \text{ for } k \in [Q].$$

We invoke the security of $\mathsf{mi\text{-}prFE}$ with sampler $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$ that outputs

$$\begin{pmatrix} \text{Function:} & \text{Universal Circuit } U \\ \text{Inputs:} & \{x_1^{j_1} = j_1, \ldots, x_c^{j_c} = j_c, x_{c+1}^k = C^k\}_{j_1,\ldots,j_c \in [\lambda], k \in [Q]} \\ \text{Auxiliary Information:} & \mathsf{aux}_{\mathcal{A}} \end{pmatrix} \tag{6.47}$$

Using the $\mathsf{mi\text{-}prFE}$ security with sampler $\mathrm{Samp}_{\mathsf{mi\text{-}prFE}}$, we have that

$$\begin{aligned} &\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U, \left\{\mathsf{mi\text{-}prFE.ct}_{c+1}^k\right\}_{k\in[Q]}\right) \\ &\approx_c \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{mi\text{-}prFE.mpk}, \{\delta_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U, \left\{\delta_{c+1}^k\right\}_{k\in[Q]}\right) \end{aligned} \tag{6.48}$$

given

$$\left(\mathsf{aux}_{\mathcal{A}}, \left\{U(C^k, x_1^{j_1}, \ldots, x_c^{j_c})\right\}_{j_1,\ldots,j_c\in[\lambda],k\in[Q]}\right) \approx_c \left(\mathsf{aux}_{\mathcal{A}}, \{\Delta_i^k \leftarrow \{0,1\}\}_{i\in[N],k\in[Q]}\right) \tag{6.49}$$

where $\delta_{i,j} \leftarrow \mathsf{Sim}_i(\mathsf{mi\text{-}prFE.msk})$ for $i \in [c], j \in [\lambda]$ and $\delta_{c+1}^k \leftarrow \mathcal{CT}_{\mathsf{mi\text{-}prFE},c+1}$[16].

This implies

$$\begin{aligned} &\left(\mathsf{aux}_{\mathcal{A}}, \mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U, \left\{\mathsf{mi\text{-}prFE.ct}_{c+1}^k\right\}_{k\in[Q]}\right) \\ &\approx_c \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{mi\text{-}prFE.mpk}, \{\delta_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U, \left\{\delta_{c+1}^k\right\}_{k\in[Q]}\right) \\ &\approx_c \left(\mathsf{aux}_{\mathcal{A}}, \mathsf{mi\text{-}prFE.mpk}, \{\mathsf{mi\text{-}prFE.ct}_{i,j}\}_{i\in[c],j\in[\lambda]}, \mathsf{mi\text{-}prFE.sk}_U, \left\{\delta_{c+1}^k\right\}_{k\in[Q]}\right) \end{aligned} \tag{6.50}$$

given Equation (6.49), where the first indistingusihability follows from Equation (6.48)

[16] This follows from the fact that the simulator for the $c+1$-th slot outputs a random string as is required by Definition 6.11.

and the second also from Equation (6.48) by noting that replacing the last term with independently chosen random variable makes the distinguishing task of the distributions even harder. Equation (6.49) follows from Equation (6.46) and the definition of circuit U, hence the proof. ■

Instantiation. Instantiating the underlying mi-prFE scheme with constant arity $c + 1$ as in Theorem 6.28, we get the following theorem.

Theorem 6.37. *Assuming evasive LWE (Assumption 2.6) and LWE (Assumption 2.5), there exists a secure* pPRIO *scheme for fixed input domain supporting circuits of bounded size* $L = \text{poly}(\lambda)$ *with*

$$|\mathsf{Obf}_{\mathsf{off}}| = \text{poly}(L, \lambda), \quad |\mathsf{Obf}_{\mathsf{on}}| = \text{poly}(L, \lambda).$$

6.6.3 Construction for Flexible Input Domain

Here, we provide a construction of pPRIO for flexible input domain Flex.pPRIO = Flex.(ObfOff, ObfOn, Eval) for any circuits with binary output space from pPRIO for fixed input domain. The restriction that the output domain of the circuits being binary is removed in Section 6.6.4.

Building Blocks. Below, we list the ingredients for our construction.

1. A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ with key space, input space and output space as $\{0,1\}^\lambda$. The input to the PRF will be in the form of $\mathbf{x} \in [\lambda^c]$ for some constant $c \in \mathbb{N}$. Since we have $2^\lambda > \text{poly}(\lambda)$, the input space of the PRF is large enough to accommodate inputs $\mathbf{x} \in [\lambda^c]$ with an appropriate encoding. It is known that PRFs can be constructed from one-way functions.

2. A fixed input domain pPRIO scheme Fixed.(ObfOff, ObfOn, Eval) as constructed in Section 6.6.2 with the efficiency requirement defined in Definition 6.19 and the security defined as per Definition 6.20. The input domain of pPRIO should be $[\lambda^c]$ for some constant c.

$\mathsf{Flex.Obf}(1^\lambda, C)$. For a circuit C such that $\mathsf{size}(C) \le L$ for $L = L(\lambda)$, the obfuscation

algorithm works as following.

$\mathsf{Flex.ObfOff}(1^\lambda, 1^L)$. The offline phase of obfuscation algorithm does the following.

- Run $(\mathsf{Fixed.Obf}_{\mathsf{off},i}, \mathsf{Fixed.st}_i) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$.
- Output $\mathsf{obf}_{\mathsf{off}} := \{\mathsf{Fixed.Obf}_{\mathsf{off},i}\}_{i\in[\lambda]}$ and $\mathsf{st} = \{\mathsf{Fixed.st}_i\}_{i\in[\lambda]}$.

$\mathsf{Flex.ObfOn}(\mathsf{st}, C)$. The online phase of obfuscation algorithm takes as input a circuit $C : [N] \to \{0,1\}$ and parses the input as $\mathsf{st} = \{\mathsf{Fixed.st}_i\}_{i\in[\lambda]}$. It then does the following:

- Find $c \in \mathbb{N}$ such that $\lambda^{c-1} < N \leq \lambda^c$.
- Sample $\mathsf{sd} \leftarrow \{0,1\}^\lambda$
- Construct a circuit $C[\mathsf{sd}] : [\lambda^c] \to \{0,1\}$ that is defined as, on input $\mathbf{x} \in [\lambda^c]$, $C[\mathsf{sd}](\mathbf{x}) = \begin{cases} C(\mathbf{x}) & \text{if } \mathbf{x} \leq N \\ \mathsf{PRF}(\mathsf{sd}, \mathbf{x}) & \text{if } N < \mathbf{x} \leq \lambda^c \end{cases}$.
- Compute $\mathsf{Fixed.Obf}_{\mathsf{on},c} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_c, C[\mathsf{sd}])$.
- Output $\mathsf{obf}_{\mathsf{on}} := \mathsf{Fixed.Obf}_{\mathsf{on},c}$.

$\mathsf{Flex.Eval}(\mathsf{Obf}, \mathbf{x})$. The evaluation algorithm does the following.

- Parse $\mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) = (\{\mathsf{Fixed.Obf}_{\mathsf{off},i}\}_{i\in[\lambda]}, \mathsf{Fixed.Obf}_{\mathsf{on},c})$.
- Construct $\mathsf{Fixed.Obf} = (\mathsf{Fixed.Obf}_{\mathsf{off},c}, \mathsf{Fixed.Obf}_{\mathsf{on},c})$[17].
- Output $\mathsf{Fixed.Eval}(\mathsf{Fixed.Obf}, \mathbf{x})$.

Correctness. For $\mathsf{Obf} = (\mathsf{obf}_{\mathsf{off}}, \mathsf{obf}_{\mathsf{on}}) = (\{\mathsf{Fixed.Obf}_{\mathsf{off},i}\}_{i\in[\lambda]}, \mathsf{Fixed.Obf}_{\mathsf{on},c})$, we have $\mathsf{Fixed.Obf}_{\mathsf{off},c} \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^c)$ and $\mathsf{Fixed.Obf}_{\mathsf{on},c} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_c, C[\mathsf{sd}])$ where $C[\mathsf{sd}] : [\lambda^c] \to \{0,1\}$. From the correctness of the underlying $\mathsf{Fixed.pPRIO}$ scheme we have

[17] We assume that we can obtain c from the length of $\mathbf{x}$.

$\mathsf{Fixed.Eval}(\mathsf{Fixed.Obf}, \mathbf{x}) = C[\mathsf{sd}](\mathbf{x}) = C(\mathbf{x})$ for any input $\mathbf{x} \in [\lambda^c]$.

Efficiency. It is straightforward to see that the algorithms run in polynomial time in the input length and λ. The only non-trivial part is to bound the running time of $\mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$. The running time of this part can be bounded by $\mathrm{poly}(\lambda, L, \log \lambda^i) = \mathrm{poly}(\lambda, L)$ by the efficiency property of the underlying fixed input domain pPRIO as per Definition 6.19. In particular, we have

$$|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda)$$

where $(\mathsf{Obf}_{\mathsf{off}}, \mathsf{Obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit $C : [N] \to [M]$ whose size is bounded by $L = L(\lambda)$.

Security. We prove the security of the above construction. Before doing so, we prove the following useful lemma. The lemma essentially says that if a part of the auxiliary information is pseudorandom in the pre-condition distribution, then it is pseudorandom in the corresponding post-condition distribution where we apply pPRIO. Conceptually similar lemma is proven in Lemma 3.4 of [22] in the context of evasive LWE.

Lemma 6.38. *Let* $\mathsf{pPRIO} = (\mathsf{Fixed.ObfOff}, \mathsf{Fixed.ObfOn}, \mathsf{Fixed.Eval})$ *be a* pPRIO *scheme for fixed input domain and* Samp *be a PPT algorithm that takes as input* 1^λ *and outputs*

$$\left(1^{NQ}, 1^L, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2) \in \{0,1\}^* \times \mathcal{X}, \{C^j\}_{j\in[Q]}\right)$$

for some set $\mathcal{X}$*. Let us assume that*

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^L, \left\{C^j(i)\right\}_{i\in[N], j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^L, \left\{\Delta_i^j\right\}_{i\in[N], j\in[Q]}\right)$$

holds for $\mathbf{x} \leftarrow \mathcal{X}, \Delta_i^j \leftarrow \{0,1\}$ *and assume the security of* pPRIO *with respect to* Samp*. We then have*

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right), \tag{6.51}$$

where we have $(\mathsf{obf}_{\mathsf{off}}, \mathsf{st}) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, N)$, $\mathsf{obf}^j_{\mathsf{on}} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{st}, C^j)$ *and* $\delta^j \leftarrow \mathcal{CT}_{\mathsf{ObfOn}}$ *for* $j \in [Q]$, *where* $\mathcal{CT}_{\mathsf{ObfOn}}$ *is the set of binary strings with the same length as* $\mathsf{obf}^j_{\mathsf{on}}$.

Proof. From the assumption, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), 1^L, \left\{C^j(i)\right\}_{i\in[N], j\in[Q]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), 1^L, \left\{\Delta^j_i\right\}_{i\in[N], j\in[Q]}\right) \tag{6.52}$$

which implies $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L)$. This further implies

$$(\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{\Delta^j_i\}_{i\in[N], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L, \{\Delta^j_i\}_{i\in[N], j\in[Q]}) \tag{6.53}$$

since adding independently sampled random terms does not make the task of indistinguishing the distributions easier. Equation (6.52) and Equation (6.53) implies $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{C^j(i)\}_{i\in[N], j\in[Q]}) \approx_c (\mathsf{aux}_1, \mathsf{aux}_2, 1^L, \{\Delta^j_i\}_{i\in[N], j\in[Q]})$. Applying pPRIO security definition with respect to Samp, we get

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}^j_{\mathsf{on}}\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right).$$

Recall that we have $(\mathsf{aux}_1, \mathsf{aux}_2, 1^L) \approx_c (\mathsf{aux}_1, \mathbf{x}, 1^L)$. This further implies

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right), \tag{6.54}$$

since $\mathsf{obf}_{\mathsf{off}}$ can be sampled independently given 1^L, N and $\{\delta^j\}_{j\in[Q]}$.

From Section 6.6.3 and Equation (6.54) we deduce

$$\left(\mathsf{aux}_1, \mathsf{aux}_2, \mathsf{obf}_{\mathsf{off}}, \left\{\mathsf{obf}^j_{\mathsf{on}}\right\}_{j\in[Q]}\right) \approx_c \left(\mathsf{aux}_1, \mathbf{x}, \mathsf{obf}_{\mathsf{off}}, \left\{\delta^j\right\}_{j\in[Q]}\right)$$

and hence the lemma. ∎

Following theorem asserts the security of our construction.

Theorem 6.39. *Our* pPRIO *scheme for flexible input domain is secure.*

Proof. Let us consider a sampler Samp that outputs $\left(1^{\sum_{j\in[Q]} N_j}, 1^L, \mathsf{aux}, \{C^j\}_{j\in[Q]}\right)$. To prove the theorem, we show that

$$\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \mathsf{obf}^1_{\mathsf{on}}, \ldots, \mathsf{obf}^Q_{\mathsf{on}}\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \delta^1 \ldots, \delta^Q\right), \tag{6.55}$$

holds assuming

$$\left(\mathsf{aux}, \{C^1(i)\}_{i\in[N_1]}, \ldots, \{C^Q(i)\}_{i\in[N_Q]}\right) \approx_c \left(\mathsf{aux}, \{\Delta^1_i\}_{i\in[N_1]}, \ldots, \{\Delta^Q_i\}_{i\in[N_Q]}\right)$$

where

$\mathsf{obf}_{\mathsf{off}} = \{\mathsf{Fixed.Obf}_{\mathsf{off},i}\}_{i\in[\lambda]}$, $(\mathsf{Fixed.Obf}_{\mathsf{off},i}, \mathsf{Fixed.st}_i) \leftarrow \mathsf{Fixed.ObfOff}(1^\lambda, 1^L, \lambda^i)$ for $i \in [\lambda]$,

$\mathsf{obf}^1_{\mathsf{on}} = \mathsf{Fixed.Obf}_{\mathsf{on},c_j}$,

$\mathsf{Fixed.Obf}_{\mathsf{on},c_j} \leftarrow \mathsf{Fixed.ObfOn}(\mathsf{Fixed.st}_{c_j}, C^j[\mathsf{sd}^j])$ for $c_j \in \mathbb{N}$ s.t. $\lambda^{c_j-1} < N_j \le \lambda^{c_j}$,

$\delta^j \leftarrow \mathcal{CT}^j$ where $\mathcal{CT}^j$ is the set of binary strings with same length as $\mathsf{obf}^j_{\mathsf{on}}$, for $j \in [Q]$

$\Delta^j_i \leftarrow \{0,1\}$ for $j \in [Q], i \in [N_j]$

We define S_k to be $S_k \coloneqq \{j \in [Q] : \lambda^{k-1} < N_j \le \lambda^k\}$. Since Samp is a PPT algorithm, there exists a constant c_{max} such that $\lambda^{c_{\mathsf{max}}-1} < \mathsf{max}_{j\in[Q]} N_j \le \lambda^{c_{\mathsf{max}}}$ and therefore we have $[Q] = \cup_{k\in[c_{\mathsf{max}}]} S_k$. Rearranging the terms, it suffices to show that

$$\left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\mathsf{obf}^j_{\mathsf{on}}\}_{j\in S_1}, \ldots, \{\mathsf{obf}^j_{\mathsf{on}}\}_{j\in S_{c_{\mathsf{max}}}}\right) \approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\delta^j\}_{j\in S_1}, \ldots, \{\delta^j\}_{j\in S_{c_{\mathsf{max}}}}\right)$$

holds assuming

$$\begin{aligned}&\left(\mathsf{aux}, \left\{C^j(i)\right\}_{j\in S_1, i\in[N_j]}, \ldots, \left\{C^j(i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[N_j]}\right)\\ &\approx_c \left(\mathsf{aux}, \left\{\Delta^j_i\right\}_{j\in S_1, i\in[N_j]}, \ldots, \left\{\Delta^j_i\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[N_j]}\right).\end{aligned} \tag{6.56}$$

To prove this, we prove the following lemma.

Lemma 6.40. *For $c \in [0, c_{\mathsf{max}}]$, let us consider the following statement:*

$$\begin{pmatrix} \mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\mathsf{obf}^j_{\mathsf{on}}\}_{j\in S_1}, \ldots, \{\mathsf{obf}^j_{\mathsf{on}}\}_{j\in S_c},\\ \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c+1}, i\in[\lambda^{c+1}]}, \ldots, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]} \end{pmatrix}$$

$$\approx_c \left(\mathsf{aux}, \mathsf{obf}_{\mathsf{off}}, \{\delta^j\}_{j\in S_1}, \ldots, \{\delta^j\}_{j\in S_c}, \left\{\Delta_i^j\right\}_{j\in S_{c+1}, i\in[\lambda^{c+1}]}, \ldots, \left\{\Delta_i^j\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]}\right). \tag{6.57}$$

Then, assuming the security of pPRIO*, the above computational indistinguishability for* $c = c^*$ *implies that for* $c^* + 1$ *for all* $c^* \in [0, c_{\mathsf{max}} - 1]$.

Proof. By setting

$$\begin{aligned}
\mathsf{aux}_1 &:= \left(\mathsf{aux}, \left\{\mathsf{Fixed.Obf}_{\mathsf{off},k}\right\}_{k\in[\lambda]\setminus\{c^*+1\}}\right),\\
\mathsf{aux}_2 &:= \begin{pmatrix} \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_1}, \ldots, \{\mathsf{obf}_{\mathsf{on}}^j\}_{j\in S_{c^*}},\\ \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c^*+2}, i\in[\lambda^{c^*+2}]}, \ldots, \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c_{\mathsf{max}}}, i\in[\lambda^{c_{\mathsf{max}}}]} \end{pmatrix},\\
\mathsf{Circuits} &:= \left\{C^j[\mathsf{sd}^j]\right\}_{j\in S_{c^*+1}}
\end{aligned}$$

Equation (6.57) with $c = c^*$ implies

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \left\{C^j[\mathsf{sd}^j](i)\right\}_{j\in S_{c^*+1}, i\in[\lambda^{c^*+1}]}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \left\{\Delta_i^j\right\}_{j\in S_{c^*+1}, i\in[\lambda^{c^*+1}]}\right),$$

where $\mathbf{x}$ is a random string with the same length as aux_2. By applying Theorem 6.38, we have

$$\left((\mathsf{aux}_1, \mathsf{aux}_2), \mathsf{Obf}_{\mathsf{off},c^*+1}, \left\{\mathsf{obf}_{\mathsf{on}}^j\right\}_{j\in S_{c^*+1}}\right) \approx_c \left((\mathsf{aux}_1, \mathbf{x}), \mathsf{Obf}_{\mathsf{off},c^*+1}, \left\{\delta^j\right\}_{j\in S_{c^*+1}}\right).$$

By rearranging the terms, we can observe that the above equation is equivalent to Equation (6.57) with $c = c^* + 1$. ∎

We then conclude the proof of Theorem 6.39 using Theorem 6.40. Our goal is to prove Equation (6.57) with $c = c_{\mathsf{max}}$, since it is equivalent to Equation (6.55). This is proven by the induction, where the induction step is already provided by Theorem 6.40. Therefore, it suffices to prove Equation (6.57) with $c = 0$. This immediately follows from Equation (6.56) and by the security of PRF, since the LHS distribution of Equation (6.57) with $c = 0$ is obtained by padding the LHS distribution of Equation (6.56) with PRF

values $\{\mathsf{PRF}(\mathsf{sd}^j, i)\}_{c\in[c_{\max}],j\in S_c,i\in[N_j+1,\lambda^c]}$, which are pseudorandom. This completes the proof of Theorem 6.39. ■

Instantiation. Instantiating the underlying pPRIO scheme as in Theorem 6.37, we get the following theorem.

Theorem 6.41. *Assuming evasive LWE (Assumption 2.6) and LWE (Assumption 2.5), there exists a secure* pPRIO *scheme for flexible input domain supporting circuits of bounded size* $L = \mathrm{poly}(\lambda)$ *with*

$$|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{Obf}_{\mathsf{on}}| = \mathrm{poly}(L, \lambda).$$

6.6.4 Extending the Output Length.

In our construction of pPRIO with flexible input space in Section 6.6.3, we only consider the case of the output space of the obfuscated circuit being $\{0, 1\}$. We can extend the output space to be $\{0, 1\}^{\ell_{\mathsf{out}}}$ for arbitrary polynomial ℓ_{out} as follows. We do not change the off-line phase of the obfuscation algorithm. When we run on-line phase of the obfuscation algorithm, we derive sub-circuits $C_1, \ldots C_{\ell_{\mathsf{out}}}$ of C with binary outputs, where C_j takes on input i and outputs the j-th bit of $C(i)$. We then run $\mathsf{ObfOn}(\mathsf{st}, C_j)$ for all $j \in [\ell_{\mathsf{out}}]$ and output them as the obfuscation of C. The evaluation algorithm then recovers each output separately and combines them. It is straightforward to see that the security of the construction is preserved with this modification. As for the efficiency, we still maintain the asymptotic efficiency of

$$|\mathsf{Obf}_{\mathsf{off}}| = \mathrm{poly}(L, \lambda), \quad |\mathsf{Obf}_{\mathsf{on}}| = \ell_{\mathsf{out}}\mathrm{poly}(L, \lambda) = \mathrm{poly}(L, \lambda)$$

where $(\mathsf{Obf}_{\mathsf{off}}, \mathsf{Obf}_{\mathsf{on}}) \leftarrow \mathsf{Obf}(1^\lambda, C)$ for circuit C whose size is bounded by $L = L(\lambda)$.

6.7 INSTANTIATING THE RANDOM ORACLES USING prIO

Hohenberger, Sahai, and Waters [119] show that some applications of random oracles (ROM) can be made secure in the standard model by instantiating the hash functions using iO in a specific manner. In this section, we discuss that we can replace full-fledged iO with prIO in these applications. As a concrete example, we show that full-domian hash (FDH) signatures can be proven secure in the standard model if we instantiate the hash function using prIO in place of iO. We also refer to Section 6.8 for the application of prIO for instantiating random oracle in Sakai-Ohgishi-Kasahara ID-based NIKE.

6.7.1 Full-Domain Hash Signatures (Selectively Secure) from prIO

Ingredients. We make use of the following ingredients.

1. A one-way trapdoor permutation family (TDP).
2. Punctured PRFs
3. A prIO scheme.

1. $\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter and does the following:
   - It runs the setup of the TDP to obtain a public index PK along with a trapdoor SK, yielding the map $g_{\mathsf{PK}} : \{0,1\}^n \to \{0,1\}^n$ together with its inverse g^{-1}_{SK}.
   - It chooses a puncturable PRF key K for F where $F(K,\cdot) : \{0,1\}^\lambda \to \{0,1\}^n$. Then, it creates a prIO obfuscation of the of the program Full Domain Hash Figure 6.2. We refer to the obfuscated program as the function $H : \{0,1\}^\lambda \to \{0,1\}^n$. We need the truth table of the PRF to be pseudorandom against an adversary whose size is polynomial in κ, where κ is the parameter specified by our prIO. This can be achieved assuming the subexponential security for the PRF.
   - It outputs the verification key VK as the trapdoor index PK as well as the hash function $H(\cdot)$. The secret key is the trapdoor SK.
2. $\mathsf{Sign}(\mathsf{SK}, \mathsf{m})$: Output $\sigma = g^{-1}_{\mathsf{SK}}(H(m))$.
3. $\mathsf{Verify}(\mathsf{VK}, m, \sigma)$: Check if $g_{\mathsf{PK}}(\sigma) = H(m)$ and output "Accept" if and only if this is true.

Correctness follows from the correctness of the TDP. We sketch security next.

Full Domain Hash H

Constants: PRF key K, TDP key PK
Input: Message m.
- Output $g_{\mathsf{PK}}(F(K, m))$.

Figure 6.2: Full Domain Hash

Full Domain Hash* H

Constants: Punctured PRF key $K(m^*)$, m^*, z^*, TDP key PK
Input: Message m.
- If $m = m^*$, output z^* and exit.
- Else output $g_{\mathsf{PK}}(F(K, m))$.

Figure 6.3: Full Domain Hash*

Security The security proof closely resembles that of [119], except that it can be simpler since the security guarantee of prIO is stronger than $i\mathcal{O}$ for the specific case of pseudorandom functionalities. In particular, prIO does not require two circuits to have identical truth tables in order to guarantee security, but allows different truth tables so long as they are pseudorandom (given auxiliary information).

Theorem 6.42. *If the obfuscation scheme in Section 6.5.2 is secure as per Definition 6.14 with respect to a parameter κ and samplers that output circuits defined as Figure 6.2 and 6.3, F is subexponentially secure punctured* PRF*, and the trapdoor permutation scheme TDP is one-way, then the above signature scheme is selectively secure.*

Proof. The proof follows a similar sequence of hybrids as [119] [18]. In the first hybrid we move to using the obfuscation of the circuit in Figure 6.3 with z^* being a random point in $\{0,1\}^n$. Since the truth tables of the two programs are pseudorandom given PK and SK (even against an adversary that runs in polynomial time in κ), indistinguishability holds by the guarantee of prIO. This allows to reduce the security to that of the TDP, exactly

[18] We can skip Hybrid 1 in [119] because we don't need the programs to be exactly equivalent as discussed above.

as in [119] since a valid signature would imply a preimage to z^* and other signatures can be simulated using the PRF key. ■

6.7.2 Discussion about Other Applications.

Hohenberger, Sahai, and Waters [119] demonstrate that the selective security of the full-domain hash (FDH) signature based on trapdoor permutations (TDP), the adaptive security of RSA FDH signatures [79], the selective security of BLS signatures, and the adaptive security of BLS signatures [45] can be proven in the standard model by carefully instantiating the underlying hash function by $i\mathcal{O}$ for each application. As shown in Section 6.7.1, the random oracle in the FDH signature can be instantiated using prIO instead of full-fledged $i\mathcal{O}$. Similarly, we can instantiate the random oracle in selectively secure BLS signatures with prIO, following a strategy similar to that in [119]. At a high level, these proofs follow those in the random oracle model (ROM), where we use $i\mathcal{O}$ to obfuscate a derandomized version of the simulator for the hash function in ROM-based proofs. In these settings, the truth table of the simulated hash function is pseudorandom, allowing us to follow the same proof strategy using prIO.

For adaptively secure RSA FDH and BLS signatures, the situation is different. In these cases, Hohenberger et al. adopt an alternative proof strategy that deviates from the high level strategy of obfuscating the simulator for the proof in the ROM. This is due to the fact that the original proofs [45, 79] are incompatible with the conditions required for using $i\mathcal{O}$, where the truth table of the hash functions must remain unchanged across game hops. To be compatible with $i\mathcal{O}$, they introduce a structure for the hash function, making its truth table no longer pseudorandom. This prevents us from replacing the hash function with prIO following their approach.

To instantiate the hash function with prIO, we revert to the original ROM-based proof strategy [45, 79]. Unlike the $i\mathcal{O}$-based approach, prIO-based proof does not require the truth table of the hash function to remain unchanged across game hops; it only requires

the truth table to be pseudorandom. This relaxed condition enables the use of the original ROM security proofs. We omit the details on these applications, since they are similar to and simpler than our random oracle instantiation for the Sakai-Ohgishi-Kasahara ID-based NIKE in Section 6.8.

6.8 ID-BASED NON-INTERACTIVE KEY EXCHANGE

In this section we show our construction of ID-NIKE scheme. The construction is the same as the ID-based NIKE system by Sakai, Ohgishi, and Kasahara [159] except that the hash function modeled as random oracle is replaced with an obfuscation of a PRF. This leads to the first instantiation of ID-NIKE without multi-linear maps [88] or indistinguishability obfuscation in the standard model.

6.8.1 Construction

Building Blocks. We require the following building blocks for our construction.

1. We use prIO scheme (iO, Eval) with input space $\mathcal{ID} \stackrel{\text{def}}{=} \{0,1\}^n$ that is secure as per Definition 6.14 with parameter κ. By using our construction in Section 6.5.2 instantiated by $\mathsf{mi\text{-}prFE}$ construction in Section 6.3.2, we have $\kappa = \lambda^{n^2 \log \lambda}$.

2. Symmetric pairing $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ with prime order p that is equipped with an efficiently computable function $\mathsf{MapToPoint} : \mathcal{D}_{\mathsf{MTP}} \to \mathbb{G}$, where $\mathcal{D}_{\mathsf{MTP}}$ is an efficiently samplable domain. We assume that the DBDH assumption (Assumption 6.16) holds on $\mathcal{G}$.
We need that there is an efficiently computable *randomized* function $\mathsf{MapToPoint}^{-1}$ satisfying $\mathsf{MapToPoint}(\mathsf{MapToPoint}^{-1}(h)) = h$ for all $h \in \mathbb{G}$. We also need $\mathsf{MapToPoint}^{-1}(h) \equiv x$ for $x \leftarrow \mathcal{D}_{\mathsf{MTP}}$ and $h \leftarrow \mathbb{G}$. We denote the randomness space of the algorithm $\mathsf{MapToPoint}^{-1}$ by $\mathcal{R}_{\mathsf{MTP}}$.

3. A subexponentially secure pseudorandom function $\mathsf{PRF} : \{0,1\}^{\Lambda} \times \mathcal{ID} \to \mathcal{D}_{\mathsf{MTP}}$. Here, $\Lambda = \Lambda(\lambda)$ is set so that $2^{\Lambda^\delta} > \kappa^{\omega(1)}$ holds, where δ is defined as a constant such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$ against PRF for all sufficiently large λ. An example choice would be to take $\Lambda := (n^2\lambda)^{1/\delta}$.

We describe the construction below.

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter 1^λ, it chooses the description of symmetric

pairing groups $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ with prime order $p > 2^{2\lambda}$, a PRF key $\mathsf{sd} \leftarrow \{0,1\}^{\Lambda}$, and $\alpha \leftarrow \mathbb{Z}_p$ and computes $\tilde{C} \leftarrow i\mathcal{O}(\mathsf{PRF}(\mathsf{sd}, \cdot))$. Here, the obfuscated circuit $\mathsf{PRF}(\mathsf{sd}, \cdot)$ is appropriately padded so that its size is the same as the circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}]$ described in Figure 6.4. Finally, it outputs $\mathsf{mpk} = (\mathcal{G}, \tilde{C})$ and $\mathsf{msk} = \alpha$.

$\mathsf{Ext}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$**:** On input $\mathsf{mpk} = (\mathcal{G}, \tilde{C})$, $\mathsf{msk} = \alpha$, and an identity id, it computes $\mathsf{H}(\mathsf{id})$, where the hash function $\mathsf{H} : \mathcal{ID} \to \mathbb{G}$ is defined as $\mathsf{H}(\mathsf{id}) \stackrel{\text{def}}{=} \mathsf{MapToPoint}(\tilde{C}(\mathsf{id}))$. It then computes and outputs $\mathsf{usk}_{\mathsf{id}} = \mathsf{H}(\mathsf{id})^{\alpha}$.

$\mathsf{Share}(\mathsf{mpk}, \mathsf{usk}_{\mathsf{id}_1}, \mathsf{id}_2)$**.** Given the master public key mpk, a user secret key $\mathsf{usk}_{\mathsf{id}_1} = \mathsf{H}(\mathsf{id}_1)^{\alpha}$, and an identity $\mathsf{id}_2 \in \mathcal{ID}$, it computes and outputs $K \stackrel{\text{def}}{=} e(\mathsf{usk}_{\mathsf{id}_1}, \mathsf{H}(\mathsf{id}_2))$ as the shared key.

Correctness. The correctness of the construction can be seen by $e(\mathsf{usk}_{\mathsf{id}_1}, \mathsf{H}(\mathsf{id}_2)) = e(\mathsf{H}(\mathsf{id}_1), \mathsf{H}(\mathsf{id}_2))^{\alpha} = e(\mathsf{usk}_{\mathsf{id}_2}, \mathsf{H}(\mathsf{id}_1))$.

6.8.2 Security Proof

Theorem 6.43. *The construction above is secure as per Definition 6.8 assuming that the DBDH assumption over $\mathcal{G}$ holds (See Assumption 6.16),* prIO *is secure with parameter κ (as per Definition 6.14) and with respect to a sampler specified in the proof of Theorem 6.44, and* PRF *is subexponentially secure.*

Proof. We prove the security of the scheme via the following hybrids. Let us fix a PPT adversary $\mathcal{A}$. We denote the advantage of the adversary by ϵ and without loss of generality, we assume that $\mathcal{A}$ makes exactly Q key extraction queries. For the sake of contradiction, let us assume that ϵ is non-negligible. In the following, we denote the advantage of the adversary in Game-xx by ϵ_{xx}.

Game-0. This is the security game for ID-NIKE. By definition, we have $\epsilon_0 = \epsilon$.

Game-1. In this game, we sample random function $\mathsf{R} : \mathcal{ID} \to [2Q]$ at the end of the game independently from anything else in the game. We then change the adversary $\mathcal{A}$ so that it outputs a random coin as its guess if the following does *not* hold:

$$\mathsf{R}(\mathsf{id}_1^*) = 1 \ \wedge \ \mathsf{R}(\mathsf{id}_2^*) = 2 \ \wedge \ \mathsf{R}(\mathsf{id}_1) \notin \{1,2\} \ \wedge \ \cdots \ \wedge \ \mathsf{R}(\mathsf{id}_Q) \notin \{1,2\}, \quad (6.58)$$

where id_1^* and id_2^* are the challenge identities and $\mathsf{id}_1, \ldots, \mathsf{id}_Q$ are the identities for which key extraction queries are made by $\mathcal{A}$. If the abort condition above is not satisfied, the output of $\mathcal{A}$ is unchanged. Since Equation (6.58) holds with probability $\frac{1}{4Q^2} \cdot \left(1 - \frac{2}{2Q}\right)^Q = \Theta(\epsilon_0/Q^2)$, we have $\epsilon_1 = \Theta(\epsilon_0/Q^2)$.

Game-2. In this game, we choose R at the beginning of the game and stop the game immediately and force $\mathcal{A}$ to output random guess if $\mathcal{A}$ makes a key extraction query or challenge query that violates Equation (6.58). With this change, the distribution of the output by $\mathcal{A}$ is unchanged and therefore we have $\epsilon_2 = \epsilon_1$.

Game-3. In this game, we replace R with a pseudorandom function $\overline{\mathsf{PRF}} : \{0,1\}^\lambda \times \mathcal{ID} \to [2Q]$. In more detail, the game samples $\overline{\mathsf{sd}} \leftarrow \{0,1\}^\lambda$ and uses $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \cdot)$ in place of the function $R(\cdot)$ when we check the abort condition. It is straightforward to see that $|\epsilon_3 - \epsilon_2| \leq \mathsf{negl}(\lambda)$ by the security of PRF.

Game-4. In this game, we change how $\tilde{C}$ is computed. In more detail, we sample $\beta, \gamma \leftarrow \mathbb{Z}_p$ and $\widetilde{\mathsf{sd}} \leftarrow \{0,1\}^\Lambda$ at the beginning of the game. We then set $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$ as Figure 6.4, where we use yet another pseudorandom function $\widetilde{\overline{\mathsf{PRF}}} : \{0,1\}^\Lambda \times \mathcal{ID} \to \mathcal{R}_{\mathsf{MTP}} \times \mathbb{Z}_p$ in the description of the circuit. In this game, $\tilde{C}$ is computed as $\tilde{C} \leftarrow i\mathcal{O}(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma])$. As we will show in Theorem 6.44, we have $|\epsilon_4 - \epsilon_3| \leq \mathsf{negl}(\lambda)$ by the security of PRIO and by the subexponential security of PRF and $\widetilde{\overline{\mathsf{PRF}}}$.

Circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}]$

- On input id, compute $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id})$.
- Compute

$$v_{\mathsf{id}} \stackrel{\text{def}}{=} \begin{cases} \mathsf{MapToPoint}^{-1}(g^{\beta+x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{if } \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 1, \\ \mathsf{MapToPoint}^{-1}(g^{\gamma+x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{if } \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 2, \\ \mathsf{MapToPoint}^{-1}(g^{x_{\mathsf{id}}}; R_{\mathsf{id}}) & \text{otherwise.} \end{cases} \tag{6.59}$$

- Output v_{id}.

Figure 6.4: Description of the circuit $F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}]$.

Game-5. In this game, we no longer use the exponents β and γ explicitly in answering the key extraction queries. In more detail, when the adversary makes a key extraction query to id, we proceed as follows. First, we check whether $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \cdot) \notin \{1, 2\}$ or not. If it does not hold, we abort the game as specified in Game-2. Otherwise, we compute $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id})$ and return $\mathsf{usk}_{\mathsf{id}} \stackrel{\text{def}}{=} (g^{\alpha})^{x_{\mathsf{id}}}$ as the secret key.

We claim that this change does not alter the view of the adversary. To see this, we observe that

$$\begin{aligned} \mathsf{H}(\mathsf{id}) = \mathsf{MapToPoint}(\tilde{C}(\mathsf{id})) &= \mathsf{MapToPoint}\big(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^{\beta}, g^{\gamma}](\mathsf{id})\big) \\ &= \mathsf{MapToPoint}\big(\mathsf{MapToPoint}^{-1}(g^{x_{\mathsf{id}}}; R_{\mathsf{id}})\big) = g^{x_{\mathsf{id}}} \end{aligned}$$

holds, where the first equation holds by the definition of H, the second by the correctness of iO, the third by $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) \notin \{1, 2\}$ and Equation (6.59), and the fourth by the property of $\mathsf{MapToPoint}^{-1}$. This implies $(g^{\alpha})^{x_{\mathsf{id}}} = \mathsf{H}(\mathsf{id})^{\alpha}$ as desired. We therefore have $\epsilon_5 = \epsilon_4$.

Game-6. In this game, we change the way K^* is computed when $\mathsf{coin} = 0$. In particular, we compute $T = e(g, g)^{\alpha\beta\gamma}$ at the beginning of the game and when the adversary

asks for the challenge key, we return

$$K^* = T \cdot e(g^\alpha, g^\beta)^{x_{\mathsf{id}_2^*}} \cdot e(g^\alpha, g^\gamma)^{x_{\mathsf{id}_1^*}} \cdot e(g, g^\alpha) \tag{6.60}$$

to the adversary, where $(R_{\mathsf{id}_i^*}, x_{\mathsf{id}_i^*}, y_{\mathsf{id}_i^*}) = \widetilde{\mathsf{PRF}}(\widetilde{\mathsf{sd}}, \mathsf{id}_i^*)$ for $i = 1, 2$.

We claim that this change does not alter the view of the adversary. To see this, recall that the game aborts unless $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}_1^*) = 1 \wedge \overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}_2^*) = 2$. The first condition implies $\mathsf{H}(\mathsf{id}_1^*) = \mathsf{MapToPoint}(v_{\mathsf{id}_1^*}) = g^{\beta + x_{\mathsf{id}_1^*}}$ by Equation (6.59) and by the property of $\mathsf{MapToPoint}^{-1}$. Similarly, we have $\mathsf{H}(\mathsf{id}_1^*) = g^{\gamma + x_{\mathsf{id}_2^*}}$. Thus, we have

$$e(\mathsf{H}(\mathsf{id}_1^*), \mathsf{H}(\mathsf{id}_2^*))^\alpha = e(g^{\beta + x_{\mathsf{id}_1^*}}, g^{\gamma + x_{\mathsf{id}_2^*}})^\alpha = T \cdot e(g^\alpha, g^\beta)^{x_{\mathsf{id}_2^*}} \cdot e(g^\alpha, g^\gamma)^{x_{\mathsf{id}_1^*}} \cdot e(g, g^\alpha)^{x_{\mathsf{id}_1^*} x_{\mathsf{id}_2^*}}$$

as desired. We therefore have $\epsilon_6 = \epsilon_5$.

Note that in this game, we do not need to know α any more to efficiently simulate the game. Rather, it suffices to know $g^\alpha, g^\beta, g^\gamma$, and $T = e(g, g)^{\alpha\beta\gamma}$.

Game-7. In this game, we replace T with random $T \leftarrow \mathbb{G}_T$. We claim that the adversary cannot distinguish this game from the previous game. This can be seen by straightforward reduction to the DBDH assumption, where we simulate Game-6 if $T = e(g, g)^{\alpha\beta\gamma}$ and Game-7 if $T \leftarrow \mathbb{G}_T$. Indeed, we only need to know g^β and g^γ for simulating mpk, g^α for answering the key extraction queries, and g^α, g^β, g^γ, and T for simulating the challenge key. We therefore have $|\epsilon_6 - \epsilon_7| \leq \mathsf{negl}(\lambda)$.

Game-8. In this game, we change the challenge shared key K^* to be random group element in $\mathbb{G}_T$, regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$. We claim that this is a conceptual change. This can be easily seen by observing that K^* computed as Equation (6.60) is distributed uniformly at random over $\mathbb{G}_T$ if so is T. Thefefore,

we have $\epsilon_8 = \epsilon_7$.

Clearly, we have $\epsilon_8 = 0$, since no information of coin is leaked to the adversary in Game-8. However, from the above discussion, we have $0 = \epsilon_8 \geq \Theta(\epsilon)/Q^2 - \mathsf{negl}(\lambda)$ by the triangle inequality. This contradicts our assumption that ϵ is non-negligible. ■

To complete the proof of Theorem 6.43, it remains to prove Theorem 6.44.

Lemma 6.44. *Assuming that $i\mathcal{O}$ is a secure PRIO scheme as per Definition 6.14 with the parameter κ and PRF and $\widetilde{\mathsf{PRF}}$ are subexponentially secure, we have $|\epsilon_3 - \epsilon_4| \leq \mathsf{negl}(\lambda)$.*

Proof. To prove the lemma, it suffices to show $(i\mathcal{O}(\mathsf{PRF}(\mathsf{sd}, \cdot)), \overline{\mathsf{sd}}, \mathcal{G}) \approx_c (i\mathcal{O}(F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]), \overline{\mathsf{sd}}, \mathcal{G})$, since we can simulate the game in a way that we simulate Game-3 if the given terms come from the left distribution and Game-4 otherwise. Any adversary distingushing the games can be turned into an adversary that distinguishes the distributions. By considering a sampler Samp that takes as input 1^λ and outputs the auxiliary information $\mathsf{aux} \stackrel{\mathrm{def}}{=} (\overline{\mathsf{sd}}, \mathcal{G})$ and circuits $C_0 \stackrel{\mathrm{def}}{=} \mathsf{PRF}(\mathsf{sd}, \cdot)$ and $C_1 \stackrel{\mathrm{def}}{=} F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]$ and invoking the security of PRIO, we can see that it suffices to prove

$$\begin{aligned}(1^\kappa, \mathsf{aux}, \{\mathsf{PRF}(\mathsf{sd}, \mathsf{id})\}_{\mathsf{id} \in \mathcal{ID}}) &\approx_c (1^\kappa, \mathsf{aux}, \{\delta_{\mathsf{id}} \leftarrow \mathcal{D}_{\mathsf{MTP}}\}_{\mathsf{id} \in \mathcal{ID}}) \\ &\approx_c \left(1^\kappa, \mathsf{aux}, \{v_{\mathsf{id}} = F[\overline{\mathsf{sd}}, \widetilde{\mathsf{sd}}, g^\beta, g^\gamma]\}_{\mathsf{id} \in \mathcal{ID}}\right). \quad (6.61)\end{aligned}$$

The former indistinguishability holds by the subexponential security of PRF. In particular, by our choice of the parameter $2^{\Lambda^\delta} > \kappa^{\omega(1)}$, we can conclude that the adversary with running time $\mathrm{poly}(\kappa)$ cannot distinguish the distributions with advantage more than $\mathsf{negl}(\kappa)$. To prove the latter indistinguishability, we further consider the following distributions. In the following, we do not change the distribution of $\mathsf{aux} = (\overline{\mathsf{sd}}, \mathcal{G})$ and only focus on the distribution of $\{v_{\mathsf{id}}\}_{\mathsf{id}}$.

- This is the rightmost distribution of Equation (6.61), where v_{id} is computed as Equation (6.59) for $(R_{\mathsf{id}}, x_{\mathsf{id}}) = \widetilde{\mathsf{PRF}}(\mathsf{id})$.
- This is the same as the previous distribution except that $(R_{\mathsf{id}}, x_{\mathsf{id}})$ that is used for

computing v_{id} is chosen uniformly at random from their respective domains.

- $\{v_{\mathsf{id}}\}_{\mathsf{id}}$, where $v_{\mathsf{id}} \leftarrow \mathcal{D}_{\mathsf{MTP}}$. Observe that this is the middle distribution in Equation (6.61).

It suffices to prove that the first and the third distributions are indistinguishable given $(1^\kappa, \mathsf{aux})$. We first observe that the first and the second distributions are computationally indistinguishable by the subexponential security of $\widetilde{\mathsf{PRF}}$ and our choice of parameter Λ, where we have $2^{\Lambda^\delta} > \kappa^{\omega(1)}$. To conclude the proof, we show that the second and the third distributions are actually the same distribution. In particular, we show that v_{id} is uniformly distributed over $\mathcal{D}_{\mathsf{MTP}}$ for each id in the second distribution. Here, we consider the case of $\overline{\mathsf{PRF}}(\overline{\mathsf{sd}}, \mathsf{id}) = 1$. The other cases follow similarly. By the assumption, we know that $g^{\beta + x_{\mathsf{id}}}$ is distributed uniformly at random over $\mathbb{G}$. Then, by the property of $\mathsf{MapToPoint}^{-1}$ and by the fact that R_{id} is random, v_{id} is uniformly distributed over $\mathcal{D}_{\mathsf{MTP}}$ as desired. ■

Theorem 6.45. *Let $\kappa = \lambda^{n^2 \log \lambda}$. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, non-uniform sub-exponential LWE, and the DBDH assumption, there exists a secure ID-NIKE scheme.*

APPENDIX

6.A PSEUDORANDOM FE WITH STRONGER SECURITY

In this section, we explain how to extend [12] construction to achieve strengthened security notion for pseudorandom FE that we call non-uniform κ-prCT security (Definition 6.3), which is required for many of our applications. The stronger version of the security is not proven in [12], but will be required for the construction of $\mathsf{mi\text{-}prFE}$ in Section 6.3.

6.A.1 Proof for Non-Uniform κ-prCT Security

Theorem 6.46. *Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary,*

and non-uniform sub-exponential LWE (Assumption 2.5), there exists a prFE *scheme satisfying* κ-prCT *security as per Definition 6.3.*

Proof. We prove that the construction in Section 5.3.2 with λ being replaced by appropriately chosen $\Lambda = \mathrm{poly}(\lambda)$ satisfies the security notion. The reason why we need this scaled version of the security parameter is that we have to consider an adversary whose running time can be κ, which is exponential in λ. In particular, in the security proof, we require LWE and PRF to be secure even against an adversary that runs polynomial time in κ. To handle such an adversary, we rely on the subexponential security of LWE and PRF. By our assumption, there exists $0 < \delta < 1$ such that there is no adversary with size 2^{λ^δ} and distinguishing advantage $2^{-\lambda^\delta}$ against LWE and PRF for all sufficiently large λ. To satisfy the requirement, we generate PRF and LWE instances with respect to a larger security parameter Λ that satisfies $2^{\Lambda^\delta} \geq \kappa^{\omega(1)}$. An example choice of the parameter would be $\Lambda := \lambda^{(c+1)/\delta}$.

The overall structure of the proof is the same as that of Theorem 5.15.

We start with a sampler $\mathsf{Samp}_{\mathsf{prFE}}$ and an adversary $\mathcal{A}_1$ satisfying $\mathrm{Size}(\mathsf{Samp}_{\mathsf{prFE}}) \leq \mathrm{poly}(\lambda')$ and $\mathrm{Size}(\mathcal{A}_1) \leq \mathrm{poly}(\kappa)$ for $\lambda' < \kappa$.[19] We denote the size of $\mathcal{A}_1$ by t and the distinguishing advantage for the distributions in Equation (5.19) by ϵ. Assuming non-uniform κ-evasive LWE with respect to Samp defined from $\mathsf{Samp}_{\mathsf{prFE}}$ as in the proof of Theorem 5.15, we obtain an adversary $\mathcal{A}_0$ whose size is $Q(\lambda')t$ and the distinguishing advantage against the distributions in Equation (5.22) is $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ for some polynomial Q by applying Theorem 6.15. We then consider the same sequence of hybrids as that for the proof of Theorem 5.15. Note that here, the security parameter for the construction λ is replaced by Λ and Q_{msg} and Q_{key} are bounded by $\mathrm{poly}(\lambda')$, since the size of the sampler is $\mathrm{poly}(\lambda')$. By the definition of the hybrids, the adversary has the distinguishing advantage $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$ for Hyb_0 and Hyb_8. Furthermore, we argue

[19] Here, we deviate from our convention that the adversary runs in time polynomial in its input length. The input length of $\mathcal{A}_1$ is $\mathrm{poly}(\lambda')$, but its running time is $\mathrm{poly}(\kappa)$, which may be super-polynomial in λ'.

that the distinguishing advantage between Hyb_0 and Hyb_7 is only $\mathsf{negl}(\kappa)$. We inspect this in the following:

- The changes from Hyb_0 to Hyb_1, from Hyb_2 to Hyb_3, from Hyb_4 to Hyb_5, and from Hyb_6 to Hyb_7 are statistical, where each statistical difference is bounded by $\mathrm{poly}(\lambda')/2^{-\Lambda}$. We have $\mathrm{poly}(\lambda')/2^{-\Lambda} \leq \mathrm{poly}(\kappa)/2^{-\Lambda} = \mathsf{negl}(\kappa)$ by our choice of Λ.

- The change from Hyb_1 to Hyb_2 is computational, which is dependent on the hardness of LWE. Since the size of $\mathcal{A}_0$ is bounded by $\mathrm{poly}(\kappa)$, the distinguishing advantage between Hyb_1 and Hyb_2 should be bounded by $\mathsf{negl}(\kappa)$ by the subexponential hardness of LWE and by our choice of Λ.

- The change from Hyb_3 to Hyb_4 is computational, which is dependent on the security of PRF. Since the size of $\mathcal{A}_0$ is bounded by $\mathrm{poly}(\kappa)$, the distinguishing advantage between Hyb_3 and Hyb_4 should be bounded by $\mathsf{negl}(\kappa)$ by the subexponential security of PRF.

- The changes from Hyb_5 to Hyb_6 is conceptual and thus they are equivalent.

We therefore conclude that the distinguishing advantage of $\mathcal{A}_0$ against Hyb_7 and Hyb_8 should be $\epsilon/Q(\lambda') - \mathsf{negl}(\kappa)$. Then, from $\mathcal{A}_0$, it is straightforward to extract a distinguisher $\mathcal{A}'_0$ against the distributions in Equation (5.20) with the same advantage and almost the same size. This concludes the proof of the theorem. ■

Theorem 6.47. *Let $\kappa = 2^{\lambda^c}$ for some constant c. Assuming non-uniform κ-evasive LWE (Assumption 6.14), subexponentially secure PRF against non-uniform adversary, and non-uniform sub-exponential LWE (Assumption 2.5), there exists a* prFE *scheme for function class $\mathcal{F}_{L(\lambda),\ell(\lambda),\mathsf{dep}(\lambda)} = \{f : \{0,1\}^L \to \{0,1\}^\ell\}$ satisfying κ-*prCT *security as per Definition 6.3 with efficiency*

$$|\mathsf{mpk}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{sk}_f| = \ell \cdot \mathrm{poly}(\mathsf{dep}, \lambda), \quad |\mathsf{ct}| = L \cdot \mathrm{poly}(\mathsf{dep}, \lambda).$$

where $\mathsf{dep} = \mathrm{poly}(\lambda)$ is the depth bound on the functions supported by the scheme.

CHAPTER 7

EVASIVE LWE: CLASSES, ATTACKS, AND REPERCUSSIONS

The evasive LWE assumption, introduced by [169, 163], has emerged as a powerful framework to argue the security of advanced cryptographic primitives. Over time, several variants of this assumption have been formulated, depending on the visibility of randomness and structural components. We recall the general formulation of the evasive LWE assumption. Let Samp be any PPT algorithm that, on input 1^λ, outputs a matrix $\mathbf{P}$ and auxiliary information aux containing all randomness used by Samp. Let $\mathbf{B}$ be sampled uniformly at random from $\mathbb{Z}_q^{n\times m}$, and $\mathbf{s}$ be short vector sampled uniformly at random from $\mathbb{Z}_q^n$.

The assumption postulates that:

$$\text{if} \quad (\mathbf{B}, \mathbf{P}, \mathbf{s}^\intercal\mathbf{B}, \mathbf{s}^\intercal\mathbf{P}, \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \$, \$, \mathsf{aux}),$$

$$\text{then} \quad (\mathbf{B}, \mathbf{P}, \mathbf{s}^\intercal\mathbf{B}, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \$, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux}),$$

where $\mathbf{B}^{-1}(\mathbf{P})$ denotes a short preimage of $\mathbf{P}$ under $\mathbf{B}$.

This chapter systematically explores the different formulations of the evasive LWE assumption. We first define the public-coin version, where the sampler's randomness is fully exposed. We then discuss two flavors of the private-coin version: the *binding* variant, where the matrices $\mathbf{B}$ and $\mathbf{P}$ remain publicly known but the sampler's internal randomness is hidden, and the *hiding* variant, where $\mathbf{B}$, $\mathbf{P}$, or both are partially or fully hidden from the adversary's view. Later, we also discuss the *circular* variant of evasive LWE, where the distributions in the assumption additionally includes some circular encodings.

For each variant, we present the formal definitions, followed by known attacks that highlight their limitations. We also discuss the implications of these attacks on the prior versions of our constructions and the assumptions. We now proceed to formally define each variant, starting with the public-coin evasive LWE assumption.

7.1 PUBLIC-COIN EVASIVE LWE

The public-coin evasive LWE was introduced by [169]. In this version, the sampler Samp reveals all the random coins used during sampling.

Definition 7.1 (Public-Coin Evasive LWE)**.** Let Samp be a PPT algorithm that, on input 1^λ, outputs

$$\mathbf{A}' \in \mathbb{Z}_q^{n\times m'}, \quad \mathbf{P} \in \mathbb{Z}_q^{n\times t}, \quad \mathsf{aux} \in \{0,1\}^*.$$

We define the following advantage functions:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PRE}}_{\mathcal{A}_0}(\lambda) := \mathsf{Pr}\big[\mathcal{A}_0(\mathbf{A}',\ \mathbf{s}^\mathsf{T}\mathbf{A}'+\mathbf{e}',\ \mathbf{s}^\mathsf{T}\mathbf{B}+\mathbf{e},\ \mathbf{s}^\mathsf{T}\mathbf{P}+\mathbf{e}'',\ \mathsf{aux}) = 1\big] \\ - \mathsf{Pr}\big[\mathcal{A}_0(\mathbf{A}',\ \mathbf{c},\ \mathbf{c}_0,\ \mathbf{c}',\ \mathsf{aux}) = 1\big],\end{aligned} \tag{7.1}$$

$$\begin{aligned}\mathsf{Adv}^{\mathsf{POST}}_{\mathcal{A}_1}(\lambda) := \mathsf{Pr}\big[\mathcal{A}_1(\mathbf{A}',\ \mathbf{s}^\mathsf{T}\mathbf{A}'+\mathbf{e}',\ \mathbf{s}^\mathsf{T}\mathbf{B}+\mathbf{e},\ \mathbf{K},\ \mathsf{aux}) = 1\big] \\ - \mathsf{Pr}\big[\mathcal{A}_1(\mathbf{A}',\ \mathbf{c},\ \mathbf{c}_0,\ \mathbf{K},\ \mathsf{aux}) = 1\big],\end{aligned} \tag{7.2}$$

where

$$\begin{aligned}(\mathbf{A}',\mathbf{P},\mathsf{aux}) \leftarrow \mathrm{Samp}(1^\lambda), \mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}, \quad \mathbf{s} \leftarrow \mathbb{Z}_q^n, \\ \mathbf{c} \leftarrow \mathbb{Z}_q^m, \quad \mathbf{c}_0 \leftarrow \mathbb{Z}_q^t, \quad \mathbf{c}' \leftarrow \mathbb{Z}_q^{m'}, \\ \mathbf{e} \leftarrow \mathcal{D}^m_{\mathbb{Z}_q,\chi}, \quad \mathbf{e}' \leftarrow \mathcal{D}^{m'}_{\mathbb{Z}_q,\chi}, \quad \mathbf{e}'' \leftarrow \mathcal{D}^t_{\mathbb{Z}_q,\chi}, \\ \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P},\tau) \text{ with standard deviation } \tau = O(\sqrt{m\log q}).\end{aligned}$$

We say that the *evasive* LWE assumption holds if for every PPT Samp, $\mathcal{A}_0$, $\mathcal{A}_1$, there

exists another PPT algorithm and a polynomial $Q(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{PRE}}_{\mathcal{A}_0}(\lambda) \geq \frac{\mathsf{Adv}^{\mathsf{POST}}_{\mathcal{A}_1}(\lambda)}{Q(\lambda)} - \mathsf{negl}(\lambda).$$

Remark 42 (Noise Magnitudes [169])**.** For simplicity, we stated the assumption with all the LWE error terms $\mathbf{e}, \mathbf{e}', \mathbf{e}''$ having the same Gaussian parameter χ. It is straightforward to adapt the assumption and the scheme to a quantitatively weaker variant where the error terms in the post-condition have a larger Gaussian parameter than those in the pre-condition.

Attack on the public-coin evasive LWE and repercussions [19] presents an attack on a stronger version of public-coin evasive LWE, where the noise magnitude in the pre-condition is larger than the noise magnitude in the post-condition. We note that their attack *does not* affect the underlying assumption of the cpABE scheme of [169], which we use as a blackbox tool in Section 3.7 and Section 4.5.3.

7.2 PRIVATE-COIN BINDING EVASIVE LWE.

In the binding setting, Samp is private-coin, meaning its internal randomness remains hidden, but the matrices $\mathbf{B}$ and $\mathbf{P}$ are explicitly revealed.

Definition 7.2 (Private-Coin Binding Evasive LWE, [66, Definition 8])**.** Let $m, n, k, \ell > 0$ be integers and let q be a modulus. Let $\tau, \sigma, \sigma' > 0$. Let Samp be an algorithm that, on input 1^λ, outputs a matrix $\mathbf{P} \in \mathbb{Z}_q^{n\times k}$, a matrix $\mathbf{S} \in \mathbb{Z}_q^{\ell\times n}$, and auxiliary information aux. Let

$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, \tau), (\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathrm{Samp}(1^\lambda),$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}_q^{\ell\times m}, \sigma}, \quad \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z}_q^{\ell\times k}, \sigma'}, \mathbf{C} \leftarrow \mathbb{Z}_q^{\ell\times m}, \quad \mathbf{C}' \leftarrow \mathbb{Z}_q^{\ell\times k}.$$

For PPT distinguishers $\mathcal{A}_0$ and $\mathcal{A}_1$, we define the following functions:

$$\mathsf{Adv}^{\mathsf{Pre}}_{\mathcal{A}_0}(\lambda) := \mathsf{Pr}\left[\mathcal{A}_0(\mathbf{B}, \mathbf{P}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}) = 1\right]$$

$$- \Pr\left[\mathcal{A}_0(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{C}', \mathsf{aux}) = 1\right], \tag{7.3}$$

$$\mathsf{Adv}^{\mathsf{Post}}_{\mathcal{A}_1}(\lambda) := \Pr\left[\mathcal{A}_1(\mathbf{B}, \mathbf{P}, \mathbf{SB} + \mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1\right]$$

$$- \Pr\left[\mathcal{A}_1(\mathbf{B}, \mathbf{P}, \mathbf{C}, \mathbf{K}, \mathsf{aux}) = 1\right]. \tag{7.4}$$

We say that the binding evasive LWE assumption $\mathsf{evLWE}(q, n, m, k, \ell, \mathrm{Samp}, \tau, \sigma, \sigma')$ holds if there exists a polynomial Q such that $\deg_\lambda(Q) \leq c \cdot \deg_\lambda(|\mathrm{Samp}|)$ (for some universal constant $c > 0$), and for every PPT distinguisher $\mathcal{A}_1$ there exists a PPT distinguisher $\mathcal{A}_0$ such that

$$\mathsf{Adv}^{\mathsf{Pre}}_{\mathcal{A}_0}(\lambda) \geq \frac{\mathsf{Adv}^{\mathsf{Post}}_{\mathcal{A}_1}(\lambda)}{Q(\lambda)} - \mathsf{negl}(\lambda),$$

and $\mathrm{time}(\mathcal{A}_0) \leq \mathrm{time}(\mathcal{A}_1) \cdot Q(\lambda)$.

7.2.1 Attacks

Attack by [19, 121] The attacks proposed by [19] and [121], on the binding variant of evasive LWE, largely overlaps. They design the sampler of their counter-example by relying on the prior version of our scheme in Chapter 5 [1] and using the tools from [120]. We give a high level overview of the attack next, majorly borrowed from Section 5.1.4. Construct a sampler Samp that does the following

- Choose a function f and an input vector $\mathbf{x} \in \{0,1\}^L$ such that $f(\mathbf{x})$ is pseudorandom.

- Samples a GSW secret key $\mathbf{s}$ and a PRF seed $\mathsf{sd} \leftarrow \{0,1\}^\lambda$. It then computes a GSW ciphertext, $\mathbf{X} = \mathbf{A}_{\mathsf{fhe}}\mathbf{R} - (\mathbf{x}, \mathsf{sd}) \otimes \mathbf{G}$, using public key $\mathbf{A}_{\mathsf{fhe}} = (\bar{\mathbf{A}}_{\mathsf{fhe}} \;\; \bar{\mathbf{s}}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^\intercal_{\mathsf{fhe}})$ and randomness $\mathbf{R}$, – followed by a BGG^+ encoding of $\mathbf{X}$ using randomness $\mathbf{s}$ as $\mathbf{c}^\intercal_{\mathsf{att}} := \mathbf{s}^\intercal(\mathbf{A}_{\mathsf{att}} - \mathbf{X} \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{att}}$. It additionally computes $\mathbf{c}^\intercal_{\mathbf{B}} := \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B}}$.

- Samples a nonce $\mathbf{r} \leftarrow \{0,1\}^\lambda$ and defines function $\mathrm{F}[f, \mathbf{r}]$, with f and $\mathbf{r}$ hardwired, as
$$\mathrm{F}[f, \mathbf{r}](\mathbf{x}, \mathsf{sd}) = f(\mathbf{x}) \lfloor q/2 \rceil + \mathsf{PRF}(\mathsf{sd}, \mathbf{r}).$$
It then computes the FHE evaluation circuit $\mathsf{VEval}_{\mathrm{F}}$ w.r.t. the function $\mathrm{F}[f, \mathbf{r}]$ (this can be computed using the knowledge of $\mathrm{F}[f, \mathbf{r}]$). Next, it computes the matrix $\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$ for the circuit $\mathsf{VEval}_{\mathrm{F}}$ using the public matrix $\mathbf{A}_{\mathsf{att}}$. It sets $\mathbf{A}_{\mathrm{F}} = \mathbf{A}_{\mathsf{att}} \cdot \mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}}}$

[1] please refer to Section 5.1.4 for the prior version of our prFE scheme

and samples $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{A}_{\mathrm{F}})$.

– It outputs $(\mathbf{s}, \mathbf{P} = \mathbf{A}_{\mathrm{F}}, \mathsf{aux} = (\mathbf{X}, \mathbf{c}_{\mathsf{att}}, \mathbf{c}_{\mathbf{B}}, f, \mathbf{r}, \mathbf{A}_{\mathsf{att}}))$.

Next, we discuss why the above sampler satisfies pre-condition but breaks post-condition and thus violating the private-coin binding evaisve LWE assumption.

- *Pre-condition holds.* To argue that the pre-condition holds, it suffices to show the pseudorandomness of $(\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X}, \mathbf{c}_{\mathbf{P}})$, where $\mathbf{c}_{\mathbf{P}} = \mathbf{sP} + \mathbf{e}_{\mathbf{P}}$ with fresh Gaussian noise $\mathbf{e}_{\mathbf{P}}$. The pseudrandomness is argued via the followig sequence of hybrids.

  Hyb_0. The adversary is given $(\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}, \mathbf{X}, \mathbf{c}_{\mathbf{P}})$, where all components are computed honestly by the sampler.

  Hyb_1. Modify $\mathbf{c}_{\mathbf{P}}$ to be computed as $\mathbf{c}_{\mathbf{P}} = \mathbf{c}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} + f(\mathbf{x}) + \mathbf{e}_{\mathbf{P}}$. We use the observation that $\mathbf{c}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} + f(\mathbf{x}) + \mathbf{e}_{\mathbf{P}} = \mathbf{sP} - \mathbf{e}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} + \mathbf{e}_{\mathbf{P}}$, where we set $\mathbf{e}_{\mathbf{P}}$ to be large enough to smudges the error terms $\mathbf{e}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{att}}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}}$. So, we have $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.

  Hyb_2. Replace $\mathbf{c}_{\mathbf{B}}$, $\mathbf{c}_{\mathsf{att}}$, and the GSW public key $\mathbf{A}_{\mathsf{fhe}}$ with uniformly random matrices. Since the GSW public key $\mathbf{A}_{\mathsf{fhe}}$ is an LWE sample with respect to the secret key $\mathbf{s}$, and $\mathbf{c}_{\mathbf{B}}, \mathbf{c}_{\mathsf{att}}$ are BGG^+ encodings, this replacement is computationally indistinguishable by the hardness of LWE.

  Hyb_3. Replace the GSW ciphertext $\mathbf{X}$ with a uniformly random matrix. This hybrid is statistically indistinguishable from Hyb_2 using leftover hash lemma.

  Hyb_4. Finally, replace $\mathbf{c}_{\mathbf{P}}$ with a uniformly random string. This follows from the pseudorandomness of $f(\mathbf{x})$, and the fact that $\mathbf{x}$ is not used elsewhere once $\mathbf{X}$ has been replaced by a random matrix.

- *Post-condition fails.* The attack builds upon the ideas from [120] to correlate the least significant bit of $f(\mathbf{x})$ with the error term $\mathbf{e}_{\mathsf{fhe}}$ from the GSW homomorphic encryption. Specifically, they carefully design the circuit implementing f such that

$$f(\mathbf{x}) \equiv \mathbf{e}_{\mathsf{fhe}}^{\top} \pmod 2.$$

  Once this correlation is established, the attack proceeds as follows. given $\mathbf{c}_{\mathbf{B}}$, $\mathbf{c}_{\mathsf{att}}$, $\mathbf{X}$, and $\mathbf{K}$,

  – Compute $\mathbf{v} = \mathbf{c}_{\mathbf{B}}^{\top}\mathbf{K} - \mathbf{c}_{\mathsf{att}}^{\top}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} \bmod 2 = f(\mathbf{x}) + \mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K} - (\mathbf{e}_{\mathsf{fhe}}^{\top}\mathbf{R}_{\mathrm{F}} + \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}})$. In this equation, the small error terms $\mathbf{e}_{\mathsf{fhe}}$ and $\mathbf{e}_{\mathsf{att}}$ cancel each other modulo 2.

  – Find vectors $\mathbf{e}_{\mathbf{B}}$ and $\mathbf{e}_{\mathsf{att}}$ such that

$$\mathbf{e}_{\mathbf{B}}^{\top}\mathbf{K} - \mathbf{e}_{\mathsf{att}}^{\top}\mathbf{H}^{\mathrm{F}}_{\mathbf{A}_{\mathsf{att}},\mathbf{X}} = \mathbf{v} \bmod 2.$$

When the given terms are strictured one can use sufficiently many equations to recover the error terms $\mathbf{e_B}$ and $\mathbf{e}_{\mathsf{att}}$. On the other hand, had the term $\mathbf{e}_{\mathbf{B}}^{\intercal}\mathbf{K}$ been truly random, such a system of equations would not admit any solution. This leads to a distinguishing strategy.

Attack 2 [85] The attacks proposed by [85] do not directly affect our construction, but they break the evasive LWE assumption underlying the proof of our initial construction when considered for *arbitrary* samplers. Specifically, they construct a counterexample via a malicious sampler that outputs a uniformly random matrix $\mathbf{P}$, a short-norm Gaussian matrix $\mathbf{S}$, and auxiliary information $\mathsf{aux} = \mathbf{SP} - 2\mathbf{T} \bmod q$, where $\mathbf{T}$ consists of entries uniformly sampled from $[0, \lfloor q/2 \rfloor]$. They show the the pre-condition $(\mathbf{B}, \mathbf{P}, \mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) \approx_c (\mathbf{B}, \mathbf{P}, \$, \$, \mathsf{aux})$, where $\$$ represents random, holds by relying on LWE with small secrets and noise flooding. They then give a distinguisher for the post-condition, which asserts that $(\mathbf{SB} + \mathbf{E}, \mathbf{D}, \mathsf{aux} = \mathbf{SP} - 2\mathbf{T}) \approx_c (\$, \mathbf{D}, \mathsf{aux})$. The distinguisher computes $\mathbf{W} = (\mathbf{CD} - \mathsf{aux}) \bmod q \bmod 2$ and checks whether the first row of $\mathbf{W}$ lies in the row span of $\mathbf{D}$ mod 2. In the real distribution, where $\mathbf{C} = \mathbf{SB} + \mathbf{E}$, the structure of $\mathbf{ED}$ leaks through, and $\mathbf{W}$ aligns with the span of $\mathbf{D}$ mod 2. In the ideal distribution, where $\mathbf{C}$ is uniform, $\mathbf{W}$ appears uniform.

Effects on our Constructions and Assumptions. The attacks proposed by [19, 121] on the private-coin binding evasive LWE breaks our initial construction of the prFE scheme. The attacks proposed by [85] do not directly affect our constructions, but they break the evasive LWE assumption underlying the proof of our initial construction when considered for *arbitrary* samplers. In Section 5.1.4, we discuss in detail the fix to our initial construction and refining of our underlying assumption. Briefly, we follow the approach of [19] to restrict the sampler (please see Assumption 2.6 for the refined definition of evasive LWE) for which evasive LWE is conjectured true. To the best of our understanding, the original intuition of Evasive LWE still holds if malicious samplers such as the above are avoided.

7.3 PRIVATE-COIN HIDING EVASIVE LWE.

The hiding setting of the private-coin evasive LWE assumption is the strongest – here the sampler Samp is private-coin, and the matrices $\mathbf{B}, \mathbf{P}$ are partially or completely hidden from the view of the adversary. In this variant, the assumption includes the requirement that $(\mathbf{P}, \mathsf{aux}) \approx_c (\mathbf{P}+\mathbf{R}, \mathsf{aux})$ where $\mathbf{R}$ is a bounded-norm matrix. The indistinguishability between $\mathbf{P}$ and $\mathbf{P}+\mathbf{R}$ guarantees that $\mathbf{P}$ cannot be efficiently approximated, even with access to auxiliary information.

Definition 7.3 (Private-Coin Hiding Evasive LWE, [66, Definition 9])**.** Let $m, n, k, \ell > 0$ be integers and let q be a modulus. Let $\tau, \sigma, \sigma' > 0$. Let Samp be an algorithm that, on input 1^λ, outputs a matrix $\mathbf{P} \in \mathbb{Z}_q^{n\times k}$, a matrix $\mathbf{S} \in \mathbb{Z}_q^{\ell\times n}$, and auxiliary information aux. Let

$$\mathbf{B} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, \tau), (\mathbf{S}, \mathbf{P}, \mathsf{aux}) \leftarrow \mathrm{Samp}(1^\lambda),$$
$$\mathbf{E} \leftarrow \mathcal{D}_{\mathbb{Z}_q^{\ell\times m},\sigma}, \quad \mathbf{E}' \leftarrow \mathcal{D}_{\mathbb{Z}_q^{\ell\times k},\sigma'}, \mathbf{C} \leftarrow \mathbb{Z}_q^{\ell\times m}, \quad \mathbf{C}' \leftarrow \mathbb{Z}_q^{\ell\times k}, \mathbf{R} \leftarrow \mathcal{U}([\kappa])^{n\times k}.$$

For PPT distinguishers $\mathcal{A}'$, $\mathcal{A}''$, and $\mathcal{A}$, define the following functions:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Pre1}}_{\mathcal{A}'}(\lambda) &:= \mathsf{Pr}\big[\mathcal{A}'(\mathbf{SB}+\mathbf{E}, \mathbf{SP}+\mathbf{E}', \mathsf{aux}) = 1\big] \\ &\quad - \mathsf{Pr}\big[\mathcal{A}'(\mathbf{C}, \mathbf{C}', \mathsf{aux}) = 1\big], \qquad (7.5)\\ \mathsf{Adv}^{\mathsf{Pre2}}_{\mathcal{A}''}(\lambda) &:= \mathsf{Pr}\big[\mathcal{A}''(\mathbf{P}, \mathsf{aux}) = 1\big] \\ &\quad - \mathsf{Pr}\big[\mathcal{A}''(\mathbf{P}+\mathbf{R}, \mathsf{aux}) = 1\big], \qquad (7.6)\\ \mathsf{Adv}^{\mathsf{Post}}_{\mathcal{A}}(\lambda) &:= \mathsf{Pr}\big[\mathcal{A}(\mathbf{SB}+\mathbf{E}, \mathbf{K}, \mathsf{aux}) = 1\big] \\ &\quad - \mathsf{Pr}\big[\mathcal{A}(\mathbf{C}, \mathbf{K}, \mathsf{aux}) = 1\big]. \qquad (7.7)\end{aligned}$$

We say that the hiding evasive LWE assumption $\mathsf{evLWE}(q, m, n, k, \ell, \mathrm{Samp}, \kappa, \tau, \sigma, \sigma')$ holds if there exists a polynomial Q such that $\deg_\lambda(Q) \le c \cdot \deg_\lambda(|\mathrm{Samp}|)$ (for some universal constant $c > 0$), and for every PPT distinguisher $\mathcal{A}$ there exist PPT distinguishers

$\mathcal{A}'$, $\mathcal{A}''$ such that

$$\mathsf{Adv}^{\mathsf{Pre1}}_{\mathcal{A}'}(\lambda) + \mathsf{Adv}^{\mathsf{Pre2}}_{\mathcal{A}''}(\lambda) \geq \frac{\mathsf{Adv}^{\mathsf{Post}}_{\mathcal{A}}(\lambda)}{Q(\lambda)} - \mathsf{negl}(\lambda),$$

and

$$\mathrm{time}(\mathcal{A}') \leq \mathrm{time}(\mathcal{A}) \cdot Q(\lambda), \quad \mathrm{time}(\mathcal{A}'') \leq \mathrm{time}(\mathcal{A}) \cdot Q(\lambda).$$

7.3.1 Attacks and Repercussions

Attack by [164] [164] noted that the private-coin evasive LWE assumption was already prone to heuristic obfuscation-based counterexamples. The attack works as follows: Let aux be an obfuscation of the following program $\Pi_{\mathbf{P},\tau}$, which has the matrix $\mathbf{P}$ and a corresponding trapdoor τ hard-wired into it. On input $(\mathbf{C}, \mathbf{D})$, where $\mathbf{C} \in \mathbb{Z}_q^{2m \times m}$ and $\mathbf{D} \in \mathbb{Z}_q^{m \times 2m}$, the program attempts to find a short matrix $\mathbf{S}_0$ such that $\mathbf{CD} \approx \mathbf{S}_0\mathbf{P} \bmod q$. If such a short $\mathbf{S}_0$ exists, the program outputs 1; otherwise, it outputs 0.

- *Pre-Condition Holds.* Under the LWE assumption, $(\mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}')$ is pseudorandom. Thus, the pre-condition (indistinguishability from random) remains satisfied.

- *Post-Condition Broken.* The adversary receives $(\mathbf{SB}+\mathbf{E}, \mathbf{B}^{-1}(\mathbf{P}), \mathsf{aux})$. It computes $(\mathbf{SB} + \mathbf{E})\mathbf{B}^{-1}(\mathbf{P}) \approx \mathbf{SP}$, and feeds $(\mathbf{SB} + \mathbf{E}, \mathbf{B}^{-1}(\mathbf{P}))$ into the obfuscated program. Using the embedded trapdoor τ, the program verifies whether $\mathbf{SP}$ is close to some $\mathbf{S}_0\mathbf{P}$ and outputs 1 if so. For random matrices $\mathbf{C}$, the product $\mathbf{CD}$ behaves randomly, and the program outputs 0. Thus, the adversary can distinguish valid from random instances, violating the post-condition.

Attacks by [66] The obfuscation-based counterexample of [66] shows that the private-coin evasive LWE assumption can be broken even without relying on full-fledged obfuscation, using only null-iO and the hardness of LWE. In this attack, the sampler Samp generates a tall matrix $\mathbf{S} \in \mathbb{Z}_q^{m_P \times n}$ and a wide matrix $\mathbf{P} \in \mathbb{Z}_q^{n \times m_P}$, both uniformly random, and outputs auxiliary information aux containing a null-iO obfuscation of a circuit $C_{\mathbf{W}}$. Here, $\mathbf{W} = \mathbf{SP} + \mathbf{E}''$ with $\mathbf{E}''$ sampled as short noise. The obfuscated circuit $C_{\mathbf{W}}$ is designed to check, on input matrices $(\mathbf{M}_1, \mathbf{M}_2)$, whether the product $\mathbf{M}_1\mathbf{M}_2$ closely approximates the matrix $\mathbf{W}$. It outputs 1 if they are close and 0 otherwise.

- *Post-Condition Broken.* In the post-condition, the adversary receives $(\mathbf{SB} +$

$\mathbf{E}, \mathbf{B}^{-1}(\mathbf{P})$). By computing $(\mathbf{SB} + \mathbf{E})\mathbf{B}^{-1}(\mathbf{P})$, the adversary effectively recovers $\mathbf{SP}$ up to small noise. Feeding this into the obfuscated circuit results in the circuit outputting 1 with overwhelming probability. This allows the adversary to distinguish between real and random samples, thereby breaking the post-condition.

- *Pre-Condition Holds.* In the pre-condition, the auxiliary information aux becomes useless for distinguishing. By the hardness of LWE, the matrix $\mathbf{SP} + \mathbf{E}''$ is computationally indistinguishable from a uniformly random matrix $\mathbf{R}$. Consequently, the circuit $C_{\mathbf{R}}$ behaves essentially as the constant-zero function. By the security of null-iO, the obfuscation of $C_{\mathbf{R}}$ can be safely replaced with an obfuscation of a padded zero circuit, which leaks no meaningful information to the adversary.

[66] also present several other simple algebraic attacks that target the hiding variant of private-coin evasive LWE.

Attack by [124] [124] first shows that the counterexample presented by [164] is invalid. Specifically, it demonstrates that the pre-condition distributions in the VWW counterexample are actually distinguishable, contrary to their claim. Since evasive LWE only concerns the case where pre-condition distributions are indistinguishable, this renders the original VWW attack ineffective. Nevertheless, [124] constructs a new valid counterexample that breaks the underlying assumption of [164], using LWE and the existence of instance-hiding witness encryption (ihWE). Briefly, instance-hiding WE is a strengthening of witness encryption where encryptions with respect to two different unsatisfiable instances are computationally indistinguishable.

The counterexample is as follows : The sampler Samp outputs a random tall matrix $\mathbf{S}$, a wide matrix $\mathbf{P}$, and auxiliary information aux consisting of an ihWE encryption of $\mathbf{SP} + \mathbf{E}''$, for short noise $\mathbf{E}''$. In the pre-condition, by LWE hardness and the instance-hiding property of WE, the ciphertext aux hides any information about $\mathbf{S}$ or $\mathbf{P}$, and thus indistinguishability is preserved. However, in the post-condition, when the adversary receives $(\mathbf{SB}+\mathbf{E}, \mathbf{B}^{-1}(\mathbf{P}))$, it can use these to approximate $\mathbf{SP}$. Feeding this approximation into the ihWE decryption algorithm successfully recovers the hidden random string

embedded inside the ciphertext, allowing the adversary to distinguish real samples from random ones and thus break the post-condition.

Effects on our Constructions and Assumptions. We note that none of the initial versions of our constructions in Chapters 4, 5, or 6 relied on the hiding variant of the private-coin evasive LWE assumption. In all our schemes, the matrix $\mathbf{B}$ is publicly available, and the matrix $\mathbf{P}$ is publicly computable using the auxiliary information. Thus, initial versions of our constructions and the underlying assumptions *did not* get affected by the attacks discussed in this section.

7.4 CIRCULAR SMALL-SECRET EVASIVE LWE

Here we define the circular small-secret evasive LWE (evcsLWE) variant introduced by [122].

Assumption 7.1 (evcsLWE)**.** Let $\mathcal{S}(1^\lambda; \mathsf{aux})$ be an algorithm that, given randomness aux, outputs

$$\mathbf{A}_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(n+1)\times(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \bar{\mathbf{A}}' \in \mathbb{Z}_q^{n\times m'},\ \mathbf{P} \in \mathbb{Z}_q^{n\times J},\ \sigma,\ \sigma',\ \sigma_{-1},\ \sigma_{\mathsf{post}},\ \sigma_{\mathsf{pre}}$$

where $m \geq m_0(n,q)$ and $\sigma_{-1} \geq \sigma_0(n,m)$ and $\sigma_{\mathsf{post}} \geq \sigma_{\mathsf{pre}}$. Suppose

$$\bar{\mathbf{A}}_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^{n\times m}, \quad (\mathbf{B},\tau) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \quad \mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}),$$

$$\mathbf{e}_{\mathsf{fhe}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m,\ \mathbf{e}_{\mathsf{circ}} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \mathbf{e}' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma'}^{m'},\ \mathbf{e}_{\mathbf{B}} \in \mathbb{Z}^m,\ \mathbf{e}_{\mathbf{P}} \in \mathbb{Z}^J,$$

$$\delta_{\mathsf{fhe}} \leftarrow \mathbb{Z}_q^m,\ \delta_{\mathsf{circ}} \leftarrow \mathbb{Z}_q^{(m(n+1)^2\lceil\log_2 q\rceil^2+1)m},\ \delta' \leftarrow \mathbb{Z}_q^{m'},\ \delta_{\mathbf{B}} \leftarrow \mathbb{Z}_q^m,\ \delta_{\mathbf{P}} \leftarrow \mathbb{Z}_q^J,$$

$$\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^n, \quad \mathbf{s} \leftarrow (\mathbf{r}^\intercal, -1)^\intercal, \quad \mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\lceil\log_2 q\rceil m}, \quad \Delta \leftarrow \mathbb{Z}_q^{(n+1)\times(n+1)\lceil\log_2 q\rceil m},$$

$$\mathbf{S} = \begin{pmatrix} \bar{\mathbf{A}}_{\mathsf{fhe}} \\ \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}_{\mathsf{fhe}}^\intercal \end{pmatrix} \mathbf{R} - (\mathsf{bits}(\mathbf{s})) \otimes \mathbf{G}$$

In the precondition, the entries of $\mathbf{e}_{\mathbf{B}}, \mathbf{e}_{\mathbf{P}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{pre}}}$, and

$\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^\intercal_{\mathsf{fhe}}, \mathbf{S}, \\ \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{circ}} \\ \mathbf{r}^\intercal \bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal \mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B}}, \mathbf{r}^\intercal \mathbf{P} + \mathbf{e}^\intercal_{\mathbf{P}} \end{pmatrix}\right\}_{\lambda \in \mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta^\intercal_{\mathsf{fhe}}, \Delta, \\ \delta^\intercal_{\mathsf{circ}}, \\ (\delta')^\intercal, \delta^\intercal_{\mathbf{B}}, \delta^\intercal_{\mathbf{P}} \end{pmatrix}\right\}_{\lambda \in \mathbb{N}}.$$

In the postcondition, the entries of $\mathbf{e}_{\mathbf{B}}$ are independent and follow $\mathcal{D}_{\mathbb{Z},\sigma_{\mathsf{post}}}$, and $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ states that

$$\left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \mathbf{r}^\intercal \bar{\mathbf{A}}_{\mathsf{fhe}} + \mathbf{e}^\intercal_{\mathsf{fhe}}, \mathbf{S}, \\ \mathbf{s}^\intercal (\mathbf{A}_{\mathsf{circ}} - (1, \mathsf{bits}(\mathbf{S})) \otimes \mathbf{G}) + \mathbf{e}^\intercal_{\mathsf{circ}} \\ \mathbf{r}^\intercal \bar{\mathbf{A}}' + (\mathbf{e}')^\intercal, \mathbf{r}^\intercal \mathbf{B} + \mathbf{e}^\intercal_{\mathbf{B}}, \mathbf{K} \end{pmatrix}\right\}_{\lambda \in \mathbb{N}} \approx \left\{\begin{pmatrix} 1^\lambda, \mathsf{aux}, \bar{\mathbf{A}}_{\mathsf{fhe}}, \mathbf{B}, \delta^\intercal_{\mathsf{fhe}}, \Delta, \\ \delta^\intercal_{\mathsf{circ}}, \\ (\delta')^\intercal, \delta^\intercal_{\mathbf{B}}, \mathbf{K} \end{pmatrix}\right\}_{\lambda \in \mathbb{N}}.$$

The *evasive circular small-secret LWE assumption* states that $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{pre}}$ implies $\mathsf{evcsLWE}^{\mathcal{S}}_{\mathsf{post}}$ for all efficient samplers $\mathcal{S}$.

Remark 43. In our assumption (Assumption 5.18) used in constructing prFE (Section 5.3.4) , the FHE encoding $\mathbf{S}$ additionally encodes the attribute $\mathbf{x}$, whereas in above it only encode the FHE secret key $\mathbf{s}$. This is created by the difference between ABE and FE, since the attribute $\mathbf{x}$ can be public in the former and must be hidden in the latter.

Attacks and repercussions. [19] presents an attack on the circular small-secret evasive LWE variant introduced by [122], which is very similar to the attack on the private-coin binding evasive assumption. To safeguard our construction of prFE (Section 5.3.4) based on a variant of this assumption, we implement the modulus reduction fix as discussed in Section 5.1.4 and also further refine the assumption itself. In particular, we conjecture that our sampler, used in the security proof of our prFE, for natural classes of functions, lies within the secure class of samplers $\mathcal{SC}$ for which the evasive circular small-secret LWE assumption holds.

7.5 CONTRIVED FUNCTIONALITY ATTACKS.

While we prove the existence of our prFE and prIO schemes assuming LWE and (a variant) of private-coin binding of Evasive LWE, the work of [65] and [19] showed that there exist contrived, somehow "self-referential", classes of pseudorandom functionalities for which pseudorandom obfuscation and pseudorandom FE cannot exist.

Impossibility of Pseudorandom Obfuscation. Here we describe the impossibility result of [65] for pseudorandom obfuscation (PRO). Recall that in pseudorandom obfuscation (PRO), for a PRF family $\mathcal{F} = \{f_K\}_{K \in \mathcal{K}}$ and an auxiliary input function $\mathsf{aux} : \mathcal{K} \to \{0,1\}^*$, the definition requires:

- If the truth-tables $\{f_K(x)\}_{x \in \mathcal{X}}$, together with aux_K, are pseudorandom, that is,

  $$\{f_K(x)\}_{x \in \mathcal{X}}, \mathsf{aux}_K \approx_c \{u_x\}_{x \in \mathcal{X}}, \mathsf{aux}_K,$$

  where each $u_x \leftarrow \mathcal{Y}$ is uniform,

- then the obfuscations $\mathsf{PRO.Obf}(f_K)$ and $\mathsf{PRO.Obf}(f_{K'})$ must be indistinguishable in the presence of aux_K:

  $$\mathsf{PRO.Obf}(f_K), \mathsf{aux}_K \approx_c \mathsf{PRO.Obf}(f_{K'}), \mathsf{aux}_K,$$

  where K, K' are independent uniformly random keys.

The counterexample idea is to let aux_K contain a witness encryption (WE) ciphertext for the NP statement φ asserting that "there exists a small program P computing f_K." The pre-condition of PRO holds via the pseudorandomness of $\mathcal{F}$ and the semantic security of the witness encryption scheme: when the function outputs are pseudorandom (or replaced with random values), the instance φ corresponds to a "no-instance" and the WE ciphertexts become computationally indistinguishable from random strings.

However, the post-condition of PRO becomes vulnerable: the obfuscated program $\mathsf{PRO.Obf}(f_K)$ provides a short program P that serves as a valid witness for φ, allowing the adversary to decrypt the auxiliary input aux_K. In contrast, without access to $\mathsf{PRO.Obf}(f_K)$, the authors use a counting argument to show that the probability of

correctly decrypting aux_K is negligible. This yields a distinguishing attack that violates the PRO security guarantee. They further extend their counter example to the case where there is no auxilliary input.

Impossibility of Pseudorandom Functional Encryption. [19] adapts the counterexamples from [65] against the existence of PRO for all function families to show that there exist pseudorandom function families for which multi-challenge PRFE is not possible.

Repercussions and Countermeasures. As discussed in [19], this impossibility result can be seen as analogous to the known impossibilities for the random oracle model (ROM) [68] and virtual black-box (VBB) obfuscation [34]. In more detail, despite these impossibility results, the practical usefulness of ROM in cryptographic protocols, and of VBB obfuscation for restricted classes of functionalities [167, 69], is widely accepted. Similarly, the pseudorandom functionalities used in our applications – such as computing blind garbled circuits or functional encryption ciphertexts—are quite natural and do not fall prey to the contrived counterexamples constructed for PRO impossibility.

Furthermore, we note that the above counterexamples do not apply to the single-challenge pseudorandom functional encryption setting, where we set $Q_{\mathsf{msg}} = 1$ in Definition 5.13. Thus, even for general functionalities, single-challenge prFE remains feasible. Consequently, our applications stated in Theorem 5.8, Theorem 5.9, and Corollary 5.9.1 remain safe against these attacks.

7.6 OUR PERSPECTIVE.

We view these attacks as an important step forward in our understanding of evasive LWE. Note that evasive LWE should be seen as a family of assumptions parametrized by the description of the sampler and choice of error distributions, which, if invoked in full generality, is now known to be false, even in the public-coin setting. However,

as discussed in [19], the original intuition by Wee [169] and Tsabary [163] about the security of evasive LWE can be recovered by taking precautions to identify and respect a "safe zone" for evasive LWE – thus, by refining/restraining the formulation of evasive LWE, even the original construction (presented in the first online posting of this work) is secure. In addition, we modify our original construction of prFE to implement a modulus reduction step, which negates the effect of choosing contrived circuits in the construction – for this modified construction, we do not even know of any attack using malicious samplers. We also remark that in the real world, the circuit representation of functions is chosen by the key generator who is an honest party (it holds the master secret key), and we view the counter-examples emerging from such circuit choices as indicating the limits of the assumption rather than the security of the existing constructions.

While counter-examples can (and should) create concern about indiscriminately using the assumption, one school of thought might be to completely discard the assumption by labeling it false and meaningless. Another school of thought might be that counter-examples are also progress, and that they shed light on if/how the assumption should be refined to regain security, and constructions from these refined assumptions are still meaningful (especially in the absence of alternatives).

Our perspective is that disciplined new conjectures are important to make meaningful progress on problems that have resisted solutions from standard assumptions. In this context, we believe that the evasive LWE assumption has played a crucial role in enabling constructions that could not be built from plain LWE despite significant effort by the community over several years. By examining carefully the nature of counter-examples and seeing whether there are meaningful lessons to learn, we can hope to arrive at stable versions that allow to expand the boundaries of cryptography. These constructions and their proofs could yield insights that would eventually enable candidates from standard assumptions (as happened in the world of pairings, for instance) [165, 103, 18, 105, 132].

CHAPTER 8

CONCLUSIONS

This thesis explores how evasive LWE can unlock expressive and fine-grained encryption primitives that remain out of reach under standard lattice assumptions. Motivated by the need for controlled access and secure computation in modern distributed systems, we explore a suite of fine-grained encryption primitives that are efficient, quantum-resilient, and provably secure under explicit hardness assumptions.

We begin with broadcast, trace, and revoke (TR) systems, aiming to distribute encrypted content while preventing access by revoked users and supporting tracing of pirate decoders. Our constructions achieve optimal parameters and embed user identities directly into secret keys, thereby eliminating the need for index-to-identity mappings and enhancing user anonymity. In the public trace setting, we rely only on polynomial-hardness assumptions, namely compact functional encryption and key-policy attribute-based encryption (both instantiable under well-understood assumptions). In the secret trace setting, we achieve the first optimal-size TR system with embedded identities from assumptions outside Obfustopia—specifically, LWE, lockable obfuscation, and a ciphertext-policy ABE built from evasive and tensor LWE. Both variants extend to support super-polynomial revocation lists under subexponential LWE.

Next, we construct the first attribute-based encryption (ABE) scheme for Turing machines with unbounded collusion resistance from lattice-based assumptions. Our construction builds on LWE, evasive LWE, and a new circular tensor LWE assumption, allowing encryption over unbounded-length inputs and decryption time that scales with the input. As a stepping stone, we also obtain the first ciphertext-policy ABE for circuits of unbounded depth from the same assumptions.

We then present the first compact functional encryption (FE) for pseudorandom functionalities from LWE and private-coin evasive LWE,achieving compactness without relying on pairings or indistinguishability obfuscation. This construction leads to optimal key-policy and ciphertext-policy ABE for unbounded-depth circuits under the same assumptions. We further compile this construction into multi-input FE and indistinguishability obfuscation (iO) for pseudorandom functionalities.

Finally, we address recent counterexamples to evasive LWE, which show that its most general forms can fail under malicious samplers or contrived functionalities. We argue that the assumption remains meaningful when used in well-defined "safe zones".

Future Directions. This thesis opens up several promising directions for future research. Below, we highlight some key directions:

- *Quantum-Safe Public Traceable Broadcast, Tace and Revoke with Optimality.* Our construction of optimal broadcast, trace and revoke (TR) in the public trace setting currently relies on quantum-insecure assumptions – specifically, compact functional encryption schemes that are not known to be quantum-safe. An important open problem is to construct TR systems in the public trace setting under quantum-safe assumptions, ideally from lattice-based primitives alone.

- *Applications of New Tools.* We introduced several new cryptographic tools such as functional encryption for pseudorandom functionalities (prFE), multi-input functional encryption for pseudorandom functionalities (mi-prFE) and indistinguishability obfuscation for pseudorandom functionalities (PRIO). Exploring new applications of these tools in broader cryptographic contexts could further demonstrate their utility.

- *Weaker Notions of* prCT *Security.* Due to known impossibility results, our compact functional encryption for pseudorandom functionalities does not achieve prCT security in the multi-challenge setting. Investigating relaxed or alternative security notions that are still meaningful for the applications we study or the new applications would be valuable.

- *Constructions from Falsifiable Assumptions.* Several of our constructions rely on variants of evasive LWE, which are currently non-falsifiable. An important goal is to base these primitives on well-studied, falsifiable assumptions. This may involve developing new reductions from existing lattice assumptions or refining our current assumptions to admit falsifiability.

BIBLIOGRAPHY

[1] Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Cham, Switzerland, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34618-8_19

[2] Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Cham, Switzerland, Beijing, China (Apr 14–17, 2019). https://doi.org/10.1007/978-3-030-17259-6_5

[3] Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 19–23, 2018). https://doi.org/10.1007/978-3-319-96884-1_20

[4] Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Cham, Switzerland, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56620-7_21

[5] Agrawal, S.: Indistinguishability Obfuscation Without Multilinear Maps: New Techniques for Bootstrapping and Instantiation. In: EUROCRYPT. LNCS, vol. 11476, pp. 191–225. Springer (2019), https://doi.org/10.1007/978-3-030-17653-2_7

[6] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer Berlin Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010). https://doi.org/10.1007/978-3-642-13190-5_28

[7] Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010). https://doi.org/10.1007/978-3-642-14623-7_6

[8] Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Cham, Switzerland, Virtual Event (Aug 16–20,

2021). https://doi.org/10.1007/978-3-030-84259-8_8

[9] Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: TCC. LNCS, vol. 13043, pp. 224–255. Springer (2021), https://doi.org/10.1007/978-3-030-90453-1_8

[10] Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: TCC. LNCS, vol. 13747, pp. 711–740. Springer (2022), https://doi.org/10.1007/978-3-031-22318-1_25

[11] Agrawal, S., Kumari, S., Yadav, A., Yamada, S.: Broadcast, trace and revoke with optimal parameters from polynomial hardness. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 605–636. Springer (2023). https://doi.org/10.1007/978-3-031-30620-4_20

[12] Agrawal, S., Kumari, S., Yamada, S.: Compact Pseudorandom Functional Encryption from Evasive LWE. IACR Cryptology ePrint Archive (2024)

[13] Agrawal, S., Kumari, S., Yamada, S.: Attribute Based Encryption for Turing Machines from Lattices. In: Crypto (2024). https://doi.org/10.1007/978-3-031-68382-4_11

[14] Agrawal, S., Kumari, S., Yamada, S.: Pseudorandom Multi-Input Functional Encryption and Applications. IACR Cryptology ePrint Archive (2024)

[15] Agrawal, S., Maitra, M.: FE and iO for Turing machines from minimal assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 473–512. Springer, Cham, Switzerland, Panaji, India (Nov 11–14, 2018). https://doi.org/10.1007/978-3-030-03810-6_18

[16] Agrawal, S., Maitra, M., Vempati, N.S., Yamada, S.: Functional encryption for Turing machines with dynamic bounded collusion from LWE. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 239–269. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84259-8_9

[17] Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 765–797. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26951-7_26

[18] Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption for deterministic finite automata from DLIN. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 91–117. Springer, Cham,

Switzerland, Nuremberg, Germany (Dec 1–5, 2019). https://doi.org/10.1007/978-3-030-36033-7_4

[19] Agrawal, S., Modi, A., Yadav, A., Yamada, S.: Evasive LWE: Attacks, Variants & Obfustopia. IACR Cryptology ePrint Archive (2025)

[20] Agrawal, S., Pellet-Mary, A.: Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In: EUROCRYPT. LNCS, vol. 12105, pp. 110–140. Springer (2020), https://doi.org/10.1007/978-3-030-45721-1_5

[21] Agrawal, S., Pellet-Mary, A.: Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 110–140. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45721-1_5

[22] Agrawal, S., Rossi, M., Yadav, A., Yamada, S.: Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 532–564. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38551-3_17

[23] Agrawal, S., Tomida, J., Yadav, A.: Attribute-based multi-input FE (and more) for attribute-weighted sums. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 464–497. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38551-3_15

[24] Agrawal, S., Wichs, D., Yamada, S.: Optimal broadcast encryption from lwe and pairings in the standard model. In: TCC (2020). https://doi.org/10.1007/978-3-030-64375-1_6

[25] Agrawal, S., Yadav, A., Yamada, S.: Multi-input attribute based encryption and predicate encryption. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 590–621. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15802-5_21

[26] Agrawal, S., Yamada, S.: CP-ABE for circuits (and more) in the symmetric key setting. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 117–148. Springer, Cham, Switzerland, Durham, NC, USA (Nov 16–19, 2020). https://doi.org/10.1007/978-3-030-64375-1_5

[27] Agrawal, S., Yamada, S.: Optimal broadcast encryption from pairings and LWE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 13–43. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020).

https://doi.org/10.1007/978-3-030-45721-1_2

[28] Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: CRYPTO (2015). https://doi.org/10.1007/978-3-662-48000-7_32

[29] Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-47989-6_15

[30] Ananth, P., Jain, A., Sahai, A.: Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive, 2015:730 (2015)

[31] Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: Theory of Cryptography Conference. pp. 174–198. Springer (2019). https://doi.org/10.1007/978-3-030-36030-6_8

[32] Ananth, P.V., Sahai, A.: Functional encryption for Turing machines. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 125–153. Springer Berlin Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016). https://doi.org/10.1007/978-3-662-49096-9_6

[33] Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer Berlin Heidelberg, Germany, Taormina, Italy (Mar 6–9, 2011). https://doi.org/10.1007/978-3-642-19379-8_6

[34] Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_1

[35] Bellare, M., Hoang, V.T., Rogaway, P.: Adaptively secure garbling with applications to one-time programs and secure outsourcing. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 134–153. Springer Berlin Heidelberg, Germany, Beijing, China (Dec 2–6, 2012). https://doi.org/10.1007/978-3-642-34961-4_10

[36] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press, Raleigh, NC, USA (Oct 16–18, 2012). https://doi.org/10.1145/2382196.2382279

[37] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993). https://doi.org/10.1145/168588.168596

[38] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. pp. 321–334 (2007). https://doi.org/10.1109/SP.2007.11

[39] Bitansky, N., Paneth, O., Rosen, A.: On the cryptographic hardness of finding a Nash equilibrium. In: Guruswami, V. (ed.) 56th FOCS. pp. 1480–1498. IEEE Computer Society Press, Berkeley, CA, USA (Oct 17–20, 2015). https://doi.org/10.1109/FOCS.2015.94

[40] Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. FOCS (2015). https://doi.org/10.1109/FOCS.2015.20, http://eprint.iacr.org/2015/163

[41] Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. Journal of the ACM **65**(6), 39:1–39:37 (2018). https://doi.org/10.1145/3234511

[42] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_13

[43] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer Berlin Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_30

[44] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: CRYPTO (2005). https://doi.org/10.1007/11535218_16

[45] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer Berlin Heidelberg, Germany, Gold Coast, Australia (Dec 9–13, 2001). https://doi.org/10.1007/3-540-45682-1_30

[46] Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8. pp. 253–273.

Springer (2011). https://doi.org/10.1007/978-3-642-19571-6_16

[47] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer Berlin Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013). https://doi.org/10.1007/978-3-642-42045-0_15

[48] Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: CRYPTO (2014). https://doi.org/10.1007/978-3-662-44371-2_12

[49] Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Algorithmica **79**(4), 1233–1285 (2017). https://doi.org/10.1007/978-3-662-44371-2_27

[50] Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: CRYPTO (2006). https://doi.org/10.1007/11818175_17

[51] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer Berlin Heidelberg, Germany, Buenos Aires, Argentina (Mar 26–28, 2014). https://doi.org/10.1007/978-3-642-54631-0_29

[52] Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Annual Cryptology Conference. pp. 868–886. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_50

[53] Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Factoring and pairings are not necessary for io: Circular-secure LWE suffices. Cryptology ePrint Archive (2020)

[54] Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Candidate iO from homomorphic encryption schemes. Journal of Cryptology **36**(3), 27 (2023). https://doi.org/10.1007/s00145-023-09471-5

[55] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014). https://doi.org/10.1145/2090236.2090262

[56] Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In: EUROCRYPT. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_30

[57] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 575–584. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013). https://doi.org/10.1145/2488608.2488680

[58] Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 535–564. Springer, Cham, Switzerland, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_20

[59] Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. Journal of Cryptology **31**(1), 202–225 (2018). https://doi.org/10.1007/s00145-017-9255-y

[60] Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private constrained PRFs (and more) from LWE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 264–302. Springer, Cham, Switzerland, Baltimore, MD, USA (Nov 12–15, 2017). https://doi.org/10.1007/978-3-319-70500-2_10

[61] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 97–106. IEEE Computer Society Press, Palm Springs, CA, USA (Oct 22–25, 2011). https://doi.org/10.1109/FOCS.2011.12

[62] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011). https://doi.org/10.1007/978-3-642-22792-9_29

[63] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on computing **43**(2), 831–871 (2014). https://doi.org/10.1109/FOCS.2011.12

[64] Brakerski, Z., Vaikuntanathan, V.: Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. In: 13th Innovations in Theoretical Computer Science Conference (ITCS 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2022)

[65] Branco, P., Döttling, N., Jain, A., Malavolta, G., Mathialagan, S., Peters, S., Vaikuntanathan, V.: Pseudorandom Obfuscation and Applications. Cryptology ePrint Archive, Paper 2024/1742 (2024), https://eprint.iacr.org/2024/1742

[66] Brzuska, C., Ünal, A., Woo, I.K.: Evasive LWE assumptions: Definitions, Classes, and Counterexamples. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 418–449. Springer (2024). https://doi.org/10.1007/978-981-96-0894-2_14

[67] Brzuska, C., Farshim, P., Mittelbach, A.: Indistinguishability obfuscation and

UCEs: The case of computationally unpredictable sources. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 188–205. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014). https://doi.org/10.1007/978-3-662-44371-2_11

[68] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM **51**(4), 557–594 (2004). https://doi.org/10.1145/1008731.1008734

[69] Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer Berlin Heidelberg, Germany, Zurich, Switzerland (Feb 9–11, 2010). https://doi.org/10.1007/978-3-642-11799-2_5

[70] Caro, A.D., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: CRYPTO (2013). https://doi.org/10.1007/978-3-642-40084-1_29

[71] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer Berlin Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010). https://doi.org/10.1007/978-3-642-13190-5_27

[72] Chen, Y., Vaikuntanathan, V., Waters, B., Wee, H., Wichs, D.: Traitor-tracing from LWE made simple and attribute-based. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 341–369. Springer, Cham, Switzerland, Panaji, India (Nov 11–14, 2018). https://doi.org/10.1007/978-3-030-03810-6_13

[73] Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 33–65. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2017). https://doi.org/10.1007/978-3-319-63715-0_2

[74] Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: CRYPTO (1994). https://doi.org/10.1007/3-540-48658-5_25

[75] Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Cham, Switzerland, Brisbane, Queensland, Australia (Dec 2–6, 2018). https://doi.org/10.1007/978-3-030-03329-3_24

[76] Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 630–656. Springer Berlin Heidelberg, Germany,

Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-48000-7_31

[77] Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: IMA Int. Conf. (2001). https://doi.org/10.1007/3-540-45325-3_32

[78] Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC. pp. 1115–1127. ACM Press, Cambridge, MA, USA (Jun 18–21, 2016). https://doi.org/10.1145/2897518.2897651

[79] Coron, J.S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2000). https://doi.org/10.1007/3-540-44598-6_14

[80] Coron, J., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: CRYPTO (2013). https://doi.org/10.1007/978-3-642-40041-4_26

[81] Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 245–277. Springer, Cham, Switzerland, Rio de Janeiro, Brazil (Mar 25–29, 2018). https://doi.org/10.1007/978-3-319-76581-5_9

[82] Devadas, L., Quach, W., Vaikuntanathan, V., Wee, H., Wichs, D.: Succinct LWE sampling, random polynomials, and obfuscation. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19. pp. 256–287. Springer (2021). https://doi.org/10.1007/978-3-030-90453-1_9

[83] Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2017). https://doi.org/10.1007/978-3-319-63688-7_18

[84] Döttling, N., Garg, S., Hajiabadi, M., Masny, D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 3–31. Springer, Cham, Switzerland, Rio de Janeiro, Brazil (Mar 25–29, 2018). https://doi.org/10.1007/978-3-319-76578-5_1

[85] Döttling, N., Jain, A., Malavolta, G., Mathialagan, S., Vaikuntanathan, V.: Simple and General Counterexamples for Private-Coin Evasive LWE. Cryptology ePrint Archive (2025)

[86] Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO (1993). https://doi.org/10.1007/3-540-48329-2_40

[87] Francati, D., Friolo, D., Malavolta, G., Venturi, D.: Multi-key and Multi-input Predicate Encryption (for Conjunctions) from Learning with Errors. Journal of Cryptology **37**(3), 24 (2024). https://doi.org/10.1007/s00145-024-09504-7

[88] Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 513–530. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_28

[89] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: EUROCRYPT (2013). https://doi.org/10.1007/978-3-642-38348-9_1

[90] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press, Berkeley, CA, USA (Oct 26–29, 2013). https://doi.org/10.1109/FOCS.2013.13

[91] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)

[92] Garg, S., Pandey, O., Srinivasan, A.: Revisiting the cryptographic hardness of finding a nash equilibrium. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 579–604. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53008-5_20

[93] Garg, S., Pandey, O., Srinivasan, A., Zhandry, M.: Breaking the sub-exponential barrier in obfustopia. In: EUROCRYPT (2017). https://doi.org/10.1007/978-3-319-56617-7_6

[94] Garg, S., Srinivasan, A.: Single-key to multi-key functional encryption with polynomial loss. In: TCC (2016). https://doi.org/10.1007/978-3-662-53644-5_16

[95] Gay, R., Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In: EUROCRYPT (2021). https://doi.org/10.1007/978-3-030-77883-5_4

[96] Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory

of Computing. pp. 736–749 (2021). https://doi.org/10.1145/3406325.3451070

[97] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009). https://doi.org/10.1145/1536414.1536440

[98] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press, Victoria, BC, Canada (May 17–20, 2008). https://doi.org/10.1145/1374376.1374407

[99] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_5

[100] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479. IEEE Computer Society Press, Singer Island, Florida (Oct 24–26, 1984). https://doi.org/10.1109/SFCS.1984.715949

[101] Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer Berlin Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_32

[102] Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run Turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 536–553. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40084-1_30

[103] Gong, J., Waters, B., Wee, H.: ABE for DFA from k-Lin. In: Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39. pp. 732–764. Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_25

[104] Gong, J., Wee, H.: Adaptively secure ABE for DFA from k-Lin and more. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 278–308. Springer (2020). https://doi.org/10.1007/978-3-030-45727-3_10

[105] González, A., Zacharakis, A.: Fully-succinct publicly verifiable delegation from

constant-size assumptions. In: Theory of Cryptography Conference. pp. 529–557. Springer (2021). https://doi.org/10.1007/978-3-030-90459-3_18

[106] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_11

[107] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC (2013). https://doi.org/10.1145/2488608.2488677

[108] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-48000-7_25

[109] Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing. pp. 469–477 (2015). https://doi.org/10.1145/2746539.2746576

[110] Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 361–388. Springer Berlin Heidelberg, Germany, Beijing, China (Oct 31 – Nov 3, 2016). https://doi.org/10.1007/978-3-662-53644-5_14

[111] Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: Umans, C. (ed.) 58th FOCS. pp. 612–621. IEEE Computer Society Press, Berkeley, CA, USA (Oct 15–17, 2017). https://doi.org/10.1109/FOCS.2017.62

[112] Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: STOC (2018). https://doi.org/10.1145/3188745.3188844

[113] Goyal, R., Koppula, V., Waters, B.: New approaches to traitor tracing with embedded identities. In: TCC (2019). https://doi.org/10.1007/978-3-030-36033-7_6

[114] Goyal, R., Quach, W., Waters, B., Wichs, D.: Broadcast and trace with n^{ϵ} ciphertext size from standard assumptions. In: Crypto (2019), https://eprint.iacr.org/2019/636

[115] Goyal, R., Vusirikala, S., Waters, B.: Collusion resistant broadcast and trace from positional witness encryption. In: PKC (2019). https://doi.org/10.1007/

978-3-030-17259-6_1

[116] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press, Alexandria, Virginia, USA (Oct 30 – Nov 3, 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309

[117] Hofheinz, D., Jager, T., Khurana, D., Sahai, A., Waters, B., Zhandry, M.: How to generate and use universal samplers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 715–744. Springer Berlin Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53890-6_24

[118] Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40041-4_27

[119] Hohenberger, S., Sahai, A., Waters, B.: Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer Berlin Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_12

[120] Hopkins, S.B., Jain, A., Lin, H.: Counterexamples to new circular security assumptions underlying iO. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 673–700. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84245-1_23

[121] Hsieh, Y.C., Jain, A., Lin, H.: Lattice-Based Post-Quantum iO from Circular Security with Random Opening Assumption (Part II: zeroizing attacks against private-coin evasive LWE assumptions). Cryptology ePrint Archive (2025)

[122] Hsieh, Y.C., Lin, H., Luo, J.: Attribute-based encryption for circuits of unbounded depth from lattices. In: 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS). pp. 415–434. IEEE (2023). https://doi.org/10.1109/FOCS57990.2023.00031

[123] Hsieh, Y.C., Lin, H., Luo, J.: A general framework for lattice-based ABE using evasive inner-product functional encryption. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 433–464. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58723-8_15

[124] Huang, T.H., Hung, W.H., Yamada, S.: A Note on Obfuscation-based Attacks on Private-coin Evasive LWE. Cryptology ePrint Archive (2025)

[125] Hubácek, P., Yogev, E.: Hardness of continuous local search: Query complexity and cryptographic lower bounds. In: Klein, P.N. (ed.) 28th SODA. pp. 1352–1371. ACM-SIAM, Barcelona, Spain (Jan 16–19, 2017). https://doi.org/10.1137/1.9781611974782.88

[126] Jain, A., Lin, H., Lou, P., Sahai, A.: Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum iO. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part I. LNCS, vol. 14004, pp. 205–235. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30545-0_8

[127] Jain, A., Lin, H., Luo, J.: On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 479–510. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30620-4_16

[128] Jain, A., Lin, H., Matt, C., Sahai, A.: How to Leverage Hardness of Constant-Degree Expanding Polynomials over R to build iO. In: EUROCRYPT (2019). https://doi.org/10.1007/978-3-030-17653-2_9

[129] Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Khuller, S., Williams, V.V. (eds.) 53rd ACM STOC. pp. 60–73. ACM Press, Virtual Event, Italy (Jun 21–25, 2021). https://doi.org/10.1145/3406325.3451093

[130] Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 670–699. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-06944-4_23

[131] Jin, Z., Kalai, Y.T., Lombardi, A., Mathialagan, S.: Universal SNARGs for NP from Proofs of Correctness. Cryptology ePrint Archive, Paper 2024/2015 (2024), https://eprint.iacr.org/2024/2015

[132] Kalai, Y., Lombardi, A., Vaikuntanathan, V., Wichs, D.: Boosting batch arguments and RAM delegation. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing. pp. 1545–1552 (2023). https://doi.org/10.1145/3564246.3585200

[133] Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Journal of cryptology **26**, 191–224

(2013). https://doi.org/10.1007/s00145-012-9119-4

[134] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 669–684. ACM Press, Berlin, Germany (Nov 4–8, 2013). https://doi.org/10.1145/2508859.2516668

[135] Kim, S., Wu, D.J.: Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In: ASIACRYPT (2020). https://doi.org/10.1007/978-3-030-64834-3_3

[136] Kitagawa, F., Nishimaki, R., Tanaka, K., Yamakawa, T.: Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 521–551. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_17

[137] Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 32–61. Springer, Cham, Switzerland, Rio de Janeiro, Brazil (Mar 25–29, 2018). https://doi.org/10.1007/978-3-319-76578-5_2

[138] Komargodski, I., Segev, G.: From minicrypt to obfustopia via private-key functional encryption. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 122–151. Springer, Cham, Switzerland, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56620-7_5

[139] Komargodski, I., Segev, G.: From minicrypt to obfustopia via private-key functional encryption. Journal of Cryptology **33**(2), 406–458 (2020). https://doi.org/10.1007/978-3-319-56620-7_5

[140] Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for Turing machines with unbounded memory. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 419–428. ACM Press, Portland, OR, USA (Jun 14–17, 2015). https://doi.org/10.1145/2746539.2746614

[141] Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Cham, Switzerland, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34618-8_18

[142] Lin, H., Luo, J.: Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 247–277. Springer, Cham, Switzerland, Zagreb, Croatia

(May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_9

[143] Lin, H., Luo, J.: Succinct and adaptively secure ABE for ABP from k-Lin. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 437–466. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64840-4_15

[144] Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Public-Key Cryptography–PKC 2016, pp. 447–462. Springer (2016). https://doi.org/10.1007/978-3-662-49387-8_17

[145] Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology **22**(2), 161–188 (Apr 2009). https://doi.org/10.1007/s00145-008-9036-8

[146] Mathialagan, S., Peters, S., Vaikuntanathan, V.: Adaptively Sound Zero-Knowledge Snarks for UP. In: Annual International Cryptology Conference. pp. 38–71. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_2

[147] Mathialagan, S., Peters, S., Vaikuntanathan, V.: Using PRTT Obfuscation for Unlevelled FHE. Personal Communication (2024)

[148] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer Berlin Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012). https://doi.org/10.1007/978-3-642-29011-4_41

[149] Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer Berlin Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_26

[150] Naor, M., Pinkas, B.: Efficient trace and revoke schemes. International Journal of Information Security **9**(6), 411–424 (2010). https://doi.org/10.1007/s10207-010-0121-2

[151] Nishimaki, R., Wichs, D., Zhandry, M.: Anonymous traitor tracing: How to embed arbitrary information in a key. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 388–419. Springer Berlin Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_14

[152] Pippenger, N., Fischer, M.J.: Relations among complexity measures. Journal of the ACM (JACM) **26**(2) (1979). https://doi.org/10.1145/322123.322138

[153] Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS).

pp. 859–870. IEEE (2018). https://doi.org/10.1109/FOCS.2018.00086

[154] Ragavan, S., Vafa, N., Vaikuntanathan, V.: Indistinguishability Obfuscation from Bilinear Maps and LPN Variants. Cryptology ePrint Archive (2024)

[155] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6), 34:1–34:40 (2009). https://doi.org/10.1145/1568318.1568324

[156] Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 463–472. ACM Press, Chicago, Illinois, USA (Oct 4–8, 2010). https://doi.org/10.1145/1866307.1866359

[157] Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: EUROCRYPT (2005). https://doi.org/10.1007/11426639_27

[158] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014). https://doi.org/10.1145/2591796.2591825

[159] Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. SCIS 2000 (2000)

[160] Takashima, K.: Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In: SCN (2014). https://doi.org/10.1007/978-3-319-10879-7_17

[161] Tomida, J.: Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 459–488. Springer, Cham, Switzerland, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34618-8_16

[162] Tsabary, R.: Fully secure attribute-based encryption for t-CNF from LWE. In: CRYPTO (2019). https://doi.org/10.1007/978-3-030-26948-7_3

[163] Tsabary, R.: Candidate witness encryption from lattice techniques. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 535–559. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15802-5_19

[164] Vaikuntanathan, V., Wee, H., Wichs, D.: Witness encryption and null-IO from evasive LWE. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 195–221. Springer, Cham, Switzerland, Taipei, Taiwan (Dec 5–9, 2022). https://doi.org/10.1007/978-3-031-22963-3_7

[165] Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_14

[166] Waters, B., Wee, H., Wu, D.J.: Multi-authority ABE from lattices without random oracles. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 651–679. Springer, Cham, Switzerland, Chicago, IL, USA (Nov 7–10, 2022). https://doi.org/10.1007/978-3-031-22318-1_23

[167] Wee, H.: On obfuscating point functions. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 523–532. ACM Press, Baltimore, MA, USA (May 22–24, 2005). https://doi.org/10.1145/1060590.1060669

[168] Wee, H.: Broadcast encryption with size $N^{1/3}$ and more from k-lin. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 155–178. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84259-8_6

[169] Wee, H.: Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 217–241. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_8

[170] Wee, H.: Circuit abe with poly(depth, λ)-sized ciphertexts and keys from lattices. In: Crypto (2024). https://doi.org/10.1007/978-3-031-68382-4_6

[171] Wee, H., Wichs, D.: Candidate obfuscation via oblivious LWE sampling. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 127–156. Springer, Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77883-5_5

[172] Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: Umans, C. (ed.) 58th FOCS. pp. 600–611. IEEE Computer Society Press, Berkeley, CA, USA (Oct 15–17, 2017). https://doi.org/10.1109/FOCS.2017.61

[173] Yao, A.C.C.: Protocols for secure computations extended abstract. In: 23rd FOCS. vol. 28 (1982). https://doi.org/10.1109/SFCS.1982.38

[174] Zhandry, M.: New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In: CRYPTO (2020). https://doi.org/10.1007/978-3-030-56784-2_22

CURRICULUM VITAE

NAME Simran Kumari

DATE OF BIRTH 08 October 1996

EDUCATION QUALIFICATIONS

2019 **M.Sc.**

Institution	National Institute of Technology, Warangal
Specialization	Mathematics and Scientific Computing

2017 **B.Sc.**

Institution	Delhi University, New Delhi
Specialization	Mathematics

DOCTORAL COMMITTEE

Chairperson	Prof. Krishna Moorthy Sivalingam CSE Department, IIT Madras
Guide	Prof. Shweta Agrawal CSE Department, IIT Madras
Members	Prof. John Augustine CSE Department, IIT Madras
	Prof. Aishwarya Thiruvengadam CSE Department, IIT Madras
	Prof. Venkata Koppula CSE Department, IIT Delhi