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

Building Cryptography for Distributed Data and Authority

A Thesis

Submitted by

ANSHU YADAV

For the award of the degree

Of

DOCTOR OF PHILOSOPHY

May 2023

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

Building Cryptography for Distributed Data and Authority

A Thesis

Submitted by

ANSHU YADAV

For the award of the degree

Of

DOCTOR OF PHILOSOPHY

May 2023

To

my parents, and

sweet memories of chhota bhaiya

THESIS CERTIFICATE

This is to undertake that the Thesis titled **BUILDING CRYPTOGRAPHY FOR DISTRIBUTED DATA AND AUTHORITY**, 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.

Anshu Yadav

Anshu Yadav

Chennai 600036

Date: May 25, 2023

Shweta Agrawal.

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

LIST OF PUBLICATIONS

I. PUBLICATIONS FROM THE THESIS

1. Shweta Agrawal, Junichi Tomida, Anshu Yadav. Attribute-Based Multi-Input FE (and more) for Attribute-Weighted Sums. In CRYPTO, 2023. (To appear).

2. Shweta Agrawal, Mélissa Rossi, Anshu Yadav, Shota Yamada. Constant Input Attribute Based (and Predicate) Encryption from Evasive and Tensor LWE. In CRYPTO, 2023. (To appear).

3. Shweta Agrawal, Anshu Yadav, Shota Yamada. Multi-input Attribute Based and Predicate Encryption. In CRYPTO, volume 13507 of LNCS, pages 590-621, Springer, 2022. https://doi.org/10.1007/978-3-031-15802-5_21.

4. Shweta Agrawal, Damien Stehlé, Anshu Yadav. Round-Optimal Lattice-Based Threshold Signatures, Revisited. In ICALP, pages, volume 229 of LIPIcs, pages 8:1-8:20, Dagstuhl Publishing, 2022. https://doi.org/10.4230/LIPIcs.ICALP.2022.8

5. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, Anshu Yadav. Practical, Round-Optimal, Lattice-Based Blind Signatures, In ACM-CCS, pages 39–53, ACM, 2022. https://doi.org/10.1145/3548606.3560650.

II. OTHER PUBLICATIONS

1. Shweta Agrawal, Simran Kumari, Anshu Yadav, Shota Yamada. Trace and Revoke with Optimal Parameters from Polynomial Hardness. In EUROCRYPT, volume 14006 of LNCS, pages 605-636. Springer, 2023. https://doi.org/10.1007/978-3-031-30620-4_20

ACKNOWLEDGEMENTS

I take this opportunity to thank everyone who has contributed to my journey for Ph.D. First and foremost, I express my deepest gratitude to my advisor Prof. Shweta Agrawal for her immense support and guidance throughout this journey. This work would not have been possible without her active involvement at every stage. I thank her for believing in me at the time when I was going through a low phase in my career. I feel extremely blessed that I got the opportunity to work with her. Her constant encouragement has always kept me motivated. Even beyond research, her advice towards life, in general, has been invaluable. She has always been there as a strong support in every part of my PhD journey both at the professional and the personal level. I am deeply grateful and will always be indebted to her for everything that she has done for me.

I sincerely thank all my co-authors, Prof. Shweta, Dr. Elena Kirshanova from TII Abu Dhabi, Simran Kumari from IIT Madras, Dr. Mélissa Rossi from ANSSI Paris, Prof. Damien Stehlé from ENS Lyon, Dr. Junichi Tomida from NTT Japan and Dr. Shota Yamada from AIST Tokyo, for all the interesting discussions. I got to learn a lot while working with them. I thank Prof. Damien and Dr. Shota, also for their guidance in writing papers and preparing conference talks. I am also thankful to Prof. Jung Hee Cheon and Hyeongmin Choe from Seoul National University, for insightful discussions on a project that unfortunately could not be successful. I want to thank Prof. Damien and Dr. Alain Passelègue for inviting me to ENS Lyon for a short visit and for having fruitful discussions.

I want to thank my current Doctoral Committee members - Prof. Balaram Ravindran, Prof. John Augustine, Dr. Nishanth Chandran, and Prof. Aishwarya Thiruvengadam, as well as the previous members - Prof. Narayanaswamy N. S., Prof. Chester Rebeiro, and Prof. Jayalal Sarma, for their questions, comments, and suggestions during my seminars. I am thankful to all the anonymous reviewers of our papers for their useful comments.

I also want to thankfully acknowledge Microsoft Research and Google Research for supporting me with travel grants for presenting our work at ICALP 2022 and ACM CCS 2022, respectively.

I would also like to extend my thanks to the faculties and staff from the Computer Science Department at IIT Madras. Of special mention are Prof. Shweta and Prof. Jayalal for their amazing style of teaching with lots of clarity that helped me understand many difficult concepts with much ease. It was a very good experience sitting through their lectures.

I want to thank all the members of the Cryptography Cybersecurity and Distributed Trust (CCD) lab and the previous Theoretical Computer Science (TCS) lab for interesting discussions during talks and seminars, and also for organizing various fun events. I particularly want to thank Monosij Maitra, who was a senior Ph.D. student with Shweta. He has been very helpful during my Ph.D. - in solving my research-related doubts as well as many other things. I enjoyed having technical discussions with him during my Ph.D. I would also like to specially mention Simran for all the long discussions, especially while working on a joint project and for her company in late-night snacking at Usha cafe.

Finally, I am thankful to everyone in my family who contributed in my Ph.D. journey in different ways. I express my deepest gratitude towards my parents and my brother, my cousin brother, and my sister for their immense love and strong support. My interest in mathematics developed due to my father because of his wonderful style of teaching during my early schooling. He taught me to think logically and to understand different concepts using relatable examples from daily life. I have no words to thank my parents, for always believing in me and giving me complete freedom and full support in choosing my career path and every other choice. I will always be indebted to them for all the love and care and the sacrifices that they have made. I am extremely thankful to my brother and sister for being an incredible support system. My brother has always been

the guiding force throughout my journey so far. I thank my sister for always being very caring and encouraging. My entry into the world of computer science was guided by my cousin brother. I want to express my special thanks to him for all his help and support. I must also thank my loving nephews and niece for all the interesting conversations that I had with them. Talking to them or just about them would help me forget all my problems. I thank them for being a part of my Ph.D. journey in the most beautiful way.

ABSTRACT

KEYWORDS multi-input predicate encryption; multi-input attribute based encryption; attribute weighted sums; attribute based multi-input functional encrytion; multi-client functional encryption; dynamic decentralized functional encryption; blind signature; one-more-isis; threshold signature

In the modern era of advanced digital technologies, there exists a plethora of online applications on the internet including e-commerce, net banking, online games and movies, and many more. These applications generate huge amount of data which includes sensitive information of their users, like bank account details, credit card numbers, and personal details like date of birth, mobile number, address, and such others. This poses new challenges in ensuring data privacy and secrecy without compromising with the enormous opportunities of performing useful studies on these data. For example, consider a researcher who wants to study the correlation between hypertension and the profession of patients in the age group 30-50 years and hence needs access to records of patients who fall in this category. The hospital would want to encrypt the records of patients and provide a decrypting key to the researcher that allows access to only the relevant records and nothing else. Traditional methods like standard public key encryption (PKE) do not suffice because anyone having the decryption key gets unrestricted access to all the records of all the patients. While what we want is a tailored key that lets the researcher access only the relevant data related only to her research. Advanced tools like functional encryption (FE), attribute based encryption (ABE), and predicate encryption (PE) give us such guarantees but are generally defined in a single input setting where the data is assumed to be held by a single party. In practice, data related to a single entity is generated independently in parts at different locations. In such a scenario, we would want to have the same security guarantees as if the data were generated from a single source. Furthermore, for better security in cryptographic primitives, it is often desired

that the authority is distributed among multiple parties so that there is no single point of security failure. For example, a user of an online application may want to distribute his/her key on multiple devices, like smartphone and tablet so that the security is ensured unless all the devices are compromised.

Motivated by these observations, we study different cryptographic primitives in a distributed setup. Our work can be broadly viewed in two parts - in the first part, we study such primitives, where the data to be encrypted is generated in a distributed manner. In the second part, we focus on primitives with distributed authority.

In the first part, we study ABE, PE, and FE in distributed settings. We initiate the study of multi-input PE (MIPE) and further develop multi-input ABE (MIABE). We define MIABE and MIPE in symmetric key setting and formalize the security in standard indistinguishability (IND) based paradigm against unbounded collusions. We provide the first construction for 2ABE for NC_1 using learning with errors (LWE) and pairings and give a generic compiler that for any constant k, transforms any k-input ABE to k-input PE. We also provide the first construction of MIABE for NC_1 for *any* constant arity from evasive LWE assumption, recently introduced independently by Wee (Eurocrypt 2022) and Tsabary (Crypto 2022). We extend our construction to support functions in P by using evasive and suitable strengthening of tensor LWE introduced by Wee (Eurocrypt 2022). By combining our compiler with our results for MIABE, we get 2PE for NC_1 using LWE and pairings and constant-arity PE for NC_1 and P from lattice based assumptions of evasive and tensor LWE that are conjectured to be post-quantum secure.

Furthermore, we extend FE for attribute weighted sums (AWS), recently given by Abdalla et al. (Crypto 2020), where encryption takes N (unbounded) (public-private) attribute-value pairs $\{\mathbf{x}_i, \mathbf{z}_i\}_{i\in[N]}$, the secret key is given for an arithmetic branching programs f, and the decryption returns the weighted sum $\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i$, to the significantly more challenging multi-party setting and provide the first construction for *attribute-based*

multi-input FE (MIFE) supporting AWS. We also provide the first constructions of multi-client FE (MCFE) and dynamic decentralized FE (DDFE) for AWS. Previously, these primitives were known only for linear functions (or inner products) [Abdalla *et al.* (Asiacrypt 2020), Chotard *et al.* (ePrint 2018), Abdalla *et al.* (Asiacrypt 2019), and Chotard *et al.* (Crypto 2020)].

In the second part, we study advanced digital signature schemes - threshold signatures and blind signatures. Both the primitives find applications in modern applications like cryptocurrencies, e-voting, blockchains, etc. Threshold signature is a digital signature scheme, where the signing power is distributed among N signers such that at least some threshold t number of signers are needed to generate a valid signature. We improve the only lattice based construction for t-out-of-N access structure with round optimality by Boneh et al. in terms of efficiency and security. We reduce the signature size from $\tilde{O}(\lambda^3)$ to $\tilde{O}(\lambda)$, where λ is the security parameter. Boneh et al's construction achieve only selective security, where all the corrupt parties are declared in the beginning. We give two constructions that achieve (i) partial adaptivity - which allows corruption at any time, but all the corruptions must be declared together, and (ii) full adaptivity - which allows corruption at any time in any order.

In blind signatures, there are two parties involved - the user and the signer. The user $\mathcal{U}$, holding a public key and message, may request a signature from the signer $\mathcal{S}$, holding a signing key, such that the signer is not able to link a message-signature pair with a protocol execution, and the user is not able to forge signatures even after multiple interactions with the signer. We construct the first overall practical round optimal lattice based blind signature supporting an unbounded number of signature queries. We provide a detailed estimate of parameters achieved – we obtain a signature of size slightly above 45KB, for a core-SVP hardness of 109 bits. All the run-times are also very small. Its security stems from a new and arguably natural assumption that we introduce. To gain confidence in our assumption, we provide a detailed analysis of diverse attack strategies.

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**
$\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$	Discrete Gaussian distribution over lattice Λ with std. deviation σ and mean **c**
$\mathcal{D}_{\Lambda,\sigma}$	Discrete Gaussian distribution over lattice Λ with std. deviation σ and mean **0**
$\mathcal{P}(S)$	Power set of set S
$\log x$	Base 2 logarithm of x
$\text{negl}(\lambda)$	A negligible function of λ
$\|\mathbf{v}\|$	ℓ_2 norm of vector **v**
$\|\mathbf{v}\|_\infty$	ℓ_∞ norm of vector **v**
$\text{poly}(\lambda)$	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**)
$D_1 \approx_c D_1$	Distributions D_1 and D_2 are computationally indistinguishable
$D_1 \approx_s D_1$	Distributions D_1 and D_2 are statistically indistinguishable
$D_1 \equiv D_1$	Distributions D_1 and D_2 are perfectly indistinguishable

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The developments in the field of information technology have facilitated easy storage and access to data generated at different locations. This has further facilitated the performing of different kinds of studies on these data, like medical research, market analysis, etc. These data generally contain many sensitive information about the users and therefore, it poses cryptographic challenges to provide secure and controlled access to it so that only authorized users can access only the relevant part in an efficient manner. For example, consider a medical researcher who wants to access medical records of patients from a hospital to study the efficacy of certain medicine in patients in the age group '60-80' years having covid and asthma. In this case, we would like to provide the researcher, restricted access to the records of only those patients who fall in the said group. We would like to use cryptographic tools to provide this facility. One possible way for achieving this could be that the hospital encrypts and places the medical records of all its patients in a public cloud and provides a *specialized* key to the researcher that allows her to access the records of only those patients who are relevant to her study. Clearly, traditional methods, like public key encryption schemes are not enough since anyone having the decryption key gets unrestricted access to the entire database. There exist advanced encryption schemes like attribute based encryption (ABE), functional encryption (FE) that provide such functionality and has seen remarkable progress in recent years [SW05; GPSW06; CW14; AFV11; LW11; LW12; Wat12; GVW13; Wee14; Att14; BGG$^+$14; GVW15; GV15; BV16; BW07; SBC$^+$07; GVW15; Lin16; Lin17a; LV16; Agr19; AJL$^+$19; JLMS19; GJLS21; JLS21; JLS22]. In ABE, a message m is encrypted with respect to a public attribute x (for. e.g. age, gender, and disease can be attributes of a patient corresponding to which his/her records are encrypted), and the

decryption key is generated corresponding to a function f such that the decryption is possible iff $f(x) = 1$. In the stronger notion of predicate encryption (PE), the attributes are also hidden by the ciphertext. In case of FE, the encryption is done for an input x and the key is generated for a function f such that on decryption only $f(x)$ is received and nothing else.

Note that the above solutions implicitly assume a single source of input - ABE/PE assume that the message m is generated at a single source and is encrypted with respect to a single attribute x. Similarly, in the case of FE, the input x is assumed to be generated and encrypted at a single source. However, we observe that different parts of a single logical data may be generated independently in multiple locations and it is often pertinent to consider the input as a concatenation of these correlated partial inputs. For instance, a patient is likely to visit different medical centers for the treatment of different diseases and his/her overall medical record is a concatenation of the medical data generated at different centers. Similarly, a company is likely to conduct research and development related to a given technology in different locations but the complete data pertaining to that technology is a concatenation of these. Now, it may be desirable to provide restricted access to relevant consumers of the data, exactly as before, but with the caveat that the input was generated in a distributed manner and is encoded in multiple ciphertexts.

One trivial solution to the above situation can be that the records generated at each department/center are collected at a central repository where they can then be treated as a single data. But it poses two challenges: (i) how to transfer the data to the central repository? If we send the data in plain, it raises security issues. Alternatively, each center can first encrypt the locally generated data with the repository's public key and then send this encrypted data. The repository can use its secret key to decrypt and then re-encrypt using ABE/PE or FE. However, this method is too cumbersome and wasteful. (ii) One may desire to use a public cloud for storage in which case the solution proposed in (i) will be doubly inefficient.

Instead, what we would actually want is to be able to provide the same functionality and security guarantees as in the single input setting in an efficient manner. In particular, we would want that each part of the data is independently encrypted at the source and placed on the cloud directly. For decryption, we would want to simply concatenate the independently generated ciphertexts into a single ciphertext and perform decryption as if the ciphertext is generated from a single source.

To further enhance the security against secret key getting compromised, in any cryptographic primitive, it is desired that the trust is distributed among multiple parties so that there is no single point of security failure. For example, consider a user who uses Google Pay on his/her phone. Now, if someone gets access to his/her phone, then this will completely break the security and the money will be lost. Such an attack can be addressed by distributing the key for GPay on multiple devices, like the user's phone, laptop, tablet, etc. such that access to at least two or more devices is needed to make any transaction. This reduces the security risk due to key compromise since probability of compromising a single key is higher than compromising two or more keys. Another familiar example is that of two-step authentication used by several online portals, like net banking, Gmail etc., which requires both the password and the OTP sent on the user's phone for login. This, again, ensures that unless both the login password and the users' phone are compromised, the user's account remains secure.

Motivated by these observations, the central theme of this thesis is to study various cryptographic primitives in distributed setups. In particular, we study multi-input ABE/PE, multi-party FE, threshold signatures, and blind signatures. Our contributions include formalizing the definitions for MIABE and MIPE and providing their first constructions under various assumptions. We also provide the first constructions for different forms of multi-party FE for attribute weighted sums (AWS) functionality. We improve the only known construction for round-optimal threshold signatures from lattice based assumptions in terms of efficiency and security properties. Lattice based

assumptions have the advantage of conjectured post quantum security. Finally, we give the first overall practical and round-optimal construction of blind signatures from a lattice based assumption that we introduce. We also study our assumption from different possible attack strategies to gain more confidence in its hardness. Below we give a brief overview of our thesis.

1.2 OVERVIEW OF THE THESIS

The thesis can be broadly viewed in two parts. In the first part, we study cryptographic primitives in multi-input setting where data is generated in a distributed way. The second part focuses on primitives supporting distributed authority.

1.2.1 Distributed Data

In the first part, we study ABE, PE, and FE in multi-input setting. We begin with initiating the study of multi-input predicate encryptions and extend the study of multi-input attribute based encryption. Even though both ABE and PE have been widely studied and have seen remarkable progress [SW05; GPSW06; BW07; KSW08; LOS+10; OT10; OT12; CW14; AFV11; LW11; LW12; Wat12; GVW13; Wee14; Att14; BGG+14; GVW15; GV15; BV16; BW07; SBC+07; KSW08; GVW15], all the known constructions, prior to our work, were limited to single input setting. While the more realistic setting of multi-inputs has been studied for other related primitives like fully homomorphic encryption (FHE) and functional encryption (FE) [LATV12; CM15; MW16], [GGG+14; AJ15; AGRW17; DOT18; ACF+18; CDG+18a; Tom19; ABKW19; ABG19; LT19; AGT21a], in case of ABE and PE, there has been no significant work. While the idea of ABE was introduced in the context of constructing witness encryption (WE) [BJK+18], it was not formally defined and no construction was given. In the case of PE there has been no prior study at all.

We argue that the multi-input setting is important even in the context of ABE and PE and generalizing these primitives to support multiple sources enables a host of new

and natural applications. For concreteness, we continue with the previous example of hospital in more detail. As before, consider a doctor who wants to understand the relation between Covid and other medical conditions such as asthma or cancer, each of which are treated at different locations. The records of a given patient are encrypted independently and stored in a central repository, and the doctor can be given a key that filters stored (encrypted) records according to criteria such as *condition = 'Covid' and condition = 'asthma' and age group ='60-80'* and enables decryption of these. Similarly, a company (e.g. IBM) that conducts research in quantum technologies is likely to have different teams for theoretical and experimental research, and these teams are likely to work in different locations – indeed, even members of the same team may not be co-located. Data pertaining to the research could be stored encrypted in a central location where individual ciphertexts are generated independently, and the company may desire to give restricted access to a patent office. As a third example, a company may have been sued for some malpractice, and the data pertinent to the case could span multiple locations. Now the company may wish to provide restricted access to a law firm that enables decryption only of the data pertaining to the lawsuit, encrypted independently by multiple sources.

Multi-input attribute based encryption (MIABE) or predicate encryption (MIPE) arise as natural fits to the above applications. Similarly to the single input case, the secret key corresponds to a function f but the arity of this function can now be $k > 1$ – we may have k ciphertexts generated independently encoding $(\mathbf{x}_i, m_i)_{i\in[k]}$, and decryption reveals $(m_1, \ldots, m_k)$ if and only if $f(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 1$. Indeed, any application of single input ABE and PE where the underlying data is generated in multiple locations and is correlated in meaningful ways can benefit from the abstraction of multi-input ABE and PE.

We begin with formalizing the definition of MIPE and MIABE and their security under unbounded collusions. We construct two-input ABE (2ABE) using LWE and pairings. Our construction leverages the surprising connection that we observe between the

techniques developed in the context of succinct CPABE to a seemingly unrelated setting of multi-input kpABE. We then give a generic compiler that for a constant k, transforms any k-input ABE (kABE) to k-input PE (kPE). Our compiler uses the tool of Lockable Obfuscation (LO) defined by [GKW17; WZ17] for translating single input ABE to PE in much more challenging setting of MIABE to MIPE. We note that LO can be instantiated from LWE. Using our compiler with our results for kABE, we get constructions for 2PE. We then give constructions for 3ABE for NC_1 and 2ABE for polynomial circuits by leveraging the techniques developed in [BV22] in context of constructing succinct ciphertext policy ABE (CPABE) for polynomial circuits. Similar to [BV22], these constructions are heuristic.

We then extend our result for MIABE to support any constant arity, k using recently introduced lattice based assumptions of evasive and tensor LWE [Wee22; Tsa22]. We provide construction of MIABE for the function class NC_1 for any constant arity from evasive LWE assumption. We further extend our construction to support the function class P by using evasive and a suitable strengthening of tensor LWE. For the special case of arity 2, we need only the assumptions introduced by Wee, i.e. evasive LWE for NC_1 and evasive plus tensor LWE for P (i.e. we do not need to strengthen tensor LWE).

We further continue with the same thread of distributed data where we now focus on constructing FE in the decentralized setting.

While initially defined and constructed in the single input setting, i.e. with only one encryptor and one key generator, FE soon began to be generalized to distributed settings to capture the decentralized nature of both data and authority in the modern world. Computation on encrypted data generated independently at multiple sources, with fine-grained control on which data may be combined and with secret keys supporting decryption of meaningful aggregate functionalities, holds the promise of making FE much more relevant for real-world applications. These generalizations took different

forms, from multi-input FE (miFE) [GGG+14] to multi-client FE (MCFE) [CDG+18a] to dynamic decentralized FE (DDFE) [CDSG+20] and such others [ACF+20]. These generalizations were captured via the abstraction of multi-party FE (MPFE) [AGT21b], which sought to unify these different notions in a single framework. In this thesis, we study these primitives for attribute weighted sums functionality (AWS).

The Attribute-Weighted Sums Functionality: Recently, Abdalla, Gong, and Wee [AGW20] introduced the functionality of *Attribute-Weighted Sums* (AWS) which supports computation of aggregate statistics on encrypted databases. Concretely, consider a database with N attribute-value pairs $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ where $\mathbf{x}_i$ is a public attribute associated with user i, and $\mathbf{z}_i$ is private. Given a function f, the AWS functionality on input $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ is defined as

$$\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i.$$

The AWS functionality is very natural, and Abdalla, Gong, and Wee suggested several compelling applications for it – for example, when f is a Boolean predicate then AWS can capture (i) the average salaries of minority groups holding a particular job title – here, $\mathbf{z}_i$ represents salary, while $f(\mathbf{x}_i)$ tests for membership in the minority group, (ii) approval ratings of an election candidate amongst specific demographic groups in a particular state – here, $\mathbf{z}_i$ is the rating, while $f(\mathbf{x}_i)$ computes membership in said group. Similarly, when $\mathbf{z}_i$ is Boolean, AWS can capture the average age of smokers with lung cancer, where $\mathbf{z}_i$ is lung cancer and f computes the average age.

Distributing the Data: Abdalla et al's construction work in the single input setting, where all the N attribute-value pairs are held by a single party who performs the encryption. We argue that for several applications of AWS, including the motivating examples provided by [AGW20], the data $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ is likely to be distributed across

multiple sources which must compute ciphertexts independently. Concretely, in the example of computing average salaries of minority groups holding a particular job title, the data about the individuals would be generated across organizations, which are unlikely to even be in the same location. Similarly, when we compute whether a user is in a specific demographic group in a particular state, it is natural that user data would be distributed across different states, indeed even across different cities in a given state. In the third example, data about patients with lung cancer will naturally be generated and maintained at different hospitals that offer treatment for lung cancer, which would again be distributed geographically.

Thus, to capture data generation by independent sources, we extend FE for AWS to the multi-party setting. Concretely, we focus on the construction of the following primitives for AWS functionality:

1. *Multi-Input FE (MIFE):* The primitive of multi-input FE (MIFE) [GGG+14] allows the input to a function to be distributed among multiple (say n) parties. In more detail, the i^{th} party encrypts its input $\mathbf{z}_i$ to obtain ct_i, a key authority holding a master secret generates a functional key sk_f and these enable the decryptor to compute $f(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ and nothing else.

   We consider a further generalization of MIFE, namely *attribute-based* MIFE introduced by Abdalla *et al.*[ACGU20], which enables greater control on the leakage inherent by the functionality of MIFE, making it more suitable for practical applications. In an AB-MIFE for some functionality f, an attribute $\mathbf{y}_i$ is associated with a ciphertext for slot i, in addition to an input $\mathbf{z}_i$. The secret key is associated with an access control policy g in addition to the function input $\mathbf{c}$. Decryption first checks if $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = 1$, and if so, it computes the MIFE functionality $f(\{\mathbf{z}_i\}, \mathbf{c})$.

2. *Multi-Client FE (MCFE):* MCFE [GGG+14; CDG+18a; CDG+18b] is a generalization of MIFE. In MCFE, the inputs $\mathbf{z}_i$ are additionally associated with public "labels" L_i and inputs can only be combined with other inputs that share the same label. As in MIFE, a functional key sk_f is provided which allows the decryptor to compute $f(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ as long as the corresponding labels match, i.e. $L_1 = \ldots = L_n$.

3. *Dynamic Decentralized FE (DDFE):* DDFE [CDSG+20], as the name suggests, is a decentralized variant of FE, where not only can ciphertexts be generated locally and independently but so can the keys. In DDFE for some functionality f, the

setup step is localized and run independently by users, letting them generate their private and public keys individually. During encryption, the set of users with whom a given input or key object should be combined can be chosen dynamically. In more detail, each party can specify the set of parties with which its input may be combined, a label that controls which values should be considered together and the input $\mathbf{z}_i$ itself. Similarly, every user can also generate a key object which specifies the set of parties with which the key may be combined, and a key vector $\mathbf{c}_i$. For decryption, the ciphertexts and keys from the parties who mutually agree to combine their inputs and keys are put together to compute $f(\{\mathbf{z}_i\}_i, \{\mathbf{c}_j\}_j)$.

We note that DDFE implies MCFE which implies MIFE[1].

1.2.2 Distributed Authority

In the second part, we construct threshold signatures and blind signatures from lattice based assumptions.

Threshold signature schemes [Des94] enable distribution of the signature issuing capability to multiple users to mitigate the threat of signing key compromise. In more detail, in a t-out-of-N threshold signature scheme, each of the N parties holds a partial signing key and any set of at least t parties can participate in a protocol to generate a signature. The security requires that any set of less than t parties cannot generate a signature. Thus, it provides security against $t - 1$ corruptions.

While threshold signatures have been studied for a long time [Lin17b; DKLS18; CCL+19; GGN16; GG18; LN18; DKLS19; DOK+20; CCL+20; CGG+20; GKSŚ20; DJN+20; GG20; BKP13], they have received renewed attention in recent years due to numerous applications in modern topics such as cryptocurrencies and blockchains. Most prior work has focused on creating distributed versions of ECDSA or Schnorr signatures LN18; GG18; DKLS19; CCL+19; CCL+20 which are not quantum secure. From lattice based assumptions, which are conjectured to be post quantum secure, the results are either not round optimal [CS19] or work only for more restrictive N-out-of-N access structure [DOTT21]. The only lattice-based, round-optimal threshold signature

[1] In this work, we use the term MCFE as a generalization of MIFE, so that it allows multiple uses of labels [CDG+18b]. In contrast, a weaker notion of MCFE, where each label can be used only once, does not imply MIFE [GGG+14; CDG+18a].

construction for t-out-of-N access structure is by Boneh *et al* [BGG+18], relying on the Learning With Errors problem (LWE). While their construction provides the first feasibility result, it has several limitations, which we address in this work. Firstly, they use the so-called technique of noise flooding which hides a sensitive noise into much larger noise, often known as flooding. Their construction uses exponential flooding which results in signature size of $\tilde{O}(\lambda^3)$, where λ is the security parameter. We reduce the noise flooding from exponential to polynomial by performing a careful analysis using Rényi divergence based distance measurement instead of statistical distance. This helps us bring down the signature size to $\tilde{O}(\lambda, \log^2 Q)$, where λ is the security parameter and Q is the number of signing queries. Second, the construction of [BGG+18] achieves only selective security, where the corrupt members need to be declared apriori. We provide two new constructions: (i) the first construction achieves partial adaptivity, where members can be corrupted at any stage, but all the corruptions need to be declared at once, and (ii) the second one achieves full adaptivity, where corruptions can happen in any order. Our fully adaptive construction, however, has the limitation that the signing key size grows with the number of signing queries, Q_S. Finally, to instantiate [BGG+18] scheme, we need a homomorphism-friendly signature scheme, for which we provide a variant of Lyubashevsky's signature [Lyu12] scheme which uses rejection sampling which renders it unsuitable for homomorphic computation. We trade off rejection sampling at a cost of adding a moderate noise of polynomial size.

We then continue the thread of studying distributed authority based primitives, with blind signatures. Blind signatures find numerous applications like e-voting, e-cash, cryptocurrencies, and many more. They are like standard digital signature schemes, but now the message and the signing key are held by two different parties, the user ($\mathcal{U}$) and the signer ($\mathcal{S}$). The user, $\mathcal{U}$, holding a public key and message, m may request a signature from the signer $\mathcal{S}$, such that the signer is not able to link a message-signature pair with a protocol execution, and the user is not able to forge signatures even after multiple interactions with the signer.

While there exist many practical blind signatures from number-theoretic assumptions, the situation is far less satisfactory from post-quantum assumptions. In this work, we propose the first overall practical, lattice-based blind signature, supporting an unbounded number of signature queries and additionally enjoying optimal round complexity. We provide a detailed estimate of parameters achieved – we obtain a signature of size slightly above 45KB, for a core-SVP hardness of 109 bits. The run-times of the signer, the user, and the verifier are also very small. The security of our construction stems from a new and arguably natural assumption which we introduce, called the one-more-ISIS assumption. This assumption can be seen as a lattice analogue of the one-more-RSA assumption by Bellare et al [JoC'03]. To gain confidence in our assumption, we provide a detailed analysis of diverse attack strategies.

1.3 ORGANISATION OF THE 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 describe our constructions for 2ABE for NC_1 using LWE and pairings and heuristic constructions of 3ABE for NC_1 and 2ABE for P. We also define our kABE to kPE compiler in the same chapter. We extend our construction of MIABE to support any constant arity using evasive and tensor LWE in Chapter 4. In chapter 5, we define our constructions of multi-party functional encryption for AWS. In Chapter 6, we describe our constructions for threshold signatures. We introduce our assumption of one-more-ISIS and describe our construction of blind signature based on the assumption in Chapter 7. In Chapter 8, we conclude the thesis and discuss interesting directions for future work.

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 provide definitions of lattices and discrete Gaussian distributions.

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

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

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

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

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 $\mathbf{c} \in \mathbb{R}^m$ and any real $s > 0$, the (spherical) Gaussian function with standard deviation parameter s and center $\mathbf{c}$ is defined as:

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

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 $\Lambda \subseteq \mathbb{R}^m$, 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 $\mathbf{c}$.

Definition 2.4 (Smoothing parameter)**.** The smoothing parameter of an m-dimensional lattice Λ with respect to $\epsilon > 0$, denoted by $\eta_\epsilon(\Lambda)$, is the smallest $s > 0$, such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.

2.1.1 Hardness Assumptions

We will need the Learning With Errors (LWE) problem, which is known to be at least as hard as certain standard lattice problems in the worst case [Reg09; BLP+13] when it is appropriately parameterized.

Definition 2.5 (Learning With Errors (LWE))**.** Let q, n, m, α be functions of a parameter λ and χ be some distribution over $\mathbb{Z}$. For a secret $\mathbf{s} \in \mathbb{Z}_q^n$, the distribution $A_{q,n,\chi,\mathbf{s}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained by sampling $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an $e \leftarrow \chi$, and returning $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^{n+1}$. The Learning With Errors problem $\mathsf{LWE}_{q,n,m,\chi}$ is as follows: For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, the goal is to distinguish between the distributions:

$$D_0(\mathbf{s}) := U(\mathbb{Z}_q^{m\times(n+1)}) \text{ and } D_1(\mathbf{s}) := (A_{q,n,\chi,\mathbf{s}})^m.$$

We say that a PPT algorithm $\mathcal{A}$ solves $\mathsf{LWE}_{q,n,m,\chi}$ if it distinguishes $D_0(\mathbf{s})$ and $D_1(\mathbf{s})$ with non-negligible advantage (over the random coins of $\mathcal{A}$ and the randomness of the samples), with non-negligible probability over the randomness of $\mathbf{s}$.

Subexponential LWE*:* Here we allow the adversary $\mathcal{A}$ to run in $2^{o(\lambda)}$ time and win with $2^{-o(\lambda)}$ advantage (over the random coins of $\mathcal{A}$ and the randomness of the samples), with $2^{-o(\lambda)}$ probability over the randomness of $\mathbf{s}$.

When $\chi = \mathcal{D}_{\mathbb{Z},\alpha q}$, we denote the LWE distribution as $A_{q,n,\alpha,\mathbf{s}}$ and the LWE problem as $\mathsf{LWE}_{q,n,m,\alpha}$

Definition 2.6 (Short Integer Solution (SIS))**.** Let q, n, m, β be functions of a parameter λ. An instance of the $\mathsf{SIS}_{q,n,m,\beta}$ problem is a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$. A solution to the problem is a nonzero vector $\mathbf{v} \in \mathbb{Z}^m$ such that $\|\mathbf{v}\| \leq \beta$ and $\mathbf{A} \cdot \mathbf{v} = \mathbf{0} \bmod q$.
In order for the above problem to not be vacuously hard, we need to have $\beta \geq \sqrt{m} q^{n/m}$. This ensures that there exists a solution $\mathbf{v}$.

Like LWE, the SIS problem is known to be at least as hard as certain lattice problems in the worst case [Ajt96; MR07; GPV08], when it is appropriately parameterized. The same holds for the *inhomogeneous* version of the SIS problem stated below.

Definition 2.7 (Inhomogeneous Short Integer Solution (ISIS))**.** Let q, n, m, β be functions of a parameter λ. An instance of the $\mathsf{ISIS}_{q,n,m,\beta}$ problem is a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{t} \leftarrow \mathbb{Z}_q^n$. A solution to the problem is a vector $\mathbf{v} \in \mathbb{Z}^m$ such that $\|\mathbf{v}\| \leq \beta$ and $\mathbf{A} \cdot \mathbf{v} = \mathbf{t} \mod q$.
The SIS (and ISIS) problems can be defined for other norms as well. In Chapter 7, we also use the following version of ISIS:

Let $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \{-d, \ldots, 0, \ldots, d\}^m$ and $\mathbf{t} = \mathbf{A}\mathbf{s}$. Then given $\mathbf{A}, \mathbf{t}$, the task is to find $\mathbf{s}' \in \{-d, \ldots, 0, \ldots, d\}^m$ such that $\mathbf{A}\mathbf{s}' = \mathbf{t}$.

The following lemma provides bound on the size of vectors sampled from discrete Gaussian distribution.

Lemma 2.1 ([Lyu12, Lemma 4.4])**.** *The following hold.*

1. *For any $k > 0$, $\Pr[|z| > k\sigma; z \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}] \leq 2\exp(-k^2/2)$.*
2. *For any $\sigma \geq 3/\sqrt{2\pi}$, $H_\infty(\mathcal{D}_{\mathbb{Z}^m,\sigma}) \geq m$.*
3. *For any $k > 1$,*

$$\Pr[\|\mathbf{z}\| > k\sigma\sqrt{m};\ \mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}] < k^m \exp(\frac{m}{2}(1 - k^2)).$$

2.1.2 Lattice Trapdoors

We will use algorithms for generating a random lattice with a trapdoor, and for sampling short vectors in a lattice coset. The first algorithm is derived from [Ajt99; GPV08; MP12], whereas the second is derived from [Kle00; GPV08; BLP+13].

Lemma 2.2. *Let q, n, m be positive integers with $q \geq 2$ and $m \geq 6n \log_2 q$.*

There is a PPT algorithm $\mathsf{TrapGen}(q, n, m)$ *that with probability* $1 - 2^{-\Omega(n)}$ *outputs a pair* $(\mathbf{A}, \mathbf{T}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^{m \times m}$ *such that* $\mathbf{A}$ *is within* $2^{-\Omega(n)}$ *statistical distance to uniform in* $\mathbb{Z}_q^{n \times m}$ *and* $\mathbf{T}$ *is a basis for* $\Lambda_q^{\perp}(\mathbf{A})$.

There is a PPT algorithm $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{u}, \sigma)$*, which takes as input the above pair* $(\mathbf{A}, \mathbf{T})$*, a vector* $\mathbf{u} \in \mathbb{Z}_q^n$ *and a sufficiently large* $\sigma = \Omega(\sqrt{n \log q \log m})$ *and outputs a vector* $\mathbf{e}$ *from* $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}), \sigma}$*. Further, with probability* $2^{-\Omega(n)}$*, we have* $\|\mathbf{e}\| \leq \sigma\sqrt{m}$.

For our purpose, we assume that the $\mathsf{SamplePre}$ algorithm provides the same output when invoked with the same input – this can be ensured by generating the randomness used by the algorithm using a PRF (with the given input as argument).

2.2 PUBLIC KEY ENCRYPTION

Definition 2.8 (Public Key Encryption (PKE))**.** A PKE scheme is a tuple of PPT algorithms denoted by $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The KeyGen algorithm is a probabilistic algorithm. It takes as input a security parameter λ, and outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$: The Enc algorithm is a probabilistic algorithm. It takes as input a public key pk and a message m and outputs an encryption ct of m.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m'$: The decryption algorithm is a deterministic algorithm. It takes as input a (secret) decryption key sk and a ciphertext ct and outputs a message m'.

Correctness. For correctness, we require that for all λ and for all m, following holds. For $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, we have that $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m] \geq$

$1 - \lambda^{-\omega(1)}$.

IND-CPA **Security** A PKE scheme is IND-CPA secure if for any adversary PPT adversary $\mathcal{A}$ the output of the following experiment $\mathsf{Expt}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$ is 1 with negligible probability.

1. The challenger runs generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
2. $\mathcal{A}$ then outputs two messages m_0 and m_1.
3. The challenger samples a bit $b \leftarrow \{0, 1\}$ and returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b'.
5. The experiment outputs 1 if $b' = b$.

2.3 DIGITAL SIGNATURE

Definition 2.9 (Digital Signature)**.** A digital signature scheme is a tuple of PPT algorithms denoted by $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{vk}, \mathsf{sk})$: The KeyGen algorithm is a probabilistic algorithm. It takes as input a security parameter λ and outputs a verification key vk and a secret signing key sk.
- $\mathsf{Sign}(\mathsf{sk}, m) \rightarrow \sigma$: The Sign algorithm is a probabilistic algorithm. It takes as input a signing key sk and a message $m \in \{0, 1\}^*$ and outputs a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma) \rightarrow \mathsf{accept/reject}$: The verification algorithm is a deterministic algorithm. It takes as input a verification key vk, a message m, and a signature σ, and outputs accept or reject.

Correctness. For correctness, we require that for all λ, the following holds. For $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, we have that $\Pr[\mathsf{Verify}(\mathsf{vk}, m, \sigma) = accept] \geq 1 - \lambda^{-\omega(1)}$.

Unforgeability A TS scheme is unforgeable if for any adversary PPT adversary $\mathcal{A}$ the output of the following experiment $\mathsf{Expt}^{\mathsf{uf}}_{\mathsf{Sig},\mathcal{A}}(1^\lambda)$ is 1 with negligible probability.

1. The challenger runs generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends vk to $\mathcal{A}$.
2. Adversary $\mathcal{A}$ then issues polynomial number signing queries, where for each query

it outputs a message $m \in \{0,1\}^*$.

3. The challenger computes σ_m as $\mathsf{Sign}(\mathsf{sk}, m)$ and provides it to $\mathcal{A}$.
4. At the end of the experiment, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).
5. The experiment outputs 1 if the following condition is met: m^* was not queried previously as a signing query and $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathsf{accept}$.

2.4 PSEUDORANDOM FUNCTION

Definition 2.10 (Pseudorandom functions (PRF))**.** A pseudorandom function (PRF) family $\mathcal{F} = \{\mathsf{PRF}^{\mathsf{K}}\}_{\mathsf{K}\in\mathcal{K}}$ with a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$ is a function family that consists of functions $\mathsf{PRF}^{\mathsf{K}} : \mathcal{X} \to \mathcal{Y}$. 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,

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

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

2.5 SOME USEFUL LEMMAS

Lemma 2.3 (Leftover Hash Lemma)**.** *Let $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$ be a 2-universal hash function family. Then for any random variable $X \in \mathcal{X}$, for $\varepsilon > 0$ such that $\log|\mathcal{Y}| \leq H_\infty(X) - 2\log(1/\varepsilon)$, the distributions*

$$(h, h(X)) \text{ and } (h, \mathcal{U}(\mathcal{Y}))$$

are within statistical distance ε.

Further, the family $\{\mathbf{A} \in \mathbb{Z}_q^{n\times m} : \mathbf{r} \mapsto \mathbf{Ar}\}$ is 2-universal for any prime q.

Lemma 2.4 (Smudging Lemma [WWW22])**.** *Let λ be a security parameter. Take any*

$a \in \mathbb{Z}$ where $|a| \leq B$. Suppose $\chi \geq B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ is $\mathrm{negl}(\lambda)$.

CHAPTER 3

MULTI-INPUT ATTRIBUTE BASED AND PREDICATE ENCRYPTION

In this, and the next chapter, we study multi-input attribute based and predicate encryption where the data to be encrypted is distributed among different parties.

3.1 INTRODUCTION

Attribute based encryption (ABE) is a generalization of public key encryption which enables fine grained access control on encrypted data. In an ABE scheme, the ciphertext is associated with a secret message m and a public *attribute* vector $\mathbf{x}$ while a secret key is associated with a function f. Decryption succeeds to reveal m if and only if $f(\mathbf{x}) = 1$. Security seeks ciphertext indistinguishability in the presence of *collusion* attacks, namely an adversary possessing a collection of keys $\{\mathsf{sk}_{f_i}\}_{i\in[\text{poly}]}$ should not be able to distinguish between ciphertexts corresponding to $(\mathbf{x}, m_0)$ and $(\mathbf{x}, m_1)$ unless one of the keys $\mathsf{sk}_{f_{i^*}}$ is *individually* authorised to decrypt, i.e. $f_{i^*}(\mathbf{x}) = 1$. ABE comes in two flavours – "key-policy" and "ciphertext-policy", depending on whether the function f is embedded in the key or the ciphertext.

The stronger notion of predicate encryption (PE) [BW07; SBC+07; KSW08; GVW15] further requires the attribute vector $\mathbf{x}$ to be hidden so that ciphertexts corresponding to $(\mathbf{x}_0, m_0)$ and $(\mathbf{x}_1, m_1)$ remain indistinguishable so long as $f_i(\mathbf{x}_0) = f_i(\mathbf{x}_1) = 0$ for all secret keys $\{\mathsf{sk}_{f_i}\}_{i\in[\text{poly}]}$ seen by the adversary.

Both ABE and PE have been widely studied, and possess elegant instantiations from a variety of assumptions [SW05; GPSW06; BW07; KSW08; LOS+10; OT10; OT12; CW14; AFV11; LW11; LW12; Wat12; GVW13; Wee14; Att14; BGG+14; GVW15;

GV15; BV16; BW07; SBC+07; KSW08; GVW15]. Despite all this amazing progress, however, all known constructions supported the single input setting – namely, the function f embedded in the secret key sk_f has arity one, so that the secret key can be used to decrypt only a single ciphertext at a time. While the more realistic multi-input setting has been studied for other closely related notions such as fully homomorphic encryption [LATV12; CM15; MW16] and functional encryption [GGG+14; AJ15; AGRW17; DOT18; ACF+18; CDG+18a; Tom19; ABKW19; ABG19; LT19; AGT21a], this was not investigated at all in the context of predicate encryption, and only sparingly [BJK+18] in the context of attribute based encryption, prior to our work. In concurrent work, Francati *et al.* [FFMV23] provided multi-input PE schemes for restricted functionality of conjunctions of bounded polynomial depth from LWE in a weaker security model that does not allow collusions. A more detailed comparison with their results is given in Chapter 4.

Prior Work. Brakerski et al. [BJK+18] studied the notion of MIABE and showed that MIABE for polynomial arity implies *witness encryption* (WE). However, though they provided the first definition of MIABE, they only used it as a new pathway for achieving witness encryption, not as a notion with its own applications – in their definition, only the first encryptor has any input, since this suffices for WE. They do not consider strong notions of security or provide any constructions of MIABE. They also defined the notion of non-trivially exponentially efficient witness encryption (XWE), where the encryption run-time is only required to be much smaller than the trivial 2^m bound for NP relations with witness size m. They show how to construct such XWE schemes for all of NP with encryption run-time $2^{m/2}$ using the single input ABE by [GVW13]. For encryption run-time $2^{\gamma \cdot m}$, the term γ is denoted as *compression* factor, and they explicitly left open the problem of constructing XWE schemes with an improved compression factor.

ABE and PE as special cases of functional encryption (FE): Both ABE and PE can be captured as special cases of *functional encryption* [SW05; BSW11], which has been

studied extensively, in both the single-input [SW05; BSW11; GVW13; BGG+14] and multi-input setting [GGG+14; AJ15; AGRW17; DOT18; ACF+18; CDG+18a; Tom19; ABKW19; ABG19; LT19; AGT21a]. Recall that in functional encryption (FE), a secret key is associated with a function f, a ciphertext is associated with an input $\mathbf{x}$ and decryption allows to recover $f(\mathbf{x})$ and nothing else. It is easy to see that PE and ABE are both special cases of FE – in particular, both PE and ABE achieve the same functionality but restrict the security requirements of FE. In PE, we ask that the attribute $\mathbf{x}$ be hidden but only when the adversary does not see any decrypting keys, namely $f_i(\mathbf{x}) = 0$ for all function keys f_i received by the adversary. On the other hand, in FE, the attacker may request a key sk_f, so long as f does not distinguish the challenge messages $(\mathbf{x}_0, m_0), (\mathbf{x}_1, m_1)$, namely, we may have $f(\mathbf{x}_0) = f(\mathbf{x}_1) = 1$ so long as $m_0 = m_1$[1]. In the even weaker ABE, we do not ask any notion of hiding for $\mathbf{x}$, and this may be provided in the clear with the ciphertext.

Why not Functional Encryption? The informed reader may wonder what is the advantage of studying primitives like MIPE or MIABE when these are special cases of multi-input functional encryption (miFE), which has recently been constructed from standard assumptions [JLS21; AJ15]. It was shown by [AJ15; BV15a] that FE satisfying a certain efficiency property (known as *compactness*) implies multi-input functional encryption, which in turn implies the powerful primitive of *indistinguishability obfuscation* (iO) [BGI+01]. A long line of exciting works endeavoured to construct compact FE (and hence iO) from standard assumptions [Lin16; Lin17a; LV16; Agr19; AJL+19; JLMS19; GJLS21], coming ever-closer, until the very recent work of Jain, Lin and Sahai closed the last remaining gap and achieved this much sought after goal [JLS21; JLS22]. In [JLS21; JLS22], the authors provide a construction for compact FE, which in turn implies miFE for polynomial arity (albeit with exponential loss in the reduction).

[1] We note that a message m separate from attribute $\mathbf{x}$ is not required in the definition of FE, but we include it here for simpler comparison with PE and ABE.

Going via the route of compact FE, we obtain an exciting feasibility result for miFE and hence MIABE as well as MIPE. However, we argue that using something as strong as miFE or iO to construct MIABE and MIPE is undesirable, and indeed an "overkill" for the following reasons:

- *Assumptions:* Compact FE of [JLS21] is constructed via a careful combination of 4 assumptions – Learning Parity with Noise (LPN), Learning With Errors (LWE), SXDH assumption on Pairings, and pseudorandom generators computable in constant depth. In the follow-up work of [JLS22], this set of assumptions was trimmed to exclude LWE. Therefore any construction built using compact FE must make at least this set of assumptions, which is restrictive. A major goal in the theory of cryptography is developing constructions from diverse assumptions.

- *Complexity:* The construction of compact FE is extremely complex, comprising a series of careful steps, and this must then be lifted to miFE using another complex construction [AJ15]. Unlike FE, both PE and ABE are *much simpler, "all or nothing"* primitives and permit direct constructions in the single-input setting [GVW13; BGG+14; GVW15]. Do we need the full complexity of an miFE construction to get MIPE or MIABE? Indeed, even in the context of miFE, there is a large body of work that studies direct constructions for smaller function classes such as linear and quadratic functions [AGRW17; DOT18; ACF+18; CDG+18a; Tom19; ABKW19; ABG19; LT19; AGT21a].

- *New Techniques:* Finally and most importantly, we believe that it is extremely useful to develop new techniques for simpler primitives that are not known to be in obfustopia, and provide direct constructions. While direct constructions are likely to be more efficient, and are interesting in their own right, they may also lead to new pathways even for obfustopia primitives such as witness encryption or compact FE. Note that the only known construction of FE from standard assumptions is by [JLS21; JLS22], which makes crucial (and surprising) use of LPN in order to overcome a technical barrier – is LPN necessary for other primitives implied by compact FE? We believe that exploring new methods to construct weaker primitives is of central importance in developing better understanding of cryptographic assumptions, their power and limits.

3.2 OUR RESULTS

In this chapter, we initiate the study of multi-input predicate and attribute based encryption (MIABE and MIPE) and make the following contributions:

1. *Formalizing Security:* We provide definitions for MIABE and MIPE in the *symmetric* key setting and formalize two security notions in the standard *indistinguishability*

(IND) paradigm, against *unbounded* collusions. The first (regular) notion of security assumes that the attacker does not receive any decrypting keys, as is standard in the case of PE/ABE. The second *strong* notion, allows some decrypting queries in restricted settings.

2. *Two-input* ABE *for* NC_1 *from* LWE *and Pairings:* We provide the first constructions for two-input *key-policy* ABE for NC_1 from LWE and pairings. Our construction leverages a surprising connection between techniques recently developed by Agrawal and Yamada [AY20] in the context of succinct *single*-input *ciphertext-policy* ABE, to the seemingly unrelated problem of *two*-input *key-policy* ABE. Similarly to [AY20], our construction is proven secure in the bilinear generic group model. By leveraging inner product functional encryption and using (a variant of) the KOALA knowledge assumption, we obtain a construction in the standard model analogously to Agrawal, Wichs and Yamada [AWY20].

3. *Heuristic two-input* ABE *for* P *from Lattices:* We show that techniques developed for succinct single-input ciphertext-policy ABE by Brakerski and Vaikuntanathan [BV22] can also be seen from the lens of MIABE and obtain the first two-input key-policy ABE from lattices for P. Similarly to [BV22], this construction is heuristic.

4. *Heuristic three-input* ABE *and* PE *for* NC_1 *from Pairings and Lattices:* We obtain the first *three*-input ABE for NC_1 by harnessing the powers of both the Agrawal-Yamada [AY20] and the Brakerski-Vaikuntanathan [BV22] constructions.

5. *Multi-input* ABE *to multi-input* PE *via Lockable Obfuscation:* We provide a generic compiler that lifts multi-input ABE to multi-input PE by relying on the hiding properties of Lockable Obfuscation (LO) by Wichs-Zirdelis and Goyal-Koppula-Waters (FOCS 2018), which can be based on LWE. Our compiler generalises such a compiler for single-input setting to the much more challenging setting of multiple inputs. By instantiating our compiler with our new two and three-input ABE schemes, we obtain the first constructions of two and three-input PE schemes.

In Chapter 4, we extend our results to any constant arity for class NC_1 and P under recently introduced lattice-based assumptions of evasive and tensor LWE [Tsa22; Wee22].

Our constructions of multi-input ABE provide the first improvement to the compression factor (from $1/2$ to $1/3$ or $1/4$) of *non-trivially exponentially efficient Witness Encryption* defined by Brakerski et al. [BJK+18] without relying on compact functional encryption or indistinguishability obfuscation. We believe that the unexpected connection between

succinct single-input ciphertext-policy ABE and multi-input key-policy ABE may lead to a new pathway for witness encryption.

3.3 TECHNICAL OVERVIEW

In this section, we begin with an overview of our modeling of multi-input attribute based and predicate encryption followed by an overview of the techniques used in our constructions and proof strategies.

Modeling Multi-Input Attribute Based and Predicate Encryption. Our first contribution is to model multi-input attribute based encryption (MIABE) and predicate encryption (MIPE) as relevant primitives in their own right. To begin, we observe that similarly to multi-input functional encryption (miFE) [GGG+14], these primitives are meaningful primarily in the symmetric key setting where the encryptor requires a secret key to compute a ciphertext. This is to prevent the primitive becoming trivial due to excessive leakage occurring by virtue of functionality. In more detail, let us say k encryptors compute an unbounded number ciphertexts in each slot, i.e. $\{(\mathbf{x}_1^j, m_1^j), \ldots (\mathbf{x}_k^j, m_k^j)\}_{j\in[\text{poly}]}$ and the adversary obtains secret keys corresponding to functions $\{f_i\}_{i\in[\text{poly}]}$. In the multi-input setting, ciphertexts across slots can be combined, allowing the adversary to compute $f_i(\mathbf{x}_1^{j_1}, \mathbf{x}_2^{j_2}, \ldots, \mathbf{x}_k^{j_k})$ for any indices $i, j_1, \ldots, j_k \in [\text{poly}]$. In the public key setting, an adversary can easily encrypt messages for various attributes of its choice and decrypt these with the challenge ciphertext in a given slot to learn a potentially unbounded amount of information. [2] Due to this, we believe that the primitives of MIABE and MIPE are meaningful in the

[2] The triviality of public-key MIABE depends on the function class being supported. For example, consider the inner product functionality (which is in NC_1) defined as - let $f_{\mathbf{v}}(\mathbf{x}_1, \mathbf{x}_2) = 1$ if $\langle \mathbf{v}, \mathbf{x}_1 || \mathbf{x}_2 \rangle = 0$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are inputs in the ciphertext and $\mathbf{v}$ is in the key. Given a slot-1 ciphertext $\mathsf{CT}_1(\mathbf{x}_1, m_1)$ we would like to argue that m_1 remains hidden. However, in the public key case, it is possible to compute slot-2 ciphertext $\mathsf{CT}_2(\mathbf{x}_2, m_2)$ so that for any $v, \mathbf{x}_1$ (note that these are public) the decryption condition can be satisfied and the message m_1 can be recovered. This argument can also be extended to some interesting polynomials. On the other hand, there are function classes in NC_1, such as 3-SAT where it would be hard for the attacker to find a satisfying input and even the public key setting would not create excessive leakage (although it is unclear if such a functionality is useful in practice).

symmetric key setting where encryption also requires a secret key.

For security, we require the standard notion of ciphertext indistinguishability in the presence of collusion attacks, as in the single-input setting. Recall that in the single-input setting, the adversary cannot request any decrypting keys for challenge ciphertexts to prevent trivial attacks. However, since we are in the symmetric key setting where the adversary cannot encrypt herself, we propose an additional notion of *strong* security which also permits the adversary to request decrypting ciphertexts in some cases. In more detail, for the case of MIABE, our strong security game allows the attacker to request function keys for $\{f_i\}_{i\in[\text{poly}]}$ and ciphertexts for tuples $\{(\mathbf{x}_1^j, m_{\beta,1}^j), \ldots, (\mathbf{x}_k^j, m_{\beta,k}^j)\}_{\beta\in\{0,1\},j\in[\text{poly}]}$ so that it may hold that $f_i(\mathbf{x}_1^{j_1}, \ldots, \mathbf{x}_k^{j_k}) = 1$ for some $i, j_1, \ldots, j_k \in [\text{poly}]$ as long as the challenge messages do not distinguish, i.e. $(m_{1,0}^{j_1} = m_{1,1}^{j_1}), \ldots, (m_{k,0}^{j_k} = m_{k,1}^{j_k})$. For the case of MIPE, we analogously define a strong version of security by asking that if $f_i(\mathbf{x}_{1,\beta}^{j_1}, \ldots, \mathbf{x}_{k,\beta}^{j_k}) = 1$ holds for some $i, j_1, \ldots, j_k \in [\text{poly}]$ and $\beta \in \{0, 1\}$, then it is also true that $(\mathbf{x}_{1,0}^{j_1}, \ldots, \mathbf{x}_{k,0}^{j_k}) = (\mathbf{x}_{1,1}^{j_1}, \ldots, \mathbf{x}_{k,1}^{j_k})$ and $(m_{1,0}^{j_1}, \ldots, m_{k,0}^{j_k}) = (m_{1,1}^{j_1}, \ldots, m_{k,1}^{j_k})$. For more details, please see Section 3.5.

Constructing Two Input ABE from LWE and Bilinear GGM. In constructing two input ABE (2ABE), the main difficulty is to satisfy two seemingly contradicting requirements at the same time: (1) the two ciphertexts should be created independently, (2) these ciphertexts should be combined in a way that decryption is possible. If we look at specific ABE schemes (e.g., [GPSW06; BGG+14]), it seems that these requirements cannot be satisfied simultaneously. If we want to satisfy the second requirement, the two ciphertexts should have common randomness. However to satisfy the first requirement, the randomness in the two ciphertexts needs to be sampled independently. An approach might be to fix the randomness and put it into the master secret key which is then used by both ciphertexts – but this will compromise security since fresh randomness is crucial in safeguarding semantic security.

Generating Joint Randomness: For resolving this problem, we consider a scheme that modifies two independently generated ciphertexts so that they have common randomness and then "joins" them. This common randomness is jointly generated using independently chosen randomness in each ciphertext by using a pairing operation. Specifically, we compute a ciphertext for slot 1 with randomness t_1 and encode it in $\mathbf{G}_1$ and similarly, for slot 2 with randomness t_2 in $\mathbf{G}_2$, where $\mathbf{G} : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbb{G}_T$ is a pairing group with prime order q. Then, both ciphertexts may be combined to form a new ciphertext with respect to the randomness $t_1 t_2$ on $\mathbf{G}_T$. This approach seems to be promising, because we can uniquely separate every pair of ciphertexts, since each pair (i, j) will have unique randomness $t_1^i t_2^j$. In the generic group model, this is sufficient to prohibit "mix and match" attacks that try to combine components of different ciphertexts in the same slot.

Moving Beyond Degree 2*:* However, since we "used up" the pairing operation here, it appears we cannot perform more than linear operations on the generated ciphertext, which would severely restrict the function class supported by our construction. In particular, pairing based ABE schemes seem not to be compatible with the above approach, because these require additional multiplication in the exponent during decryption, which cannot be supported using a bilinear map. However, at this juncture, a trick suggested by Agrawal and Yamada [AY20] comes to our rescue – to combine lattice based ABE with bilinear maps in a way that lets us get the "best of both".

At a high level, the Agrawal-Yamada trick rests on the observation that in certain lattice based ABE schemes [BGG+14; GV15], decryption is structured as follows: (i) evaluate the circuit f on ciphertext encodings of $\mathbf{x}$, (ii) compute a matrix-vector product of the ciphertext matrix and secret key vector, (iii) perform a rounding operation to recover the message. Crucially, step (i) in the above description is in fact a *linear* operation over the encodings, even for circuits in P, and the only nonlinear part of decryption is the

rounding operation in step (iii). They observe that steps (i) and (ii) may be done "upstairs" in the exponent and step (iii) may be done "downstairs" by recovering the exponent brute force, when it is small enough. Importantly, the exponent is small enough when the circuit class is restricted to NC_1 using asymmetry in noise growth [GV15; GVW13]. While this idea was developed in the context of a single-input ciphertext-policy ABE, it appears to be exactly what we need for *two*-input key-policy ABE!

Perspective: Connection to Broadcast Encryption: In hindsight, the application of optimal broadcast encryption requires *succinctness* of the ciphertext, which recent constructions [BV22; AY20; AWY20] obtain by relying on the *decomposability* of specific ABE schemes [BGG$^+$14; GV15] – this decomposability is also what the multi-input setting intrinsically requires, albeit for a different reason. In more detail, decomposability means that the ciphertext for a vector $\mathbf{x}$ can be decomposed into $|\mathbf{x}|$ ciphertext components each encoding a single bit $\mathbf{x}_i$, and these components can be tied together using common randomness to yield a complete ciphertext. The bit by bit encoding of the vector allows $2|\mathbf{x}|$ ciphertext components, each component encoding both bits for a given position, to together encode $2^{|\mathbf{x}|}$ possible values of $\mathbf{x}$, which leads to the succinctness of ciphertext in optimal broadcast encryption schemes [BV22; AY20; AWY20]. In the setting of multi-input ABE, decomposability allows to morph independently generated *full* ciphertexts with distinct randomness to *components of a single* ciphertext with common randomness. The randomness is "merged" using pairings (or lattices, see below) and the resultant ciphertext can now be treated like the ciphertext of a single input scheme.

Adapting to the 2ABE *Setting:* Let us recall the structure of the ciphertext in scheme of Boneh et al. [BGG$^+$14], which is denoted as BGG + 18 hereafter. As discussed above, a

ciphertext for an attribute $\mathbf{x} \in [2\ell]$[3] in BGG + 18 is generated by first generating LWE encodings (their exact structure is not important for this overview) for *all possible* values of the attribute $\mathbf{x}$, namely, $\{\psi_{i,b}\}_{i\in[2\ell],b\in\{0,1\}}$ (along with other components which are not relevant here) and then selecting $\{\psi_{i,x_i}\}_{i\in[2\ell]}$ based on $\mathbf{x}$, where x_i is the i-th bit of the attribute string $\mathbf{x}$.

Given the above structure, a candidate scheme works as follows. The setup algorithm computes encodings for all possible $\mathbf{x}$, namely $\{\psi_{i,b}\}_{i,b}$ and puts them into the master secret key. The encryptor for slot 1 chooses $t_1 \leftarrow \mathbb{Z}_q$ and encodes $(t_1, \{t_1\psi_{i,x_{1,i}}\}_{i\in[\ell]})$ in the exponent of $\mathbb{G}_1$. Similarly, the encryptor for slot 2 chooses $t_2 \leftarrow \mathbb{Z}_q$ and encodes $(t_2, \{t_2\psi_{i,x_{2,i-\ell}}\}_{i\in[\ell+1,2\ell]})$ in the exponent of $\mathbb{G}_2$. In decryption, we compute a pairing of matching components of the two ciphertexts to obtain $(t_1t_2, \{t_1t_2\psi_{i,x_i}\}_{i\in[2\ell]})$ in the exponent of $\mathbb{G}_T$. Using the BGG + 18 decryption procedure described above, we may perform linear operations to evaluate the circuit, apply the BGG + 18 secret key and obtain the message plus noise in the exponent, which is brought "downstairs" by brute force to perform the rounding and recover the message.

Challenges in Proving Security. While the above sketch provides a construction template, security is far from obvious. Indeed, some thought reveals that the multi-input setting creates delicate attack scenarios that need care to handle. As an example, consider the strong security definition which allows the adversary to request ciphertexts that are decryptable by secret keys so long as they do not lead to a distinguishing attack. For simplicity, let us restrict to the setting where only the slot 1 ciphertext carries a message and slot 2 ciphertexts carry nothing except attributes (this restriction can be removed). Now, a slot 1 ciphertext may carry a message that depends on the challenger's secret bit as long as it is not decryptable by any key. However, slot 2 ciphertexts may participate in decryption with other slot 1 ciphertexts that do not encode the challenge bit, and decryption can (and does) lead to randomness leakage of participating ciphertexts. When

[3]The length of the attribute is set to 2ℓ to match our two-input setting.

such a “leaky” slot 2 ciphertext is combined with the challenge slot 1 ciphertext for decryption, security breaks down.

For concreteness, let us consider the setting where the adversary makes slot 1 ciphertext queries for $(\mathbf{x}_1, (m_0, m_1))$ and $(\mathbf{x}'_1, (m'_0, m'_1))$ and slot 2 ciphertext query for $(\mathbf{x}_2)$. Furthermore, the adversary makes a single key query for a circuit F such that $F(\mathbf{x}_1, \mathbf{x}_2) = 0$ (unauthorized) and $F(\mathbf{x}'_1, \mathbf{x}_2) = 1$ (authorized). Note that to prevent trivial attacks, we pose the restriction that $m'_0 = m'_1$, but we may have $m_0 \neq m_1$. In this setting, the 2ABE construction described above is not secure since the noise associated with the slot 2 ciphertext for $\mathbf{x}_2$ leaks during decryption of the jointly generated ciphertext for $(\mathbf{x}'_1, \mathbf{x}_2)$ and this prevents using BGG + 18 security for the pair $(\mathbf{x}_1, \mathbf{x}_2)$.

To resolve the above problem, we need to somehow “disconnect” randomness used in the challenge ciphertexts of slot 1 from randomness used in leaky/decrypting ciphertexts of other slots. This is tricky since the multi-input setting insists that ciphertexts be combined across slots in an unrestricted way. Fortunately, another technique developed [AY20] for a completely different reason comes to our assistance – we discontinue encoding the BGG + 18 ciphertexts in 2ABE ciphertexts for slot 2, so that even if a slot 2 ciphertext is decrypted, this does not affect the security of the BGG + 18 encoding. Instead, we encode a binary “selection vector" in the exponent of $\mathbf{G}_2$, which enables the decryptor to recover $\psi_{2,x_{2,i}}$ when matching positions of slot 1 and slot 2 ciphertext components are paired. In the context of broadcast encryption (i.e. succinct ciphertext-policy ABE) [AY20] this trick was developed because the key generator could not know the randomness used by the encryptor, and moreover this randomness is unbounded across unbounded ciphertexts. In our setting, this trick instead allows to break the leakage of correlated randomness caused by combining ciphertexts across different slots, some of which may be challenge ciphertexts and others of which may be decrypting ciphertexts. However, though we made progress we are still not done and the formal security argument still be required to address several issues – please see Section 3.6 for more details.

Constructing 2ABE in the Standard Model. We next turn to adapting the construction to the standard model – a natural starting point is the standard model adaptation of [AY20] by Agrawal, Wichs and Yamada [AWY20] which is based on a non-standard knowledge type assumption KOALA on bilinear groups. Our proof begins with these ideas but again departs significantly due to the nuanced security game of the multi-input setting – indeed, we will run into subtle technical issues related to the distribution of auxiliary information which will require us to formulate a variant of KOALA.

We first outline our construction, which uses a version of inner product functional encryption (IPFE), where one can directly encrypt group elements (rather than $\mathbb{Z}_q$ elements) and can generate a secret key for group elements. Thus, a ciphertext may encrypt a vector $[\mathbf{v}]_1$ and a secret key is for $[\mathbf{w}]_2$ and the decryption result of the ciphertext using the secret key is $[\langle \mathbf{v}, \mathbf{w} \rangle]_T$. Using IPFE and ideas similar to our first construction discussed above, we encode vectors into ciphertexts and secret keys so that the decryption result ends up with the BGG + 18 ciphertext randomized by a secret key specific randomness t. In more detail, a slot 1 ciphertext is an IPFE ciphertext encoding $[\mathbf{v}, 0]_2$ and a slot 2 ciphertext is an IPFE secret key encoding $[t\mathbf{w}, 0]_2$ so that $[t\langle \mathbf{v}, \mathbf{w} \rangle]_T$ is recovered upon decryption, which is a corresponding BGG + 18 ciphertext randomized by t on the exponent. Here, the last 0 entries are used for the security proof. We note that compared to the solution in bilinear generic group model we explained, we dropped the randomness on the ciphertext encoding and only the secret key encoding is randomized by t. The reason why the randomness on the ciphertext encoding can be removed is that the encoding is already protected by the IPFE and this change allows to simplify the construction and proof.

In the security game, we will have $\{\mathsf{ct}^{(i)} := \mathsf{IPFE.Enc}([\mathbf{v}^{(i)}, 0]_1)\}_i$ and $\{\mathsf{sk}^{(i)} := \mathsf{IPFE.sk}([t^{(i)}\mathbf{w}^{(i)}, 0]_2)\}_i$, where $\mathsf{ct}^{(i)}$ is the i-th slot 1 ciphertext and $\mathsf{sk}^{(i)}$ is the i-th slot 2 ciphertext. Let us say that the adversary requests Q ciphertexts in each slot. The security proof is by hybrid argument, where slot 1 ciphertexts are changed from

ciphertexts for challenge bit 0 to 1 one by one. Now, to change the message in a slot 1 ciphertext i^*, we must account for its combination with *all* slot 2 ciphertexts – note that such a constraint does not arise in single input ABE/BE [AWY20]. To handle this, we leverage the power of IPFE so that the Q second slot ciphertexts hardcode the decryption value for the chosen slot 1 ciphertext i^* and behave as before with other slot 1 ciphertexts. A bit more explicitly, the j-th secret key may be hardwired with $([t^{(j)}]_2, [t^{(j)}\mathsf{BGG+18.ct}^{(j)}]_2)$, where $\mathsf{BGG+18.ct}^{(j)}$ is a set of BGG + 18 ciphertexts derived from $\mathbf{v}^{(i^\star)}$ and $\mathbf{w}^{(j)}$. We note that since $\{\mathsf{BGG+18.ct}^{(j)}\}_j$ are derived from the same vector $\mathbf{v}^{(i^\star)}$, their distribution is mutually correlated.

At this stage, we have a vector of BGG + 18 ciphertexts encoded in the exponent, randomized with a unique random term $t^{(j)}$ and would like to change the ciphertexts $\mathsf{BGG+18.ct}^{(j)}$ into random strings using the security of BGG + 18. A similar situation was dealt with by [AWY20], who essentially showed that if $\mathsf{BGG+18.ct}^{(j)}$ is individually pseudorandom given an auxiliary information aux, then by a variant of the KOALA assumption, $\{[t^{(j)}]_2, [t^{(j)}\mathsf{BGG+18.ct}^{(j)}]_2\}_j$ looks pseudorandom, even if ciphertexts are mutually correlated for $j \in [Q]$. However, this idea is insufficient for our setting due to the distribution of the auxiliary information. In more detail, for the construction of [AWY20], it sufficed to have a single BGG + 18 secret key in aux, since their construction only needed a single key secure BGG + 18. By applying a standard trick in lattice cryptography, they could sample the secret key first (setting other parameters accordingly) and thus regard aux as a random string. In contrast, our scheme crucially requires multiple BGG + 18 secret keys, which can no longer be considered as random strings. This necessitates formulating a variant of the KOALA assumption whose distribution of the auxiliary input is structured rather than random. We do not know how to weaken this assumption using our current techniques and leave this improvement as an interesting open problem. For more details, please see Section 3.7.

Compiling multi-input ABE to multi-input PE. Next, we discuss how to lift k-input MIABE to k-input MIPE. For the purposes of the introduction, let us focus on the case of $k = 2$. As a warm-up, we begin with the simpler setting of standard security, i.e. where there are no decrypting ciphertexts.

The natural first idea to construct MIPE is to replace the single input ABE, BGG + 18 in our 2ABE scheme by single input PE, which has been constructed for all polynomial circuits by Gorbunov, Vaikuntanathan and Wee [GVW15]. However, this idea quickly runs into an insurmountable hurdle – for our construction template, we need to bound the decryption noise by polynomial so that it can be recovered by brute force computation of discrete log in the final step. This is possible for ABE supporting NC_1 using asymmetric noise growth [GV15]. In the context of PE however, we do not know how to restrict the noise growth to polynomial – this is due to the usage of the fully homomorphic encryption in the scheme, which extends the depth of the evaluated circuit beyond what can be handled.

An alternative path to convert ABE to PE in the single input setting uses the machinery of *Lockable Obfuscation* (LO) [GKW17; WZ17]. Lockable obfuscation allows to obfuscate a circuit C with respect to a lock value β and a message m. The obfuscated circuit on input x outputs m if $C(x) = \beta$ and $\perp$ otherwise. For security, LO requires that if β has high entropy in the view of the adversary, the obfuscated circuit should be indistinguishable from a garbage program that does not carry any information.

Single to Multiple Inputs. The conversion in the single input setting is as follows. To encrypt a message m for an attribute $\mathbf{x}$, we first encrypt a random value β using the ABE to obtain an ABE cipheretxt ct. We then construct a circuit $C[\mathsf{ct}]$ that hardwires ct in it, takes as input an ABE secret key and decrypts the hardwired ciphertext using it. We obfuscate $C[\mathsf{ct}]$ with respect to the lock value β and the message m. The final PE ciphertext is the obfuscated circuit. It is easy to see that the PE scheme has correctness,

since if the decryption is possible, β is recovered inside the obfuscated circuit and the lock is unlocked. By the correctness of LO, the message is revealed. In the security proof, we first change β encrypted inside ct to a zero string. This is possible using the security of ABE. Now the lock value β has high entropy from the view of the adversary. We then erase the information inside the obfuscated circuit, which includes the attribute information, using the security of LO.

Some thought reveals that the above conversion breaks down completely in the multi-input setting. For instance, if we apply the above conversion to a slot 1 ciphertext, the resulting obfuscation expects to receive slot 2 ciphertext in the clear. However, a slot 2 ciphertext of PE must also constitute an obfuscated circuit since otherwise the attribute associated with it will be leaked. But then there is no way to communicate between the two ciphertexts, both of which are obfuscated circuits!

To overcome this barrier, we develop a delicate *nested* approach which takes advantage of the fact that LO is powerful enough to handle general circuits. To restore communication between two ciphertexts while maintaining attribute privacy, we obfuscate a circuit for slot 1 that takes as input *another obfuscated circuit* for slot 2 and runs this inside itself. In more detail, the outer LO takes as input the "inner" LO circuit and the 2ABE secret key $\mathsf{2ABE.sk}_f$. The inner LO instance encodes the circuit for 2ABE decryption with the LO message as an SKE secret and the lock value as random tag β. It also has hardcoded in it the slot 2 2ABE ciphertext $\mathsf{2ABE.ct}_2$ with message β. The other piece of 2ABE, namely the slot 1 ciphertext $\mathsf{2ABE.ct}_1$ is hardwired in the outer LO. The outer LO encodes a circuit which runs the inner LO on inputs $\mathsf{2ABE.ct}_1$ and $\mathsf{2ABE.sk}_f$. By correctness of the inner LO, the 2ABE decryption with $\mathsf{2ABE.ct}_1$, $\mathsf{2ABE.ct}_2$ and $\mathsf{2ABE.sk}_f$ is executed and if the functionality is satisfied, the inner LO outputs the SKE secret key. Thus, the SKE secret key signals whether the inner LO is unlocked, and if so, uses the recovered key to decrypt an SKE ciphertext which is hardcoded in the circuit. This ciphertext encrypts some random γ which is also set as the lock value of outer LO. If the SKE

decryption succeeds, the lock value matches the decrypted value and outputs the message m which is the message in the outer LO. We note that the same SKE secret key must be used for both the inner and outer LO for them to effectively communicate.

Supporting Strong Security. This construction lends itself to a proof of security for the standard game where decrypting ciphertexts are not allowed, although via an intricate sequence of hybrids especially for the case of general k. We refer the reader to Section 3.8 for details and turn our attention to the far more challenging case of strong security. In the setting of strong security, the proof fails – note that once any slot 2 ciphertext is decrypted, we no longer have the guarantee that the message value of the inner obfuscation is hidden. Since this message is a secret key of an SKE scheme and is used to encrypt the lock values for slot 1 ciphertexts, security breaks down once more.

To overcome this hurdle, we must make the construction more complex so that the message value of the inner obfuscation is no longer a global secret and does not compromise security even if revealed. To implement this intuition, we let the inner obfuscation output a slot 2 (strong) 2ABE ciphertext when the lock is unlocked, which is then used to perform 2ABE decryption in the circuit of the outer LO. Now, even if the security of a inner obfuscated circuit is compromised, this does not necessarily mean that the security of the entire system is compromised because of the guarantees of the strong security game of 2ABE. While oversimplified, this intuition may now be formalized into a proof. For more details, please see Section 3.9.

Constructing 3ABE from Pairings and Lattices. Finally, we discuss our candidate construction for three input ABE scheme based on techniques developed by Brakerski and Vaikuntanathan [BV22] in conjunction with our 2ABE construction in Section 3.6.1. The work of Brakerski and Vaikuntanathan [BV22] provided a clever candidate for succinct ciphertext-policy ABE for P from lattices. Their construction also uses decomposability in order to achieve succinctness which is the starting point for the multi-input setting

as discussed above. Additionally, they provide novel ways to handle the lack of shared randomness between the key generator and encryptor – while [AY20] use pairings to generate shared randomness, [BV22] use lattice ideas and it is this part which makes their construction heuristic. Here, we show that the algebraic structure of their construction not only fits elegantly to the demands of the two-input setting, but can also be made compatible with our current 2ABE construction to *amplify* arity to three! This surprising synergy between two completely different candidates of broadcast encryption, namely Agrawal-Yamada and Brakerski-Vaikuntanathan, created by decomposability and novel techniques of handling randomness, already provides an XWE of compression factor $1/4$ as against the previous best known $1/2$ [BJK$^+$18], and may lead to other applications as well.

Recap of the Brakerski-Vaikuntanathan construction. To dig deeper into our construction, let us first recap the core ideas of [BV22]. First recall the well known fact that security of BGG + 18 encodings is lost when we have two encodings for the same position encoding a different bit, namely, $\psi_{i,0} = \mathbf{s}\mathbf{B}_i + \mathbf{e}_{i,0}$ and $\psi_{i,1} = \mathbf{s}(\mathbf{B}_i + \mathbf{G}) + \mathbf{e}_{i,1}$, where $\mathbf{s}$ is a LWE secret, $\mathbf{B}_i$ is a matrix, and $\mathbf{e}_{1,b}$ is an error vector for $b \in \{0, 1\}$. What [BV22] suggested is, if we augment BGG + 18 encodings and mask them appropriately, then both encodings can be published and still hope to be secure. Namely, they change BGG + 18 encodings to be $\psi_{i,b} = \mathbf{S}(\mathbf{B}_i + b\mathbf{G}) + \mathbf{E}_{i,b}$, where we replace the vector $\mathbf{s}$ with a matrix $\mathbf{S}$. They then mask the encodings by public (tall) matrices $\{\mathbf{C}_{i,b}\}_{i,b}$ as

$$\widehat{\psi}_{i,b} := \mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \mathbf{S}(\mathbf{B}_i + b\mathbf{G}) + \mathbf{E}_{i,b}$$

where $\{\widehat{\mathbf{S}}_{i,b}\}_{i,b}$ are random secret matrices. By releasing appropriate information, one can recover BGG + 18 encodings with different LWE secrets. In more detail, we can publish a short vector $\mathbf{t}_\mathbf{x}$ for any binary string $\mathbf{x}$ that satisfies $\mathbf{t}_\mathbf{x}\mathbf{C}_{i,x_i} = \mathbf{0}$ (and $\mathbf{t}_\mathbf{x}\mathbf{C}_{i,1-x_i}$ is random) for all i. This allows us to compute

$$\mathbf{t}_\mathbf{x}\left(\mathbf{C}_{i,x_i}\widehat{\mathbf{S}}_{i,x_i} + \mathbf{S}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{E}_{i,x_i}\right) = \mathbf{t}_\mathbf{x}\mathbf{S}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{t}_\mathbf{x}\mathbf{E}_{i,x_i} = \mathbf{s}_\mathbf{x}(\mathbf{B}_i + x_i\mathbf{G}) + \mathbf{e}_{\mathbf{x},i,x_i}$$

where we set $\mathbf{s}_{\mathbf{x}} = \mathbf{t}_{\mathbf{x}}\mathbf{S}$ and $\mathbf{e}_{\mathbf{x},i,b} = \mathbf{t}_{\mathbf{x}}\mathbf{E}_{i,b}$. Namely, we can obtain $\mathsf{BGG}+18$ samples specific to the string $\mathbf{x}$. This is similar to the idea of using pairings to choose the appropriate encoding based on the attribute string, which is used in our two-input ABE with strong security. Similarly to that case, the obtained encodings are randomized by the user specific randomness. One of the heuristic aspects of [BV22] is that in order for their scheme to be secure, we have to assume that there is no meaningful way to combine the $\mathsf{BGG}+18$ samples obtained from different vectors $\mathbf{t}_{\mathbf{x}}$ and $\mathbf{t}_{\mathbf{x}'}$.

Let us now adapt these techniques to provide a construction of two-input ABE. In our candidate, $\{\mathbf{B}_i\}_i$ and $\{\mathbf{C}_{i,b}\}_{i,b}$ matrices are made public.[4] An encryptor for the slot 1 computes for $i \in [\ell], b \in \{0,1\}$:

$$\left\{\psi_{i,x_{1,i}} := \mathbf{S}(\mathbf{B}_i + x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}}\right\}_i, \left\{\widehat{\psi}_{i,b} := \mathbf{C}_{\ell+i,b}\widehat{\mathbf{S}}_{\ell+i,b} + \mathbf{S}(\mathbf{B}_{\ell+i} + b\mathbf{G}) + \mathbf{E}_{\ell+i,b}\right\}_{i,b}$$

where $x_{1,i}$ denotes the i-th bit of the attribute $\mathbf{x}_1$ for slot 1, ℓ denotes the length of an attribute, and $\mathbf{S}$ and $\widehat{\mathbf{S}}_{i,b}$ are freshly chosen by the encryptor. Intuitively, this is a partially stripped off version of the encodings in [BV22]. We believe this does not harm security, because the encryptor provides one out of two encodings for each position that is not masked by $\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b}$. The encryptor for slot 2 generates a vector $\mathbf{t}_{\mathbf{x}_2}$ such that $\mathbf{t}_{\mathbf{x}_2}\mathbf{C}_{i,x_{2,\ell+i}} = \mathbf{0}$ for all $i \in [\ell]$. The secret key for function F is simply $\mathsf{BGG}+18$ secret key for the same function. In the decryption, the decryptor uses $\mathbf{t}_{\mathbf{x}_2}$ to choose $\mathsf{BGG}+18$ encodings for attribute $\mathbf{x}_2$ from $\{\widehat{\psi}_{i,b}\}_{i,b}$. The obtained encodings are with respect to the LWE secret $\mathbf{t}_{\mathbf{x}}\mathbf{S}$. The decryptor can also choose $\mathsf{BGG}+18$ encodings for attribute $\mathbf{x}_1$ from $\{\psi_i\}_i$. These obtained encodings constitutes a $\mathsf{BGG}+18$ ciphertext for attribute $(\mathbf{x}_1, \mathbf{x}_2)$, which can be decrypted by the $\mathsf{BGG}+18$ secret key. The intuition about security in [BV22] is that the $\mathsf{BGG}+18$ encodings obtained by using $\mathbf{t}_{\mathbf{x}}$ vectors cannot be combined in a meaningful way due to the different randomness.

Amplifying Arity. We now amplify arity by leveraging the above techniques in conjunction

[4] The construction described here is simplified. For example, we omit the additional message carrying part in the construction, which is not necessary for the overview.

with our pairing based construction. Our idea is to develop the scheme so that the decryptor can recover the above partially stripped off version of the encoding in the exponent from slot 1 and slot 2 ciphertexts by using the pairing operations, where the encodings may be randomized. Then, slot 3 ciphertext corresponds to a vector $\mathbf{t}_{\mathbf{x}_3}$, which annihilates $\mathbf{C}_{i,b}$ matrices for corresponding positions to the attribute $\mathbf{x}_3$. To do so, an encryptor for the first slot encodes

$$\{t_1\psi_{i,x_i}\}_{i\in[\ell]}, \quad \{t_1\psi_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \quad \{t_1\widehat{\psi_{i,b}}\}_{i\in[2\ell+1,3\ell],b\in\{0,1\}}$$

of the exponent of $\mathbf{G}_1$, where t_1 is freshly chosen randomness by the encryptor. An encryptor for the second slot encodes t_2, $t_2\mathbf{d}_{\mathbf{x}_2}$ in the exponent of $\mathbf{G}_2$, where t_2 is freshly chosen randomness by the encryptor and $\mathbf{d}_{\mathbf{x}_2}$ is a selector vector that chooses $\psi_{i,x_{2,i}}$ out of $(\psi_{i,0},\psi_{i,1})$ by the pairing operation. Concretely, $\mathbf{d}_{\mathbf{x}_2} = \{d_{i,b}\}_{i,b}$, where $d_{i,b} = 1$ if $b = x_{2,i}$ and 0 otherwise. These vectors are randomized by position-wise randomness as is the case for our other schemes. Finally, an encryptor for slot 3 with attribute $\mathbf{x}_3$ chooses $\mathbf{t}_{\mathbf{x}_3}$ such that $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}} = \mathbf{0}$.

A somewhat worrying aspect of the candidate above may be that both $t_1\psi_{i,0}$ and $t_1\psi_{i,1}$ are encoded on $\mathbf{G}_1$. However, this is also the case for [AY20] and as in that work, these two encodings are randomized by the position-wise randomness and cannot be combined in a meaningful way (at least in the GGM). The only way to combine them is to take a pairing product with $\mathbf{G}_2$ elements. However, after the operation, we end up with partially stripped encoding that is randomized with t_1t_2. Therefore, a successful attack against the scheme may end up with attacking a partially stripped version of [BV22], which we expect to be as secure as the original scheme. Please see Section 3.10 for more details.

Organisation of the chapter. The rest of the chapter is organised as follows. In Section 3.4, we provide the preliminaries used in this chapter. In Section 3.5, we define MIABE and MIPE. We construct 2ABE for NC_1 from LWE and pairings in Section 3.6. In Section 3.7, we provide the construction of 2ABE for NC_1 in standard model from

KOALA assumption. We define our compiler for kABE to kPE in Section 3.8. This compiler works for the weaker security. The compiler for 2PE with stronger security is given in Section 3.9. We provide our heuristic constructions for 3ABE for NC_1 and 2ABE for P in Sections 3.10 and 3.11, respectively.

3.4 PRELIMINARIES

Notation used in this chapter. By default, in this chapter, we treat a vector as a row vector. For any vector $\mathbf{x}$ of length ℓ, we let x_i denote the i-th coordinate of $\mathbf{x}$, for $i \in [\ell]$. We use $\mathbf{1}_{\ell \times m}$ (resp. $\mathbf{0}_{\ell \times m}$) to represent a matrix of dimensions $\ell \times m$ having each entry as 1 (resp. 0). Similarly, we write $\mathbf{1}_a$ (resp. $\mathbf{0}_a$) to represent $(1, \ldots, 1) \in \mathbb{Z}_q^a$ ($(0, \ldots, 0) \in \mathbb{Z}_q^a$) We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. For two distributions $\mathcal{D}_1$, $\mathcal{D}_2$, we use the notation $\mathcal{D}_1 \approx_c \mathcal{D}_2$ to denote that a PPT adversary cannot distinguish between the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ except only with negligible distinguishing advantage.

3.4.1 Single User Attribute Based Encryption

For ease of readability, we define single user CPABE and kpABE below.

Let $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0,1\}\}_\lambda$ be a relation where A_λ and B_λ denote "ciphertext attribute" and "key attribute" spaces. An attribute-based encryption (ABE) scheme for R is defined by the following PPT algorithms:

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

$\mathsf{Enc}(\mathsf{mpk}, X, \mu) \to \mathsf{ct}$**:** The encryption algorithm takes as input a master public key mpk,

a ciphertext attribute $X \in A_\lambda$, and a message bit μ. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, Y) \to \mathsf{sk}_Y$**:** The key generation algorithm takes as input the master public key mpk, the master secret key msk, and a key attribute $Y \in B_\lambda$. It outputs a private key sk_Y.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}, X, \mathsf{sk}_Y, Y) \to \mu$ or $\perp$**:** We assume that the decryption algorithm is deterministic. The decryption algorithm takes as input the master public key mpk, a ciphertext ct, ciphertext attribute $X \in A_\lambda$, a private key sk_Y, and private key attribute $Y \in B_\lambda$. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

Definition 3.1 (Correctness)**.** An ABE scheme for relation family R is correct if for all $\lambda \in \mathbb{N}$, $X \in A_\lambda, Y \in B_\lambda$ such that $R(X, Y) = 1$, and for all messages $\mu \in \mathsf{g}$,

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

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

Definition 3.2 ($\mathsf{Ada\text{-}IND}$ security for ABE)**.** For an ABE scheme $\mathsf{ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ for a relation family $R = \{R_\lambda : A_\lambda \times B_\lambda \to \{0, 1\}\}_\lambda$ and a message space $\{\mathsf{g}_\lambda\}_{\lambda \in \mathbb{N}}$ and an adversary $\mathcal{A}$, let us define $\mathsf{Ada\text{-}IND}$ security game, $\mathsf{Expt}^{\mathsf{Ada\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}$ as follows.

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

2. **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 $Y \in B_\lambda$. For each such query, the challenger replies with $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, Y)$.

   b) **Challenge Query:** At some point, $\mathcal{A}$ submits a pair of equal length messages $(\mu_0, \mu_1) \in (\mathsf{g})^2$ and the target $X^\star \in A_\lambda$ to the challenger. The

challenger samples a random bit $b \leftarrow \{0,1\}$ and replies to $\mathcal{A}$ with $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, X^\star, \mu_b)$.

We require that $R(X^\star, Y) = 0$ holds for any Y such that $\mathcal{A}$ makes a key query for Y in order to avoid trivial attacks.

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

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

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

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

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

Definition 3.3 (Ada-INDr security for ABE)**.** We define Ada-INDr security game, $\mathsf{Expt}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}$ similarly to Ada-IND security game, $\mathsf{Expt}^{\mathsf{Ada\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}$ 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 $b = 1$. We define the advantage $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies Ada-INDr security if the quantity is negligible.

We also consider (weaker) selective versions of the above notions, where $\mathcal{A}$ specifies its target $X^\star$ at the beginning of the game.

Definition 3.4 (Sel-IND security for ABE)**.** We define Sel-IND security game, $\mathsf{Expt}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}$ as Ada-IND security game with the exception that the adversary $\mathcal{A}$ has to choose the challenge ciphertext attribute $X^\star$ before the setup phase but key queries $Y_1, Y_2, \ldots$ and choice of (μ_0, μ_1) can still be adaptive. We define the advantage $\mathsf{Adv}^{\mathsf{Sel\text{-}IND}}_{\mathsf{ABE},\mathcal{A}}(1^\lambda)$ of the adversary $\mathcal{A}$ accordingly and say that the scheme satisfies Sel-IND security (or simply *selective security*) if the quantity is negligible.

Definition 3.5 (Sel-INDr security for ABE)**.** We define Sel-INDr security game,

$\mathsf{Expt}^{\mathsf{Sel\text{-}INDr}}_{\mathsf{ABE},\mathcal{A}}$ as Ada-INDr security game with the exception that the adversary $\mathcal{A}$ has to choose the challenge ciphertext attribute $X^\star$ before the setup phase but key queries $Y_1, Y_2, \ldots$ and choice of μ can still be adaptive. 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 Sel-INDr security if the quantity is negligible.

In the following, we recall definitions of various ABEs by specifying the relation. We start with the standard notions of ciphertext-policy attribute-based encryption (CPABE) and key-policy attribute-based encryption (kpABE).

CPABE for circuits. We define CPABE for circuit class $\{C_\lambda\}_\lambda$ by specifying the relation. Here, C_λ is a set of circuits with input length $\ell(\lambda)$ and binary output. We define $A^{\mathsf{CP}}_\lambda = C_\lambda$ and $B^{\mathsf{CP}}_\lambda = \{0,1\}^\ell$. Furthermore, we define the relation R^{CP}_λ as $R^{\mathsf{CP}}_\lambda(C, \mathbf{x}) = \neg C(\mathbf{x})$.[5]

kpABE for circuits. 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 $A^{\mathsf{KP}}_\lambda = \{0,1\}^\ell$ and $B^{\mathsf{KP}}_\lambda = C_\lambda$. We also define $R^{\mathsf{KP}}_\lambda : A^{\mathsf{KP}}_\lambda \times B^{\mathsf{KP}}_\lambda \to \{0,1\}$ as $R^{\mathsf{KP}}_\lambda(\mathbf{x}, C) = \neg C(\mathbf{x})$.

Remark 1. We observe that the symmetric key variants of the above definitions can be easily obtained by letting the encryptor have access to the master secret key and permitting the adversary to make ciphertext requests in the security game.

3.4.2 Lockable Obfuscation

We define lockable obfuscation [GKW17; WZ17] below. Let n, m, d be polynomials, and $C_{n,m,d}(\lambda)$ be the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bit input and $m(\lambda)$ bit output. A lockable obfuscator for $C_{n,m,d}$ consists of algorithms Obf and Eval with the following syntax. Let $\mathcal{M}$ be the message space.

$\mathsf{Obf}(1^\lambda, P, \mathsf{g}, \alpha) \to \widetilde{P}$: The obfuscation algorithm is a randomized algorithm that takes

[5] Here, we follow the standard convention in lattice-based cryptography where the decryption succeeds when $C(\mathbf{x}) = 0$ rather than $C(\mathbf{x}) = 1$.

as input the security parameter λ, a program $P \in C_{n,m,d}$, message $\mathsf{g} \in \mathcal{M}$ and 'lock string' $\alpha \in \{0,1\}^{m(\lambda)}$. It outputs a program $\widetilde{P}$.

$\mathsf{Eval}(\widetilde{P}, x) \to y \in \mathcal{M} \cup \{\perp\}$: The evaluator is a deterministic algorithm that takes as input a program $\widetilde{P}$ and a string $x \in \{0,1\}^{n(\lambda)}$. It outputs $y \in \mathcal{M} \cup \{\perp\}$.

Correctness: For correctness, it is required that if $P(x) = \alpha$, then the obfuscated program $\widetilde{P} \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{g}, \alpha)$, evaluated on input x, outputs g, and if $P(x) \neq \alpha$, then $\widetilde{P}$ outputs $\perp$ on input x.

Definition 3.6 (Perfect Correctness)**.** Let n, m, d be polynomials. A lockable obfuscation scheme for $C_{n,m,d}$ and message space $\mathcal{M}$ is said to be perfectly correct if it satisfies the following properties:

1. For all security parameters λ, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in C_{n,m,d}$ and messages $\mathsf{g} \in \mathcal{M}$, if $P(x) = \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{g}, \alpha), x) = \mathsf{g}.$$

2. For all security parameters λ, inputs $x \in \{0,1\}^{n(\lambda)}$, programs $P \in C_{n,m,d}$ and messages $\mathsf{g} \in \mathcal{M}$, if $P(x) \neq \alpha$, then
$$\mathsf{Eval}(\mathsf{Obf}(1^\lambda, P, \mathsf{g}, \alpha), x) = \perp .$$

Definition 3.7 (Security)**.** Let n, m, d be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $C_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure if there exists a PPT simulator Sim such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that:

$$\left| \Pr\left[\mathcal{A}_1(\widetilde{P}_b, \mathsf{st}) = b \;\middle|\; \begin{array}{r} (P, \mathsf{g}, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda) \\ b \leftarrow \{0,1\}, \alpha \leftarrow \{0,1\}^{m(\lambda)} \\ \widetilde{P}_0 \leftarrow \mathsf{Obf}(1^\lambda, P, \mathsf{g}, \alpha) \\ \widetilde{P}_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|P|}, 1^{|\mathsf{g}|}) \end{array} \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Analogously, we can define the security for multiple queries case.

Definition 3.8 (LO security with multiple queries)**.** Let n, m, d be polynomials. A lockable obfuscation scheme $(\mathsf{Obf}, \mathsf{Eval})$ for $C_{n,m,d}$ and message space $\mathcal{M}$ is said to be secure (for multiple adaptive queries) if there exists a PPT simulator Sim such that for all PPT adversaries $\mathcal{A}$, the probability of winning in the following game is $1/2 + \mathrm{negl}(\lambda)$.

The security game between challenger C and adversary $\mathcal{A}$ is defined as follows:

1. C Samples a bit $b \leftarrow \{0,1\}$.
2. $\mathcal{A}$ issues $p = p(\lambda)$ adaptive queries of the form (P^i, g^i) to C.
3. For each query, C returns $\widetilde{P}^i_b$, where

$$\widetilde{P}^i_0 \leftarrow \mathsf{Obf}(1^\lambda, P^i, \mathsf{g}^i, \alpha^i), \alpha_i \leftarrow \{0,1\}^{m(\lambda)} \text{ and } \widetilde{P}^i_1 \leftarrow \mathsf{Sim}(1^\lambda, 1^{|P^i|}, 1^{|\mathsf{g}^i|})$$

4. In the end, the adversary outputs a bit b'.

The adversary wins if $b' = b$.

Reduction from multi-queries definition to single query can be shown using hybrids. We sketch the reduction here. We consider $p + 1$ hybrids Hybrid_0 to Hybrid_p. In Hybrid_i, first i programs are simulated programs and remaining $p - i$ are obfuscated programs. Indisntinguishability of Hybrid_i and Hybrid_{i+1} follows from the security in case of single query.

3.4.3 Batch Inner Product Functional Encryption

We define batch inner product functional encryption (BIPFE) in the secret key setting. This is a straightforward extension of the standard notion of the IPFE in the secret key setting [BJK15; DDM16; LV16] and is introduced for the purpose of describing our scheme with notational ease. In BIPFE, a ciphertext and a secret key are associated with matrices of the same size consisting of group components $[\mathbf{V}]_1 = [(\mathbf{v}_1^\mathsf{T}, \ldots, \mathbf{v}_n^\mathsf{T})]_1 \in \mathbb{G}_1^{B \times n}$ and $[\mathbf{W}]_2 = [(\mathbf{w}_1^\mathsf{T}, \ldots, \mathbf{w}_n^\mathsf{T})]_2 \in \mathbb{G}_2^{B \times n}$, respectively. Here, we refer to B as the batch size

and n as the dimension. Upon decryption, the following is recovered

$$[\mathbf{V} \boxdot \mathbf{W}]_T := \left[\sum_{i \in [n]} \mathbf{v}_i \odot \mathbf{w}_i \right]_T .$$

Namely, we recover inner product of each row of $\mathbf{V}$ and $\mathbf{W}$ in parallel as a decryption result. More formal definition follows.

Let $\mathsf{GroupGen}$ be a group generator that outputs bilinear group $\mathbf{G} = (p, \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, e, [1]_1, [1]_2)$. A BIPFE scheme based on $\mathbb{G}$ consists of 4 efficient algorithms:

$\mathsf{Setup}(1^\lambda, 1^B, 1^n) \to \mathsf{msk}$**:** The setup algorithm takes as input the security parameter, the batch size B, the dimension n all in unary and outputs master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}]_2) \to \mathsf{sk}_{\mathbf{W}}$**:** The key generation algorithm takes as input the master secret key and a matrix of group elements $[\mathbf{W}]_2 \in \mathbb{G}_2$, and outputs a secret key $\mathsf{sk}_{\mathbf{W}}$.

$\mathsf{Enc}(\mathsf{msk}, [\mathbf{V}]_1) \to \mathsf{ct}_{\mathbf{V}}$**:** The encryption algorithm takes as input the master secret key and a matrix of group elements $[\mathbf{V}]_1$ and outputs a ciphertext $\mathsf{ct}_{\mathbf{V}}$.

$\mathsf{Dec}(\mathsf{sk}_{\mathbf{W}}, \mathsf{ct}_{\mathbf{V}}) \to [\mathbf{Z}]_T \vee \perp$**:** The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathbf{W}}$ and a ciphertext $\mathsf{ct}_{\mathbf{V}}$, and outputs an element $[\mathbf{Z}]_T \in \mathbb{G}_T$ or $\perp$.

Definition 3.9 (Correctness)**.** We say the BIPFE scheme satisfies decryption correctness if for all $\lambda \in \mathbb{N}$, all batch size B, all dimension n, and all matrices $\mathbf{V}, \mathbf{W} \in \mathbb{Z}_p^{B \times n}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\mathbf{W}}, \mathsf{ct}_{\mathbf{V}}) = [\mathbf{W} \boxdot \mathbf{V}]_T \;\middle|\; \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^n) \\ \mathsf{sk}_{\mathbf{W}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}]_2) \\ \mathsf{ct}_{\mathbf{V}} \leftarrow \mathsf{Enc}(\mathsf{msk}, [\mathbf{V}]_1) \end{array} \right] = 1 .$$

Next, we define the function hiding property.

Definition 3.10 (Function Hiding Security)**.** Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a BIPFE scheme as defined above. The scheme is function hiding if $\mathsf{Expt}^0_{\mathsf{FH}}$ is indistinguishable from $\mathsf{Expt}^1_{\mathsf{FH}}$ for all PPT adversary $\mathcal{A}$ where $\mathsf{Expt}^b_{\mathsf{FH}}$ for $b \in \{0, 1\}$ is defined as follows:

1. **Setup**: Run the adversary $\mathcal{A}$ on input 1^λ to obtain the batch size 1^B and the dimension 1^n from $\mathcal{A}$. Let $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda, 1^B, 1^n)$ and return msk to $\mathcal{A}$.

2. **Challenge**: Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ has 2 options:

   - $\mathcal{A}$ submits $[\mathbf{W}_0^{(i)}]_2, [\mathbf{W}_1^{(i)}]_2 \in \mathbb{G}_2^{B \times n}$ as a secret key query. Upon receiving this, compute $\mathsf{sk}^{(i)} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, [\mathbf{W}_b^{(i)}]_2)$ and return this to $\mathcal{A}$.

   - $\mathcal{A}$ submits $[\mathbf{V}_0^{(i)}]_1, [\mathbf{V}_1^{(i)}]_1 \in \mathbb{G}_1^{B \times n}$ as an encryption query. Upon receiving this, compute $\mathsf{ct}^{(i)} \leftarrow \mathsf{Enc}(\mathsf{msk}, [\mathbf{V}_b^{(i)}]_1)$ and return this to $\mathcal{A}$.

3. **Guess**: $\mathcal{A}$ outputs its guess b'.

The adversary is called admissible if $\mathbf{V}_0^{(i)} \boxdot \mathbf{W}_0^{(j)} = \mathbf{V}_1^{(i)} \boxdot \mathbf{W}_1^{(j)}$ for all combinations of i and j. We say that the BIPFE scheme is function hiding if $|\Pr[b = b'] - 1/2|$ is negligible for all admissible PPT adversaries.

Note that function hiding IPFE is captured as a special case of our notion of BIPFE with the batch size $B = 1$. It can be seen that function hiding IPFE can be converted to BIPFE by running the former in parallel for B times. Function hiding IPFE schemes are constructed from various assumptions including SXDH and DLIN [DDM16; LV16] and thus BIPFE can be constructed from the same assumptions.

3.4.4 Lattice Preliminaries

We use $\mathsf{LWE}_{q,n,m,\chi}$ assumption defined in Chapter 2, where n, m, q are such that $n = \mathrm{poly}(\lambda)$, $m \geq n\lceil \log q \rceil$. We define $\chi = \mathsf{SampZ}(\gamma)$, 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 $|z| \leq \sqrt{n}\gamma$.

We also consider subexponential hardness of LWE where the advantage of the adversary

is bounded by $2^{-n^\epsilon} \cdot \mathrm{negl}(\lambda)$ for some constant $0 < \epsilon < 1$ for all PPT $\mathcal{A}$[6]. As shown by previous works [Reg09; BLP+13], for $\chi = \mathsf{SampZ}(\gamma)$, the $\mathsf{LWE}_{q,n,m,\chi}$ 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 $\mathsf{LWE}_{q,n,m,\chi}$ with noise-to-modulus ratio 2^{-n^ϵ} is likely to be (subexponentially) hard for some constant ϵ.

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 $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ be an output distribution of $\mathsf{SampZ}(\gamma)^{m \times m'}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_\gamma^{-1}(\mathbf{V}) = \mathbf{V}$. A γ-trapdoor for $\mathbf{A}$ is a trapdoor that enables one to sample from the distribution $\mathbf{A}_\gamma^{-1}(\mathbf{V})$ 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 $\mathbf{A}_\gamma^{-1}$. We also define the special gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ as the matrix obtained by padding $\mathbf{I}_n \otimes (1, 2, 4, 8, \ldots, 2^{\lceil \log q \rceil})$ with zero-columns. The following properties had been established in a long sequence of works [GPV08; CHKP10; ABB10a; ABB10b; MP12; BLP+13].

Lemma 3.1 (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 $\tau_0 = \omega(\sqrt{n \log q \log m})$.*

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 [Tsa19], but with different parameters.

Lemma 3.2 (Fully Homomorphic Computation [GV15])**.** *There exists a pair of*

[6]This notion is weaker than the definition of subexponential LWE defined in Chapter 2, where $\mathcal{A}$ is allowed to run in time $2^{o(\lambda)}$ and win with probability $2^{-o(\lambda)}$.

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*

$$\|\mathbf{H}_F\|_\infty \leq m \cdot 2^{O(d)}, \quad \|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \leq m \cdot 2^{O(d)}.$$

- *The running time of* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *is bounded by* $\mathrm{poly}(n, m, \log q, 2^d)$.

The above algorithms are taken from [GV15], which is a variant of similar algorithms proposed by Boneh et al. [BGG+14]. The algorithms in [BGG+14] work for any polynomial-sized circuit F, but $\|\mathbf{H}_F\|_\infty$ and $\|\mathbf{H}_{F,\mathbf{x}}\|_\infty$ become super-polynomial even if the depth of the circuit is shallow (i.e., logarithmic depth). On the other hand, the above algorithms run in polynomial time only when F is of logarithmic depth, but $\|\mathbf{H}_F\|_\infty$ and $\|\mathbf{H}_{F,\mathbf{x}}\|_\infty$ can be polynomially bounded. The latter property is crucial for our purpose.

Modified Noise Distribution. For a distribution χ over $\mathbb{Z}$ and an integer m, we define $\widetilde{\chi^m}$ as follows. To sample from $\widetilde{\chi^m}$, we first sample $\mathbf{x} \leftarrow \chi^m$ and $\mathbf{S} \leftarrow \{-1, 1\}^{m \times m}$ and output $\mathbf{S}\mathbf{x}$. By triangular inequality, it can be seen that if the absolute value of a sample from χ is always bounded by B, the infinity norm of a sample from $\widetilde{\chi^m}$ is always bounded mB. This modified noise distribution is used in the kpABE scheme by Boneh et al. [BGG+14] described in Sec. 3.4.5 for the case of χ being the discrete Gaussian distribution. The modification of the noise is introduced in order to make the security proof work. We refer to their paper for the details.

3.4.5 kpABE Scheme by Boneh et al. [BGG+14]

We will use a variant of the kpABE scheme proposed by Boneh et al. [BGG+14]. We call the scheme BGG + 18 and provide the description of the scheme in the following. We focus on the case where the policies associated with secret keys are limited to circuits

with logarithmic depth rather than arbitrary polynomially bounded depth, so that we can use the evaluation algorithm due to Gorbunov and Vinayagamurthy [GV15] (see Lemma 3.2). This allows us to bound the noise growth during the decryption by a polynomial factor, which is crucial for us as in [AY20].

The scheme supports the circuit class $C_{\ell(\lambda),d(\lambda)}$, which is a set of all circuits with input length $\ell(\lambda)$ and depth at most $d(\lambda)$ with arbitrary $\ell(\lambda) = \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$.

$\mathsf{Setup}(1^\lambda)$: 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 $\mathsf{msk} = \mathbf{A}_{\tau_0}^{-1}$.

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

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\mathsf{T} \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\mathsf{T})$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$: 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 defined as in Sec. 3.4.4 from χ.

2. Compute

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

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 $\mathbf{x}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, \mathsf{ct}_F)$: The decryption algorithm takes as input the master public key mpk, a secret key sk_F for a circuit F, and a ciphertext $\mathsf{ct}_\mathbf{x}$ for an attribute $\mathbf{x}$ and proceeds as follows.

1. Parse $\mathsf{ct}_\mathbf{x} \to (\{\psi_{i,x_i} \in \mathbb{Z}_q^m\}_{i\in[\ell]}, \psi_{2\ell+1} \in \mathbb{Z}_q^m, \psi_{2\ell+2} \in \mathbb{Z}_q)$, and $\mathsf{sk}_F \in \mathbb{Z}^{2m}$. If any of the component is not in the corresponding domain or $F(\mathbf{x}) = 1$, output $\perp$.
2. Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalF}(F, \mathbf{x}, \mathbf{B})$.
3. Concatenate $\{\psi_{i,x_i}\}_{i\in[\ell]}$ to form $\psi_\mathbf{x} = (\psi_{1,x_1}, \ldots, \psi_{\ell,x_\ell})$.
4. Compute
$$\psi' := \psi_{2\ell+2} - [\psi_{2\ell+1} \| \psi_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}]\mathbf{r}^\mathsf{T}.$$
5. Output 0 if $\psi' \in [-B, B]$ and 1 if $[-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$.

Remark 2. We note that the encryption algorithm above computes redundant components $\{\psi_{i,\neg x_i}\}_{i\in[\ell]}$ in the second step, which are discarded in the third step. However, due to this redundancy, the scheme has the following special structure that will be useful for us. Namely, the first and the second steps of the encryption algorithm can be executed without knowing $\mathbf{x}$. Only the third step of the encryption algorithm needs the information of $\mathbf{x}$, where it chooses $\{\psi_{i,x_i}\}_{i\in[\ell]}$ from $\{\psi_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ depending on each bit of $\mathbf{x}$ and then output the former terms along with $\psi_{2\ell+1}$ and $\psi_{2\ell+2}$.

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

$$m = n^{1.1}\log q, \qquad q = 2^{\Theta(\lambda)}, \qquad \chi = \mathsf{SampZ}(3\sqrt{n}),$$

$$\tau_0 = n \log q \log m, \qquad \tau = m^{3.1}\ell \cdot 2^{O(d)} \qquad B = \ell n^2 m^5 \tau \cdot 2^{O(d)}.$$

The parameter n will be chosen depending on whether we need Sel-INDr security or Ada-INDr security for the scheme. If it suffices to have Sel-INDr security, we set $n = \lambda^c$ for some constant $c > 1$. If we need Ada-INDr security, we have to enlarge the parameter to be $n = (\ell\lambda)^c$ in order to compensate for the security loss caused by the complexity leveraging.

We remark that if we were to use the above ABE scheme stand-alone, we would have been able to set q polynomially bounded as in [GV15]. The reason why we set q exponentially large is that we combine the scheme with bilinear maps of order q to lift the ciphertext components to the exponent so that they are "hidden" in some sense. In order to use the security of the bilinear map, we set the group order q to be exponentially large.

The following theorem summarizes the security and efficiency properties of the construction. There are two parameter settings depending on whether we assume subexponential hardness of LWE or not.

Theorem 3.3 (Adapted from [GV15; BGG+14])**.** *Assuming hardness of* $\mathsf{LWE}(n, m, q, \chi)$ *with* $\chi = \mathsf{SampZ}(3\sqrt{n})$ *and* $q = O(2^{n^{1/\epsilon}})$ *for some constant* $\epsilon > 1$*, the above scheme satisfies* Sel-INDr *security (Definition 3.5). Assuming* subexponential *hardness of* $\mathsf{LWE}(n, m, q, \chi)$ *with the same parameters, the above scheme satisfies* Ada-INDr *security (Definition 3.3) with respect to the ciphertext space* $\mathcal{CT} := \mathbb{Z}_q^{m(\ell+1)+1}$

3.4.6 Bilinear Map Preliminaries

Here, we introduce our notation for bilinear maps and the bilinear generic group model following Agrawal and Yamada [AY20], which in turn is based on [BCFG17; BFF+14] for defining generic k-linear groups to the bilinear group settings. The definition closely follows that of Maurer [Mau05], which is equivalent to the alternative formulation by Shoup [Sho97].

Notation on Bilinear Maps. A bilinear group generator GroupGen takes as input 1^λ and outputs a group description $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where q is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order q, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map, and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We require that the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as well as the bilinear map e can be efficiently computed. We employ the implicit representation of group elements: for a matrix $\mathbf{A}$ over $\mathbb{Z}_q$, we define $[\mathbf{A}]_1 := g_1^{\mathbf{A}}$, $[\mathbf{A}]_2 := g_2^{\mathbf{A}}$, $[\mathbf{A}]_T := g_T^{\mathbf{A}}$, where exponentiation is carried out component-wise.

We also use the following less standard notations. For vectors $\mathbf{w} = (w_1, \ldots, w_\ell) \in \mathbb{Z}_q^\ell$ and $\mathbf{v} = (v_1, \ldots, v_\ell) \in \mathbb{Z}_q^\ell$ of the same length, $\mathbf{w} \odot \mathbf{v}$ denotes the vector that is obtained by component-wise multiplications. Namely, $\mathbf{v} \odot \mathbf{w} = (v_1 w_1, \ldots, v_\ell w_\ell)$. When $\mathbf{w} \in (\mathbb{Z}_q^*)^\ell$, $\mathbf{v} \oslash \mathbf{w}$ denotes the vector $\mathbf{v} \oslash \mathbf{w} = (v_1/w_1, \ldots, v_\ell/w_\ell)$. It is easy to verify that for vectors $\mathbf{c}, \mathbf{d} \in \mathbb{Z}_q^\ell$ and $\mathbf{w} \in (\mathbb{Z}_q^*)^\ell$, we have $(\mathbf{c} \odot \mathbf{w}) \odot (\mathbf{d} \oslash \mathbf{w}) = \mathbf{c} \odot \mathbf{d}$. For group elements $[\mathbf{v}]_1 \in \mathbb{G}_1^\ell$ and $[\mathbf{w}]_1 \in \mathbb{G}_2^\ell$, $[\mathbf{v}]_1 \odot [\mathbf{w}]_2$ denotes $([v_1 w_1]_T, \ldots, [v_\ell w_\ell]_T)$, which is efficiently computable from $[\mathbf{v}]_1$ and $[\mathbf{w}]_2$ using the bilinear map e.

Generic Bilinear Group Model. Let $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear group setting, L_1, L_2, and L_T be lists of group elements in $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_T$ respectively, and let $\mathcal{D}$ be a distribution over L_1, L_2, and L_T. The generic group model for a bilinear group setting $\mathbf{G}$ and a distribution $\mathcal{D}$ is described in Fig. 3.1. In this model, the challenger first initializes the lists L_1, L_2, and L_T by sampling the group elements according to $\mathcal{D}$, and the adversary receives handles for the elements in the lists. For $s \in \{1, 2, T\}$, $L_s[h]$ denotes the h-th element in the list L_s. The handle to this element is simply the pair (s, h). An adversary running in the generic bilinear group model can apply group operations and bilinear maps to the elements in the lists. To do this, the adversary has to call the appropriate oracle specifying handles for the input elements. The challenger computes the result of a query, stores it in the corresponding list, and returns to the adversary its (newly created) handle. Handles are not unique (i.e., the same group element may appear

more than once in a list under different handles). As in [AY20], we replace the equality test oracle from Baltico et. al [BCFG17] with the zero-test oracle, which is given a handle (s, h) and returns 1 if $L_s[h] = 0$ and 0 otherwise only for the case of $s = T$.

State: Lists L_1, L_2, L_T over $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_T$ respectively.

Initializations: Lists L_1, L_2, L_T sampled according to distribution $\mathcal{D}$.

Oracles: The oracles provide black-box access to the group operations, the bilinear map, and equalities.

For all $s \in \{1, 2, T\}$: $\mathsf{add}_s(h_1, h_2)$ appends $L_s[h_1] + L_s[h_2]$ to L_s and returns its handle $(s, |L_s|)$.

For all $s \in \{1, 2, T\}$: $\mathsf{neg}_s(h_1, h_2)$ appends $-L_s[h_1]$ to L_s and returns its handle $(s, |L_s|)$.

$\mathsf{map}_e(h_1, h_2)$ appends $e(L_1[h_1], L_2[h_2])$ to L_T and returns its handle $(T, |L_T|)$.

$\mathsf{zt}_T(h)$ returns 1 if $L_T[h] = 0$ and 0 otherwise. All oracles return $\perp$ when given invalid indices.

Figure 3.1: Generic group model for bilinear group setting $\mathbf{G} = (q, \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, e, g_1, g_2)$ and distribution $\mathcal{D}$.

Symbolic Group Model. The symbolic group model for a bilinear group setting $\mathbf{G}$ and a distribution $\mathcal{D}_P$ gives to the adversary the same interface as the corresponding generic group model, except that internally the challenger stores lists of element in the field $\mathbb{Z}_p(X_1, \ldots, X_n)$ instead of lists of group elements, where $X_1, \ldots, X_n$ are indeterminates. The oracles add_s, neg_s, map, and zt computes addition, negation, multiplication, and equality in the field. In our work, we will use the subring $\mathbb{Z}_p[X_1, \ldots, X_n, 1/X_1, \ldots, 1/X_n]$ of the entire field $\mathbb{Z}_p(X_1, \ldots, X_n)$. Note that any element f in $\mathbb{Z}_p[X_1, \ldots, X_n, 1/X_1, \ldots, 1/X_n]$ can be represented as

$$f(X_1, \ldots, X_n) = \sum_{(c_1, \ldots, c_n) \in \mathbb{Z}^n} a_{c_1, \ldots, c_n} X_1^{c_1} \cdots X_n^{c_n}$$

using $\{a_{c_1, \ldots, c_n} \in \mathbb{Z}_p\}_{(c_1, \ldots, c_n) \in \mathbb{Z}^n}$, where we have $a_{c_1, \ldots, c_n} = 0$ for all but finite $(c_1, \ldots, c_n) \in \mathbb{Z}^n$. Note that this expression is unique.

3.5 MULTI-INPUT ATTRIBUTE BASED AND PREDICATE ENCRYPTION

We define multi-input Attribute Based Encryption (ABE) and Predicate Encryption (PE) below. Since the only difference between the two notions is in the security game, we unify the syntax for the algorithms in what follows.

A k-input ABE/PE scheme is parametrized over an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ and function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where each function maps $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ to $\{0,1\}$. Such a scheme is described by procedures $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$**:** The Setup algorithm takes as input a security parameter and outputs some public parameters pp and a master secret key msk.

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

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \alpha, b) \to \mathsf{ct}_{\alpha,b,1}$**:** The encryption algorithm for slot 1 takes as input the public parameters pp, a master secret key msk, an attribute $\alpha \in A_\lambda$, and message $b \in \{0,1\}$, and outputs a ciphertext $\mathsf{ct}_{\alpha,b,1}$. For the case of ABE, the attribute string α is included as part of the ciphertext.

$\mathsf{Enc}_i(\mathsf{pp}, \mathsf{msk}, \alpha) \to \mathsf{ct}_{\alpha,i}$ **for** $i \geq 2$**:** The encryption algorithm for the i^{th} slot where $i \in [2,k]$, takes as input the public parameters pp, a master secret key msk, and an attribute $\alpha \in A_\lambda$ and outputs a ciphertext $\mathsf{ct}_{\alpha,i}$. For the case of ABE, the attribute string α is included as part of the ciphertext.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}_{\alpha_1,b,1}, \mathsf{ct}_{\alpha_2,2}, \ldots, \mathsf{ct}_{\alpha_k,k}) \to b'$**:** The decryption algorithm takes as input the public parameters pp, a key for the function f and a sequence of ciphertext of

$(\alpha_1, b), \alpha_2, \ldots, \alpha_k$ and outputs a string b'.

Next, we define correctness and security. For ease of notation, we drop the subscript λ in what follows.

Correctness: For every $\lambda \in \mathbb{N}, b \in \{0,1\}$, $\alpha_1, \ldots, \alpha_k \in A$, $f \in \mathcal{F}$, it holds that if $f(\alpha_1, \ldots, \alpha_k) = 1$, then

$$\Pr\left[\mathsf{Dec}\left(\begin{array}{c}\mathsf{pp}, \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f), \\ \mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \alpha_1, b), \ldots, \mathsf{Enc}_k(\mathsf{pp}, \mathsf{msk}, \alpha_k)\end{array}\right) = b\right] = 1 - \mathrm{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and over the internal randomness of KeyGen and $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_k$.

Definition 3.11 (Ada-IND security for k-ABE)**.** For a k-ABE scheme $\mathsf{k\text{-}ABE} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec}\}$ for an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$, function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and an adversary $\mathcal{A}$, we define the Ada-IND security game, $\mathsf{Expt}^{\mathsf{Ada\text{-}IND}}_{\mathsf{k\text{-}ABE},\mathcal{A}}$ as follows.

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

2. **Query phase:** The challenger samples a bit $\beta \leftarrow \{0,1\}$. During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order.

   a) **Key Queries:** $\mathcal{A}$ makes polynomial number of key queries, say $p = p(\lambda)$. As an i-th key query, $\mathcal{A}$ chooses a function $f_i \in \mathcal{F}_\lambda$. The challenger replies with $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f_i)$.

   b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $p = p(\lambda)$. As an i-th query for a slot $j \in [k]$, $\mathcal{A}$ declares

   $$\begin{cases}(\alpha^i_j, (b^i_0, b^i_1)) & \text{if } j = 1 \\ \alpha^i_j & \text{if } j \neq 1\end{cases}$$

   to the challenger, where $\alpha^i_j \in A_\lambda$ is an attribute and $(b^i_0, b^i_1) \in \{0,1\} \times \{0,1\}$ is the pair of messages. Then, the challenger computes

   $$\mathsf{ct}^i_{j,\beta} = \begin{cases}\mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_j, b^i_\beta) & \text{if } j = 1 \\ \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_j) & \text{if } j \neq 1\end{cases}$$

and returns it 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 for every $f_1, \ldots, f_p \in \mathcal{F}$, it holds that $f_i(\alpha_1^{i_1}, \ldots, \alpha_k^{i_k}) = 0$ for every $i, i_1, \ldots, i_k \in [p]$.

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

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

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

Definition 3.12 (Ada-IND security for k-PE.)**.** For an k-PE scheme $\mathsf{k\text{-}PE} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_k, \mathsf{Dec}\}$ for an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$, function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and an adversary $\mathcal{A}$, we define the Ada-IND security game as follows.

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

2. **Query phase:** The challenger samples a bit $\beta \leftarrow \{0, 1\}$. During the game, $\mathcal{A}$ adaptively makes the following queries, in an arbitrary order.

   a) **Key Queries:** $\mathcal{A}$ makes polynomial number of key queries, say $p = p(\lambda)$. For each key query $i \in [p]$, $\mathcal{A}$ chooses a function $f_i \in \mathcal{F}_\lambda$. The challenger replies with $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, f_i)$.

   b) **Ciphertext Queries:** $\mathcal{A}$ issues polynomial number of ciphertext queries for each slot, say $p = p(\lambda)$. As an i-th query for a slot $j \in [k]$, $\mathcal{A}$ declares

   $$\begin{cases} ((\alpha^i_{j,0}, \alpha^i_{j,1}), (b^i_0, b^i_1)) & \text{if } j = 1 \\ (\alpha^i_{j,0}, \alpha^i_{j,1}) & \text{if } j \neq 1 \end{cases}$$

   to the challenger, where $(\alpha^i_{j,0}, \alpha^i_{j,1})$ is a pair of attributes and (b^i_0, b^i_1) is the

pair of messages. Then, the challenger computes

$$\mathsf{ct}^i_{j,\beta} = \begin{cases} \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_{j,\beta}, b^i_\beta) & \text{if } j = 1 \\ \mathsf{Enc}_j(\mathsf{pp}, \mathsf{msk}, \alpha^i_{j,\beta}) & \text{if } j \neq 1 \end{cases}$$

and returns it 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 for every $f_1, \ldots, f_p \in \mathcal{F}$, it holds that $f_i(\alpha^{i_1}_{1,\beta}, \ldots, \alpha^{i_k}_{k,\beta}) = 0$ for every $i, i_1, \ldots, i_k \in [p]$ and $\beta \in \{0, 1\}$.

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

$$\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) := \left|\Pr[\exp_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) = 1 | \beta = 0] - \Pr[\exp_{\mathsf{k\text{-}PE},\mathcal{A}}(1^\lambda) = 1 | \beta = 1]\right|.$$

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

3.5.1 Strong Security for k-ABE and k-PE

We also consider a stronger security notion for both k-ABE as well as k-PE where the adversary is allowed to make decrypting key requests for ciphertexts so long as they do not distinguish the challenge bit.

Definition 3.13 (Strong Ada-IND security for k-ABE)**.** The definition for strong Ada-IND security for k-ABE is the same as standard Ada-IND security (Definition 3.11) except for the following modification. For the k-ABE adversary to be *admissible* in the *strong* Ada-IND game, we require that

- If $f_i(\alpha^{i_1}_1, \ldots, \alpha^{i_k}_k) = 1$ holds for some $i, i_1, \ldots, i_k \in [p]$, then $b^{i_1}_0 = b^{i_1}_1$.

Let $(\alpha^i, (b^i_0, b^i_1))$ be the i^{th} ciphertext query in slot 1. Then, if $b^i_0 \neq b^i_1$, we call the ciphertext returned by the challenger as a *challenge* ciphertext as it encodes the challenge bit β. Otherwise, we refer to it as *decrypting* ciphertext, as the adversary may potentially request a key to decrypt it.

Definition 3.14 (Strong Ada-IND security for k-PE.)**.** The definition for strong Ada-IND security for k-PE is the same as standard Ada-IND security (Definition 3.12) except for the following modification. For the k-PE adversary to be *admissible* in the *strong* Ada-IND game, we require that

- If $f_i(\alpha_{1,\beta}^{i_1}, \ldots, \alpha_{k,\beta}^{i_k}) = 1$ holds for some $i, i_1, \ldots, i_k \in [p]$ and $\beta \in \{0,1\}$, then $(\alpha_{1,0}^{i_1}, \ldots, \alpha_{k,0}^{i_k}) = (\alpha_{1,1}^{i_1}, \ldots, \alpha_{k,1}^{i_k})$ and $b_0^{i_1} = b_1^{i_1}$.

Let $\big((\alpha_0^i, \alpha_1^i), (b_0^i, b_1^i)\big)$ be the i^{th} ciphertext query in slot 1. Then, if $\alpha_0^i \neq \alpha_1^i$ or $b_0^i \neq b_1^i$, we call the ciphertext returned by the challenger as a *challenge* ciphertext as it encodes the challenge bit β. Otherwise, we refer to it as *decrypting* ciphertext, as the adversary may potentially request a key to decrypt it.

Definition 3.15 (Strong VerSel-IND security for k-ABE and k-PE)**.** The definitions for strong VerSel-IND security for k-ABE and k-PE are the same as strong Ada-IND security above except that the adversary $\mathcal{A}$ is required to submit the challenge queries and secret key queries to the challenger before it samples the public key.

3.5.2 Generalization to Multi-Slot Message Scheme

In the above, we focus our attention on k-ABE and k-PE schemes that only contain a message in a single slot, the remaining slots being free of messages. We can also consider a generalized version of the notions where each slot carries a message and all the messages are recovered in successful decryption. For k polynomial, it is easy to extend a construction with single slot message to the generalized version where each slot contains a message, simply by running k instances of the scheme in parallel and rotating the slot which contains the message in each instance to cover all k slots. Moreover we claim that since the k message scheme is a concatenation of k one message schemes, security of the latter implies security of the former. In more detail, suppose there exists an adversary against the k message scheme with non-negligible advantage ϵ. This can be used to construct an adversary against one of the underlying one message schemes with non-negligible advantage ϵ/k.

3.6 TWO-INPUT ABE FOR NC_1 FROM PAIRINGS AND LWE

In this section, we construct two input ABE for NC_1 circuits. More formally, our construction can support attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$, and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{C_{2\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$, where $C_{2\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $2\ell(\lambda)$ and depth at most $d(\lambda)$. We can prove that the scheme satisfies strong security as per Definition 3.13 assuming LWE in bilinear generic group model. Since the intuition was described in Section 3.1, we proceed directly with the construction. We refer to Sec. 3.4.4 and Sec. 3.4.6 for backgrounds on lattices and pairings respectively and Sec. 3.4.5 for description of the kpABE scheme by Boneh et al. [BGG+14] on which our construction is based.

3.6.1 Construction

We proceed to describe our construction.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda), m = m(\lambda)$, noise distribution χ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, and $B = B(\lambda)$ as specified in Sec. 3.4.5. It samples a group description $\mathbf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. Sets $L := (3\ell + 1)m + 2$ and proceeds as follows.

1. Sample BGG + 18 scheme:
   a) Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
   b) Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Sample $\mathbf{w} \leftarrow (\mathbb{Z}_q^*)^L$.
3. Output $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$, $\quad \mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{w}, [1]_1, [1]_2\right)$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: Given input the public parameters pp, master secret key msk and a circuit F, compute BGG + 18 function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.

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

3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, b)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit b, encryption for slot 1 is defined as follows:

1. Sample LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and noise terms $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{e}_i, \mathbf{e}_{\ell+i,b} \leftarrow \widetilde{\chi^m}$ for $i \in [\ell], b \in \{0,1\}$, where $\widetilde{\chi^m}$ is defined as in Sec. 3.4.4.

2. For $i \in [\ell]$, compute $\psi_i := \mathbf{s}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{e}_i$.

3. For $i \in [\ell+1, 2\ell]$, $b \in \{0,1\}$, compute $\psi_{i,b} := \mathbf{s}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{e}_{i,b}$.

4. Compute $\psi_{2\ell+1} := \mathbf{sA} + \mathbf{e}$ and $\psi_{2\ell+2} := \mathbf{su}^\mathsf{T} + e_0$.

5. Set $\mu = \left\lfloor \frac{q}{2} \right\rceil b$.

6. Set $\mathbf{c} = (1, \{\psi_i\}_{i\in[\ell]}, \{\psi_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \psi_{2\ell+1}, \psi_{2\ell+2} + \mu)$.

7. Sample $t_1 \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_1 = [t_1\mathbf{c} \odot \mathbf{w}]_1$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Let $\mathbf{1}_a := (1, \ldots, 1) \in \mathbb{Z}_q^a$ and $\mathbf{0}_a := (0, \ldots, 0) \in \mathbb{Z}_q^a$. Set

$$\hat{\psi}_{i,b} := \begin{cases} \mathbf{1}_m \in \mathbb{Z}_q^m & \text{if } b = x_{2,i} \\ \mathbf{0}_m \in \mathbb{Z}_q^m & \text{if } b \neq x_{2,i} \end{cases} \quad \text{for } i \in [\ell+1, 2\ell] \text{ and } b \in \{0,1\}.$$

2. Set $\mathbf{d} = (1, \mathbf{1}_{\ell m}, \{\hat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \mathbf{1}_m, 1)$.

3. Sample $t_2 \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_2 = [t_2\mathbf{d} \oslash \mathbf{w}]_2$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: The decryption algorithm takes as input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1 and ct_2 corresponding to the two attributes $\mathbf{x}_1$ and $\mathbf{x}_2$ and proceeds as follows:

1. Take the coordinate-wise pairing between ciphertexts:
Compute $[\mathbf{v}]_T = [t_1t_2\mathbf{c} \odot \mathbf{d}]_T$ as $\mathsf{ct}_1 \odot \mathsf{ct}_2$.

2. De-vectorize obtained vector:
Expand $[\mathbf{v}]_T$ for $i \in [\ell]$, $j \in [\ell+1, 2\ell]$, $b \in \{0,1\}$, to obtain:

$$[v_0]_T = [t_1 t_2]_T,\ [\mathbf{v}_i]_T = [t_1 t_2 \psi_i]_T,$$
$$[\mathbf{v}_{j,b}]_T = [t_1 t_2 \psi'_{j,b}]_T,\ \text{where } \psi'_{j,b} = \begin{cases} \left(\mathbf{s}(\mathbf{B}_j - x_{2,j}\mathbf{G}) + \mathbf{e}_{j,b}\right), & \text{if } b = x_{2,j} \\ \mathbf{0}, & \text{if } b = 1 - x_{2,j} \end{cases},$$
$$[\mathbf{v}_{2\ell+1}]_T = [t_1 t_2 \psi_{2\ell+1}]_T,\ [v_{2\ell+2}]_T = [t_1 t_2 (\psi_{2\ell+2} + \mu)]_T.$$

3. Compute Evaluation function for BGG + 18 ciphertexts in exponent:
Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$. Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

4. Perform BGG + 18 decryption in the exponent:
Form $[\mathbf{v}_\mathbf{x}]_T = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1,x_{2,1}}, \ldots \mathbf{v}_{2\ell,x_{2,\ell}}]_T$ and parse $\mathsf{sk}_F = \mathbf{r}$ as $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$. Then compute

$$[v']_T := [(v_{2\ell+2} - \left(\mathbf{v}_{2\ell+1}\mathbf{r}_1^\mathsf{T} + \mathbf{v}_\mathbf{x}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\mathsf{T}\right))]_T$$

5. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
Find $\eta \in [-B, B] \cup [-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. To speed up the operation, one can employ the baby-step giant-step algorithm.

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

Correctness: To see correctness, we first make following observations:

1. $\mathbf{c} \odot \mathbf{d} = (1, \{\psi_i\}_{i\in[\ell]}, \{\psi'_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \psi_{2\ell+1}, \psi_{2\ell+2} + \mu)$ where,

$$\psi'_{i,b} = \begin{cases} \left(\mathbf{s}(\mathbf{B}_i - x_{2,i}\mathbf{G}) + \mathbf{e}_i\right) & \text{if } b = x_{2,i} \\ \mathbf{0} & \text{if } b = 1 - x_{2,i} \end{cases}.$$

Recall that $[\mathbf{v}]_T = [t_1 t_2 \mathbf{c} \odot \mathbf{d}]_T$. Now, letting $\mathbf{v}_\mathbf{x} = (\mathbf{v}_1, \ldots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1,x_{2,1}}, \ldots \mathbf{v}_{2\ell,x_{2,\ell}})$ and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, on de-vectorizing it the decryptor obtains

$$[v_0]_T = [t_1 t_2]_T,\ [\mathbf{v}_i]_T = [t_1 t_2 (\mathbf{s}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{e}_i)]_T \ \text{ for } i \in [\ell],$$
$$[\mathbf{v}_{i,x_i}]_T = [t_1 t_2 (\mathbf{s}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{e}_{i,x_i})]_T \ \text{ for } i \in [\ell+1, 2\ell],$$
$$[\mathbf{v}_{2\ell+1}]_T = [t_1 t_2 (\mathbf{s}\mathbf{A} + \mathbf{e})]_T,\ \ [v_{2\ell+2}]_T = [t_1 t_2 (\mathbf{s}\mathbf{u}^\mathsf{T} + e_0 + \mu)]$$

2. Next, observe that:

$$\begin{aligned}\mathbf{v}_{\mathbf{x}} &= t_1 t_2(\mathbf{s}(\mathbf{B}_1 - x_1\mathbf{G}) + \mathbf{e}_1, \ldots, \mathbf{s}(\mathbf{B}_{2\ell} - x_{2\ell}\mathbf{G}) + \mathbf{e}_{2\ell}) \\ &= t_1 t_2 \mathbf{s}((\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) - (x_1\mathbf{G}, \ldots, x_{2\ell}\mathbf{G})) + t_1 t_2(\mathbf{e}_1, \ldots, \mathbf{e}_{2\ell}) \\ &= t_1 t_2 \mathbf{s}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + t_1 t_2 \mathbf{e}_{\mathbf{x}}, \\ &\quad \text{where } \mathbf{e}_i = \mathbf{e}_{i,x_i} \text{ for } i \in [\ell+1, 2\ell] \text{ and } \mathbf{e}_{\mathbf{x}} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{2\ell})\end{aligned}$$

3. Performing BGG + 18 evaluation and decryption in the exponent yields:

$$\begin{aligned}[v']_T &= [\left(v_{2\ell+2} - (\mathbf{v}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{v}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)\right)]_T \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu + e_0) - t_1 t_2(\mathbf{s}\mathbf{A} + \mathbf{e})\mathbf{r}_1^\top - t_1 t_2(\mathbf{s}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_{\mathbf{x}})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top]_T \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}(\mathbf{A}\mathbf{r}_1^\top + (\mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^\top)) + t_1 t_2(e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)]_T \\ &\quad (\because (\mathbf{B} - \mathbf{x} \otimes \mathbf{G})\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \text{ (Lemma 3.2)}).\end{aligned}$$

For $F(\mathbf{x}) = 0$, and replacing $\mathbf{B}\mathbf{H}_F$ by $\mathbf{B}_F$, $(\mathbf{r}_1, \mathbf{r}_2)$ by $\mathbf{r}$, we get,

$$\begin{aligned}[v']_T &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top + e')]_T \quad (\text{replacing } (e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top) \text{ by } e') \\ &= [t_1 t_2(\mathbf{s}\mathbf{u}^\top + \mu - \mathbf{s}\mathbf{u}^\top + e')]_T, \quad \text{because } (\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\top = \mathbf{u}. \\ &= [t_1 t_2(\mu + e')]_T = [v_0]_T^{(\mu+e')}.\end{aligned}$$

4. Error bound in v':
Recall that we set $\chi = \mathsf{SampZ}(3\sqrt{n})$. By the definition of SampZ, we have $\|e_0\|_\infty \le 3n$ and $\|\mathbf{e}\|_\infty \le 3n$. Furthermore, we have $\|\mathbf{e}_\ell\|_\infty, \|\mathbf{e}_{\ell+i,b}\|_\infty \le 3mn$ for $i \in [\ell]$ and $b \in \{0,1\}$ by the definition of $\widetilde{\chi^m}$, $\|\mathbf{r}\|_\infty \le \sqrt{n}\tau$, and $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \le m \cdot 2^{O(d)}$, where the last inequality follows from Lemma 3.2. Thus, we have

$$e' = e_0 - \mathbf{e}\mathbf{r}_1^\top - \mathbf{e}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top \le O(\ell m^5 n^{1.5}\tau \cdot 2^{O(d)}) \le B$$

by our choice of B.

5. Finally, since $B = \mathrm{poly}(n, \ell) \cdot 2^{O(d)} = \mathrm{poly}(\lambda)$, we can recover $\eta = \mu + e'$ by brute force search in polynomial time as defined in step 5 and then the message as defined in step 6 of decryption algorithm.

3.6.2 Security

We prove the security via the following theorem.

Theorem 3.4. *Our* 2ABE *scheme for function class* NC_1 *satisfies strong* Ada-IND *security in the generic group model assuming that the* kpABE *scheme* BGG + 18 *for function class* NC_1 *satisfies* Ada-INDr *security.*

Overview. The proof is designed via a sequence of games. To begin, we prove that it is pointless for the adversary to take pairing products between non-matching positions of the ciphertexts of the two parties and then take linear combinations among them. This may be argued because of the randomness $\mathbf{w}$ in the ciphertexts which is only cancelled when matching positions are paired. This enables us to argue that the only possible strategy for the adversary is to take linear combinations among partial decryption results yielded by computing the pairing between matching positions of the ciphertexts. Next we show that taking partial decryption results between matching positions of different pairs of ciphertexts is useless, because the randomness $t_1 t_2$ will change across multiple ciphertexts. This step excludes mix and match attacks between different pairs of ciphertexts and reduces the adversary strategy to gaining information about the message(s) via results obtained by legitimate pairing of two entire ciphertexts. At this point, we invoke the security of $\mathsf{BGG+18}$ to argue that the message is hidden.

Proof. Consider a PPT adversary A that makes at most $Q_{\mathsf{ct}}(\lambda)$ ciphertext queries (in both slots) and $Q_{\mathsf{zt}}(\lambda)$ zero-test queries during the game. We denote the event that A outputs correct guess for the challenge bit β at the end of $\mathbf{Game}_{\mathrm{x}}$ as E_{x}.

$\mathbf{Game}_0$: This is the real game in the generic group model. Without loss of generality, we assume that the challenger simulates the generic group oracle for A. At the beginning of the game, the challenger samples the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and master secret key $\mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{w},\ [1]_1,\ [1]_2\right)$ as described in the scheme. It also samples a random bit β and keeps it with itself. Then, it returns the public parameters pp to A. It handles A's queries as follows:

1. Slot 1 ciphertext queries: To answer the i-th slot 1 ciphertext query $(\mathbf{x}_1^i, b_0^i, b_1^i)$, it samples $t_1^i \leftarrow \mathbb{Z}_q^*$, computes $\mathbf{c}^i = (c_1^i, \ldots, c_L^i)$ as specified by Enc_1 for message b_β^i and returns handles to $\mathsf{ct}_1^i = [t_1^i \mathbf{c}^i \odot \mathbf{w}]_1$.

2. Slot 2 ciphertext queries: To answer the i-th slot 2 ciphertext query $\mathbf{x}_2^i$, the challenger samples $t_2^i \leftarrow \mathbb{Z}_q^*$, computes $\mathbf{d}^i$ as specified by Enc_2 and returns handles to $\mathsf{ct}_2^i = [t_2^i \mathbf{d}^i \oslash \mathbf{w}]_2$.

3. Secret Key queries: To respond to the j-th key query F^j made by A, the challenger computes $\mathbf{r}^j$ as specified in the KeyGen algorithm and returns it to A.

By definition, the advantage of A against the scheme is $\left|\Pr[\mathsf{E}_0] - \frac{1}{2}\right|$.

Game$_1$: In this game, we switch partially to the symbolic group model and change the variables $(w_1, \ldots, w_L)$, $(t_1^1, \ldots, t_1^{Q_{\mathsf{ct}}})$, $(t_2^1, \ldots, t_2^{Q_{\mathsf{ct}}})$ and $(c_1^i, \ldots, c_L^i)$ to formal variables $(W_1, \ldots, W_L)$, $(T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}})$, $(T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}})$, $(C_1^i, \ldots, C_L^i)$. As a result, all handles given to A refer to elements in the ring

$$\mathbf{T} := \mathbb{Z}_q \left[\begin{array}{c} W_1, \ldots, W_L, 1/W_1, \ldots, 1/W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, \\ C_1^1, \ldots, C_L^1, \ldots, C_1^{Q_{\mathsf{ct}}}, \ldots, C_L^{Q_{\mathsf{ct}}} \end{array} \right].$$

where $\{1/W_i\}_i$ are needed to represent the components in the secret keys. However, when the challenger answers the zero-test queries, it substitutes the formal variables with corresponding elements in $\mathbb{Z}_q$. In doing so, if the variable is not assigned a value in $\mathbb{Z}_q$, we sample corresponding value from the same distribution as in the real world. Once a value is assigned to a variable, we use the same value throughout the rest of the game. As we argue in Lemma 3.5, we have:

$$\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1].$$

Here, we list all the components in $\mathbf{T}$ for which corresponding handles are given to A in **Game**$_1$ as handles to the group elements in ciphertexts of both slots:

$$\mathcal{S}_1 := \left\{ \{T_1^i \; C_k^i \; W_k\}_{k\in[L], i\in[Q_{\mathsf{ct}}]} \right\}, \quad \mathcal{S}_2 := \left\{ \{T_2^i / W_k\}_{k\in[L], i\in[Q_{\mathsf{ct}}] \text{ s.t. } d_k^i = 1} \right\}$$

Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ correspond to handles for elements in $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. We then define $\mathcal{S}_T$ as $\mathcal{S}_T := \{X \cdot Y : X \in \mathcal{S}_1, Y \in \mathcal{S}_2, X \cdot Y \neq 0\}$. If we explicitly write down

$\mathcal{S}_T$, we have $\mathcal{S}_T = \mathcal{S}_{T,1} \cup \mathcal{S}_{T,2}$, where

$$\mathcal{S}_{T,1} := \left\{ T_1^i \, T_2^j \, C_k^i \, W_k / W_{k'}, \quad \text{for } k, k' \in [L],\ i, j \in [Q_{\mathsf{ct}}] \right\}$$

$$\text{and } \mathcal{S}_{T,2} := \{ \ T_1^i \, T_2^j \, C_k^i \quad \text{for } k \in [L], i, j \in [Q_{\mathsf{ct}}], \text{ s.t. } d_k^j = 1 \ \}.$$

Note that any handle submitted to the zero-test oracle by A during the game refers to an element f in $\mathbf{T}$ that can be represented as

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, C_1^1, \ldots, C_L^{Q_{\mathsf{ct}}}) = \sum_{Z \in \mathcal{S}_T} a_Z Z \tag{3.1}$$

where the coefficients $\{a_Z \in \mathbb{Z}_q\}_{Z \in \mathcal{S}_T}$ can be efficiently computed. Furthermore, $\{a_Z \in \mathbb{Z}_q\}_{Z \in \mathcal{S}_T}$ satisfying the above equation is unique since all monomials in $\mathcal{S}_T$ are distinct.

Game$_2$: In this game, we use the formal variables $(W_1, \ldots, W_L)$, $(T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}})$, $(T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}})$ even while answering zero test queries. However, $C_1^1, \ldots, C_L^{Q_{\mathsf{ct}}}$ are still replaced by $c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}$. Namely, given a zero test query $f \in \mathbf{T}$, the challenger returns 1 if:

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) = 0. \tag{3.2}$$

We show in Lemma 3.6 that

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq Q_{\mathsf{zt}}(L+3)/q.$$

Game$_3$: In this game, we further change the way zero-test queries are answered. In particular, when A makes a zero-test query for a handle corresponding to $f \in \mathbf{T}$ that can be represented as Eq. (3.1), the challenger returns 0 if there exists $Z \in \mathcal{S}_{T,1}$ such that $a_Z \neq 0$. Otherwise, the challenger answers the query as in the previous game. As we prove in Lemma 3.7, we have $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.

Game$_4$: In this game, we partition the set $\mathcal{S}_{T,2}$ by (i, j) pairs as:

$$\mathcal{S}_{T,2} = \cup_{i,j\in[Q_{\mathsf{ct}}]}\mathcal{S}_{T,2,i,j} \quad \text{where} \quad \mathcal{S}_{T,2,i,j} = \{\ T_1^i\ T_2^j\ C_k^i \quad \text{for } k \in [L] \text{ s.t } d_k^j = 1\ \}.$$

We note the that every term Z in $\mathcal{S}_{T,2,i,j}$ can be represented by the variables T_1^i, T_2^j, and $C_1^i, \ldots, C_L^i$ by the definition of $\mathcal{S}_{T,2,i,j}$. For a zero test query f that is represented as

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, C_1^1, \ldots, C_L^{Q_{\mathsf{ct}}}) = \sum_{i,j\in[Q_{\mathsf{ct}}]} \sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z, \tag{3.3}$$

we change the game so that the challenger returns 0 if there exists a pair (i, j) such that

$$\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \ldots, c_L^i) \neq 0 \quad \text{over } \mathbf{T}. \tag{3.4}$$

As we prove in Lemma 3.8, we have $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Game$_5$: Recall that in the previous game, for a zero test query that is represented as Eq. (3.3), the challenger returns 0 unless Eq. (3.4) holds for all i, j. In this game, for (i, j) such that the i-th ciphertext for slot 1 is a challenge ciphertext (please see Section 3.5.1 to recall the definition of challenge ciphertext), we replace the check with the new one that checks whether

$$\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, C_1^i, \ldots, C_L^i) = 0$$

holds over $\mathbf{T}$. Namely, for such (i, j), we stop replacing the variables $\{C_k^i\}_k$ with the corresponding values $\{c_k^i\}_k$ in $\mathbb{Z}_q$. As we prove in Lemma 3.9, we have $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq \mathrm{negl}(\lambda)$ assuming Ada-INDr security of BGG + 18, which follows from LWE.

Next, we observe that the adversary cannot obtain any information about the encrypted messages in **Game$_5$** since the challenge ciphertexts are replaced by formal variables that do not contain any information of the challenge bit, and the answers to the zero test

queries do not depend on the challenge bit either.

Indistinguishability of Hybrids. We next argue that consecutive hybrids are indistinguishable.

Lemma 3.5 (**Game**$_0 \equiv$ **Game**$_1$)**.** *We have* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. Since zero-test queries in **Game**$_1$ are answered by using variables that are sampled from exactly the same distribution as that in **Game**$_0$, the view of A in **Game**$_1$ is not altered from that in **Game**$_0$. The lemma therefore follows. ■

Lemma 3.6 (**Game**$_1 \approx_s$ **Game**$_2$)**.** *We have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq Q_{\mathsf{zt}}(L+3)/q$.

Proof. The two games are different only when A submits a zero test query corresponding to a polynomial $f \in \mathbb{T}$ such that

$$f(w_1, \ldots, w_L, t_1^1, \ldots, t_1^{Q_{\mathsf{ct}}}, t_2^1, \ldots, t_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) = 0 \tag{3.5}$$

but

$$f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) \neq 0 \tag{3.6}$$

We will use the Schwartz-Zippel lemma to bound the probability that this occurs. We define a new polynomial $g \in \mathbf{T}$ to clear the denominators as:

$$\begin{aligned} &g(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}) \\ &= \left(\prod_{i \in [L]} W_i\right) \cdot f(W_1, \ldots, W_L, T_1^1, \ldots, T_1^{Q_{\mathsf{ct}}}, T_2^1, \ldots, T_2^{Q_{\mathsf{ct}}}, c_1^1, \ldots, c_L^{Q_{\mathsf{ct}}}) \end{aligned}$$

Observe that the polynomial has degree $L + 3$ where L comes from the leading product of W_i and 3 comes from the degree of the terms in $\mathcal{S}_T$. We can bound the probability by $\frac{Q_{\mathsf{zt}}(L+3)}{q}$. ■

Lemma 3.7 (**Game**$_2 \equiv$ **Game**$_3$)**.** *We have* $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.

Proof. We observe that $\mathbf{Game}_2$ and $\mathbf{Game}_3$ differ only when A makes a zero-test query for a handle $f \in \mathbf{T}$ represented as Eq. (3.1) such that $a_Z \neq 0$ for some $Z \in \mathcal{S}_{T,1}$ and corresponding f equals to 0 in $\mathbf{T}$. We claim that such f does not exist and two games are actually equivalent. This is because monomials in $\mathcal{S}_{T,1}$ and $\mathcal{S}_{T,2}$ are distinct even if we replace the formal variables $\{C_j^i\}_{i,j}$ with values $\{c_j^i\}_{i,j}$ in $\mathbb{Z}_q$. Hence, if there exists $Z \in \mathcal{S}_{T,1}$ such that $a_Z \neq 0$, there would be no way to cancel this term using the remaining monomials. ■

Lemma 3.8 ($\mathbf{Game}_3 \approx_s \mathbf{Game}_4$)**.** *We have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq 2Q_{\mathsf{zt}}/q$.

Proof. We observe that the two games differ only when the adversary submits a query $f \in \mathbf{T}$ represented as Eq. (3.1) such that Eq. (3.2) holds, but there is (i, j) such that Eq. (3.4) holds. Note that for such f, i, and j, we have

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i \dots c_L^i) = - \sum_{(i',j') \neq (i,j)} \sum_{Z \in \mathcal{S}_{T,2,i',j'}} a_Z Z(T_1^{i'}, T_2^{j'}, c_1^{i'} \dots c_L^{i'}).$$

However, the above is impossible unless the left hand side equals to 0 since any monomial in $\mathcal{S}_{T,2,i,j}$ never appears in $\mathcal{S}_{T,2,i',j'}$ for $(i, j) \neq (i', j')$ (since the product $T_1^i T_2^j \neq T_1^{i'} T_2^{j'}$) even if we replace the formal variables $\{C_k^i\}_k$ and $\{C_k^{i'}\}_k$ with values $\{c_k^i\}_k$ and $\{c_k^{i'}\}_k$ in $\mathbb{Z}_q$. Therefore, the change made in this game is only conceptual and $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$. ■

Lemma 3.9 ($\mathbf{Game}_4 \approx_c \mathbf{Game}_5$)**.** *There exists a PPT adversary* B *against* $\mathsf{Ada\text{-}INDr}$ *security of* $\mathsf{BGG+18}$ *such that* $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| \leq Q_{\mathsf{ct}}^2 Q_{\mathsf{zt}} \cdot \left(\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG+18},\mathsf{B}}(1^\lambda) + 1/q\right)$.

Proof. We call a zero test query $f \in \mathbf{T}$ as *bad* if f is represented as Eq. (3.3) and there exists (i, j) such that i-th ciphertext for slot 1 is a challenge ciphertext (i.e., $b_0^i \neq b_1^i$) and the following equations hold:

$$\sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \dots, c_L^i) = 0 \quad \text{and} \quad \sum_{Z \in \mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, C_1^i, \dots, C_L^i) \neq 0. \tag{3.7}$$

It is easily seen that $\mathbf{Game}_4$ and $\mathbf{Game}_5$ are equivalent unless the adversary makes a bad query. To bound the probability of a bad query being issued, consider the following sequence of games. Below, we define F_x as the event that the challenger does not abort in $\mathbf{Game}_{4,x}$.

$\mathbf{Game}_{4.0}$: This game is the same as $\mathbf{Game}_4$. However, the challenger checks whether A has made a bad query and aborts if not at the end of the game. By definition, the probability that A makes a bad query in $\mathbf{Game}_4$ is $\Pr[\mathsf{F}_0]$.

$\mathbf{Game}_{4.1}$: In this game, we change the previous game so that the challenger picks a random guess k^* for the first bad query as $k^* \leftarrow [Q_{\mathsf{zt}}]$ at the beginning of the game. Furthermore, we change the game so that the challenger aborts if the k^*-th zero-test query is not the first bad query. Since k^* is chosen uniformly at random and independent from the view of A, the guess is correct with probability $1/Q_{\mathsf{zt}}$ conditioned on F_0. Therefore, we have $\Pr[\mathsf{F}_1] = \Pr[\mathsf{F}_0]/Q_{\mathsf{zt}}$.

$\mathbf{Game}_{4.2}$: This game is the same as the previous game except that the challenger aborts the game immediately after A makes the k^*-th zero-test query. Since the occurrence of F_1 is irrelevant to how the game proceeds after the k^*-th zero-test query, we clearly have $\Pr[\mathsf{F}_2] = \Pr[\mathsf{F}_1]$.

$\mathbf{Game}_{4.3}$: In this game, we change the game so that the the challenger stop aborting even if a bad query occurs before the k^*-th zero test query. Furthermore, any query before the k^*-th one, regardless of whether bad or not, is answered as in $\mathbf{Game}_5$. Since $\mathbf{Game}_4$ and $\mathbf{Game}_5$ proceed exactly the same until the first bad query and removing the abort condition simply increases the chance of making a bad query, we have $\Pr[\mathsf{F}_3] \geq \Pr[\mathsf{F}_2]$.

Game$_{4.4}$**:** In this game, we change the previous game so that the challenger picks $(i^*, j^*) \leftarrow [Q_{\mathsf{ct}}^2]$ uniformly at random at the beginning of the game. Furthermore, we change the abort condition so that the challenger aborts if Eq. (3.7) does not hold with respect to $(i, j) = (i^*, j^*)$ when the k^*-th zero-test query f is represented as Eq. (3.3). We note that the challenger does not check the equations with respect to other indices. Since there exists at least one pair of $(i, j) \in [Q_{\mathsf{ct}}^2]$ that satisfies Eq. (3.7) as long as F_4 occurs and (i^*, j^*) is chosen uniformly at random and independent from the view of A, we have $\Pr[\mathsf{F}_4] \geq \Pr[\mathsf{F}_3]/Q_{\mathsf{ct}}^2$.

Game$_{4.5}$**:** In this game, we have the challenger abort if the (i^*, j^*)-th ciphertext queries were not made at the point when the k^*-th zero-test query was made. We claim that conditioned on F_5 happens, the challenger never aborts. To see this, we observe that if the (i^*, j^*)-th ciphertext queries have not been made, then terms that contain $T_1^{i^*}, T_2^{j^*}$ have not been given to A and there is no way to make a zero-test query for f such that $\sum_{Z \in \mathcal{S}_{T,2,i^*,j^*}} a_Z Z \neq 0$, since all terms in $\mathcal{S}_{T,2,i^*,j^*}$ are multiples of $T_1^{i^*} T_2^{j^*}$. We therefore have $\Pr[\mathsf{F}_5] = \Pr[\mathsf{F}_4]$.

Game$_{4.6}$**:** Recall in the previous game, Eq. (3.7) is checked with respect to $(i, j) = (i^*, j^*)$ and $\mathbf{c}^{i^*} = (c_1^{i^*}, \ldots, c_L^{i^*})$ is used there. In this hybrid, we only compute $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ instead of $\mathbf{c}^{i^*}$ for the k^*-th challenge query. Equivalently, we only compute $c_k^{i^*}$ for k such that $d_k^{j^*} = 1$. We claim that the game is still well-defined. To see this, we first observe that the only place in the game where we need actual ciphertexts (not formal variables) is for the k^*-th zero-test query. Furthermore, for the k^*-th zero-test query, we observe that only the terms

$$\left\{ Z(T_1^{i^*}, T_2^{j^*}, c_1^{i^*}, \ldots c_L^{i^*}) \;\middle|\; Z \in \mathcal{S}_{T,2,i^*,j^*} \right\} = \left\{ c_k^{i^*} T_1^{i^*} T_2^{j^*} \;\middle|\; k \in [L] \text{ s.t. } d_k^{j^*} = 1 \right\}$$

are required, where the equality follows from the definition of $\mathcal{S}_{T,2,i^*,j^*}$. We therefore can see that the game is well-defined and the change is only conceptual.

Hence we have $\Pr[\mathsf{F}_6] = \Pr[\mathsf{F}_5]$.

Game$_{4.7}$**:** Recall that $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ constitutes a vector that is obtained by padding the ciphertext of $\mathsf{BGG+18}$ for the attribute $(\mathbf{x}_1^{i^*}, \mathbf{x}_2^{j^*})$ with 1 and 0. In this game, we pick the ciphertext elements in $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ uniformly at random from $\mathbb{Z}_q$ except for the positions that are fixed to be 0 or 1. As we prove in Lemma 3.10, there exists a PPT adversary B such that $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG+18},\mathsf{B}}(1^\lambda) \geq |\Pr[\mathsf{F}_7] - \Pr[\mathsf{F}_6]|$.

We also show in Lemma 3.11 that $\Pr[\mathsf{F}_7] = 1/q$. This allows us to bound $\Pr[\mathsf{F}_0]$ as:

$$\Pr[\mathsf{F}_0] \leq Q_{\mathsf{ct}}^2 Q_{\mathsf{zt}} \cdot (\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG+18},\mathsf{B}}(1^\lambda) + 1/q),$$

where B is a PPT adversary. ■

Lemma 3.10. *There exists a PPT adversary* B *such that* $\mathsf{Adv}^{\mathsf{Ada\text{-}INDr}}_{\mathsf{BGG+18},\mathsf{B}}(1^\lambda) \geq |\Pr[\mathsf{F}_6] - \Pr[\mathsf{F}_7]|$.

Proof. We show that if A can distinguish **Game**$_{4.6}$ from **Game**$_{4.7}$, we can build another adversary B against $\mathsf{Ada\text{-}INDr}$ security of $\mathsf{BGG+18}$. The adversary B acts as the challenger and simulates the game for A.

Setup phase. At the beginning of the game, B is given 1^λ and the master public key of $\mathsf{BGG+18}$ $(\mathbf{A}, \mathbf{B}, \mathbf{u})$ which it returns to A. B also samples $(i^*, j^*) \leftarrow [Q_{\mathsf{ct}}^2]$, $k^* \leftarrow [Q_{\mathsf{zt}}]$, and $\beta \leftarrow \{0, 1\}$ and keeps them secret.

Key Queries. Given the j-th secret key query for F^j made by A, B makes a *secret key query* for F^j to its challenger and is given $\mathbf{r}$ sampled as $\mathbf{r} \leftarrow [\mathbf{A}\|\mathbf{B}_F]_\tau^{-1}(\mathbf{u})$.

Ciphertext Queries. When A makes ciphertext queries, B prepares handles for the ciphertexts components and returns them to A. In more detail, B returns $\{T_1^i C_k^i W_k\}_{k\in[L]}$

for i-th ciphertext query in slot 1 and $\{T_2^i/W_k\}_{k\in[L]}$, s.t. $d_k^i = 1$ for i-th ciphertext query in slot 2.

Generic Group Queries. B honestly handles the queries for the generic group oracle corresponding to addition, negation, and multiplication (bilinear map) made by A by keeping track of the underlying encodings in $\mathbf{T}$ associated with the handles.

When A makes a k-th zero test query that refers to an element $f \in \mathbf{T}$, B returns 0 if f cannot be represented as Eq.(3.3) (i.e., there is $a_Z \neq 0$ such that $Z \in \mathcal{S}_{T,1}$). Otherwise, B proceeds as follows:

1. If f is the k-th zero-test query with $k < k^*$, it runs the following test for all $i, j \in [Q_{\mathsf{ct}}]$.

   a) If the i-th ciphertext query is the challenge query (i.e., $b_0^i \neq b_1^i$), it checks whether $\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z = 0$ or not (without replacing the formal variables $(C_1^i, \ldots, C_L^i)$ in Z with values $(c_1^i, \ldots, c_L^i)$).

   b) If the i-th ciphertext query is not a challenge query, it checks whether the variables $\{C_k^i\}_k$ are already assigned values. If the values are already assigned, it will use the values. Otherwise, it samples $\mathbf{c}^i = \{c_k^i\}_k$ as in $\mathbf{Game}_4$. We then check $\sum_{Z\in\mathcal{S}_{T,2,i,j}} a_Z Z(T_1^i, T_2^j, c_1^i, \ldots, c_L^i) = 0$ or not.

   If all the equations hold, B returns 1 to A. Otherwise, it returns 0.

2. If f is the k^*-th zero-test query, B first checks whether the (i^*, j^*)-th ciphertext queries have already been made and the i^*-th query for slot 1 is the challenge query and aborts otherwise. It then requests the $\mathsf{BGG+18}$ challenger for ciphertexts for attributes $(\mathbf{x}_1^{i^*} \| \mathbf{x}_2^{j^*})$ and message bit $b_\beta^{i^*}$. It constructs $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$ using the received $\mathsf{BGG+18}$ ciphertexts. Then it checks whether Eq. (3.7) holds with respect to $(i, j) = (i^*, j^*)$. As we observed, the above check can be done only given $\mathbf{c}^{i^*} \odot \mathbf{d}^{j^*}$. It outputs 1 if they hold and 0 otherwise.

Analysis. Observe that B simulates $\mathbf{Game}_{4.6}$ if the challenge ciphertext for B is the real one and $\mathbf{Game}_{4.7}$ if it is chosen uniformly at random from the ciphertext space. Therefore, it can be seen that B outputs 1 with probability $\Pr[\mathsf{F}_6]$ if the $\mathsf{BGG+18}$ challenger returned real ciphertexts and $\Pr[\mathsf{F}_7]$ if it returned random. Therefore, B's advantage against $\mathsf{BGG+18}$ is $|\Pr[\mathsf{F}_6] - \Pr[\mathsf{F}_7]|$. This completes the proof of the lemma. ∎

Lemma 3.11. *We have* $\Pr[\mathsf{F}_7] = 1/q$.

Proof. We recall that F_7 occurs only when A makes a zero-test query that refers to a handle f that is represented as Eq. (3.3) and satisfies Eq.(3.7) with respect to $(i, j) = (i^*, j^*)$ for random $\{c_1^{i^*}, \ldots c_L^{i^*}\}$. However, this can happen only with probability at most $1/q$ by the Schwartz-Zippel lemma because f is linear in the variables $C_1^{i^*}, \ldots, C_L^{i^*}$. ■
■

3.7 TWO-INPUT ABE FOR NC_1 IN STANDARD MODEL

In this section, we propose two input ABE construction for NC^1 in the standard model. The construction is shown to be strong very selective secure under the LWE assumption and a variant of bilinear KOALA assumption introduced in [AWY20], which is proven to hold under the bilinear generic group model, assuming function hiding BIPFE is available. Note that function hiding BIPFE can be instantiated from various standard assumptions on bilinear maps including SXDH and DLIN (See Sec. 3.4.3). The construction is similar to both our construction in Sec. 3.6 and the construction in [AWY20] in high level.

3.7.1 Assumption

Here, we introduce a variant of the bilinear KOALA assumption introduced in [AWY20], which in turn is a pairing group variant of the KOALA assumption introduced in [BW19]. The security of our two input ABE scheme in the standard model will be based on the assumption.

Definition 3.16 (Bilinear KOALA Assumption)**.** Let $\mathsf{Samp}_0 = \{\mathsf{Samp}_{0,\lambda}\}_\lambda$ be an efficient sampling algorithm that takes as input an integer q and outputs a string aux and $\mathsf{Samp}_1 = \{\mathsf{Samp}_{1,\lambda}\}_\lambda$ be an efficient sampling algorithm that takes as input an integer q and a string aux and outputs a matrix $\mathbf{V} \in \mathbb{Z}_q^{\ell_1 \times \ell_2}$ with $\ell_1 < \ell_2$. For an efficient adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}$, let us define

$$\mathsf{Adv}^{\mathsf{BKOALA,dist}}_{\mathcal{A},\mathbf{G},\mathsf{Samp}}(\lambda) := |\Pr[\mathcal{A}_\lambda(\mathbf{G}, \mathsf{aux}, [\mathbf{sV}]_2) \to 1] - \Pr[\mathcal{A}_\lambda(\mathbf{G}, \mathsf{aux}, [\mathbf{r}]_2) \to 1]|.$$

where the probabilities are taken over the choice of $\mathbf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\mathbf{V} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{\ell_1}$, $\mathbf{r} \leftarrow \mathbb{Z}_q^{\ell_2}$, and the coin of $\mathcal{A}_\lambda$.

Furthermore, for an efficient adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_\lambda$, we also define

$$\mathsf{Adv}^{\mathsf{BKOALA,find}}_{\mathcal{B},\mathbf{G},\mathsf{Samp}}(\lambda) := \Pr[\mathcal{B}_\lambda(\mathbf{G}, \mathsf{aux}) \to \mathbf{x} \wedge \mathbf{V}\mathbf{x}^\mathsf{T} = \mathbf{0} \wedge \mathbf{x} \neq \mathbf{0}]$$

where the probability is taken over the choice of $\mathbf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\mathbf{V} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, and the coin of $\mathcal{B}_\lambda$.

We say that the bilinear KOALA assumption holds with respect to $\mathsf{GroupGen}$ and efficient samplers $\mathsf{Samp}_0, \mathsf{Samp}_1$ if for any efficient adversary $\mathcal{A}$, there exists another efficient adversary $\mathcal{B}$ and a polynomial function $Q(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{BKOALA,find}}_{\mathcal{B},\mathbf{G},\mathsf{Samp}}(\lambda) \geq \mathsf{Adv}^{\mathsf{BKOALA,dist}}_{\mathcal{A},\mathbf{G},\mathsf{Samp}}(\lambda)/Q(\lambda) - \mathrm{negl}(\lambda).$$

Remark 3. The above assumption is defined with respect to the non-uniform sampling algorithms $\mathsf{Samp}_0, \mathsf{Samp}_1$ and a non-uniform adversary A. All the security assumptions and proofs in this chapter except for the ones that use the above assumption can work in the uniform setting.

Remark 4 (Comparison with [AWY20])**.** Compared to the original version of the bilinear KOALA assumption [AWY20], our assumption allows Samp_0 to be an efficient sampler that possibly samples aux from some structured distribution, whereas Samp_0 is restricted to output a random string in their assumption. The reason why they restrict the distribution is to avoid a kind of attacks that embeds obfuscation into aux. In particular, as observed by the authors, if we relax the above setting so that aux is chosen along with $\mathbf{V}$, there is a concrete attack assuming sufficiently secure obfuscation. In more detail, let us consider a sampler that outputs random $\mathbf{V}$ along with auxiliary information $\mathsf{aux} = \mathcal{O}(C_\mathbf{V})$, which is an obfuscation of circuit $C_\mathbf{V}$ that takes as input group description $\mathbf{G}$ and elements $[\mathbf{v}]_2$ and returns whether $\mathbf{v}$ is in the space spanned by the rows of $\mathbf{V}$

or not. Using $O(C_{\mathbf{V}})$, one can easily distinguish $[\mathbf{sV}]_2$ from $[\mathbf{r}]_2$ with high probability. However, an efficient adversary may not be able to find a vector $\mathbf{x} \neq \mathbf{0}$ that satisfies $\mathbf{V}\mathbf{x}^\mathsf{T} = \mathbf{0}$ even given $O(C_{\mathbf{V}})$, if we use sufficiently strong obfuscator to obfuscate the circuit $C_{\mathbf{V}}$.[7] This specific attack does not work against our assumption above, since $\mathbf{V}$ is chosen *after* aux, rather than chosen at the same time. However, as a safeguard against the future attacks, we assume the assumption to hold for some specific samplers Samp_0 and Samp_1 rather than for general Samp_0 and Samp_1. We note that [AWY20] assumes that the assumption holds for all the efficient samplers Samp_1 rather than a specific one, while restricting Samp_0 to the specific sampler that outputs a random string.

To justify the assumption, [AWY20] proves that the assumption holds in the generic group model. Though they prove the theorem for the case where aux is chosen uniformly at random, the proof does not depend on this fact and the same proof works for the case where aux is chosen from general distributions. Namely, we have the following theorem.

Theorem 3.12 (Adapted from [AWY20])**.** *The bilinear KOALA assumption holds under the bilinear generic group model with respect to all efficient samplers* Samp_0 *and* Samp_1*, where* A *has access to the generic group oracles but* Samp_0 *and* Samp_1 *do not.*

The following lemma is from [AWY20] with slightly different formulation. The lemma essentially says that for a sampler that outputs a set of vectors such that the vectors are individually pseudorandom but mutually correlated, it holds that the vectors appear mutually pseudorandom when they are lifted to the exponent and randomized by vector-wise randomness assuming the bilinear KOALA assumption for the related samplers.

Lemma 3.13 (Adapted from Theorem 4.7 in [AWY20])**.** *Let* $\mathsf{Samp}_0 = \{\mathsf{Samp}_{0,\lambda}\}_\lambda$ *be an efficient sampling algorithm that takes as input an integer* q *and outputs a string* aux *and* $\mathsf{Samp}_1 = \{\mathsf{Samp}_{1,\lambda}\}_\lambda$ *be an efficient sampling algorithm that takes as input an integer* q *and a string* aux *and outputs a set of vectors* $\{\mathbf{u}^{(j)} \in \mathbb{Z}_q^m\}_{j\in[t]}$*. For an efficient adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_\lambda$ *and* $i := i(\lambda) \in \mathbb{N}$*, let us assume that*

$$|\Pr[\mathcal{A}_\lambda(\mathbf{G}, \mathsf{aux}, \mathbf{u}^{(i)}) \to 1] - \Pr[\mathcal{A}_\lambda(\mathbf{G}, \mathsf{aux}, \mathbf{v}) \to 1]| \tag{3.8}$$

[7] The explanation on the attack is taken verbatim from [AWY20, Remark 4.3].

is negligible, where the probabilities are taken over the choice of $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2) \leftarrow \mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\{\mathbf{u}^{(j)}\}_{j\in[t]} \leftarrow \mathsf{Samp}_\lambda(q, \mathsf{aux})$, $\mathbf{v} \leftarrow \mathbb{Z}_q^m$, *and the coin of* $\mathcal{A}_\lambda$. *In the above, we set* $\mathbf{u}^{(i)} := \mathbf{v}$ *if* $i > t$. *Furthermore, assume that the bilinear KOALA assumption holds with respect to* $\mathsf{GroupGen}$, Samp_0, *and* Samp_1' *that runs* Samp_1 *to obtain* $\{\mathbf{u}^{(j)}\}_{j\in[t]}$ *and then outputs*

$$\mathbf{V} = \begin{bmatrix} 1 & \mathbf{u}^{(1)} & & & & \\ & & 1 & \mathbf{u}^{(2)} & & \\ & & & & \ddots & \\ & & & & 1 & \mathbf{u}^{(t)} \end{bmatrix} \in \mathbb{Z}_q^{t\times(1+m)t}. \tag{3.9}$$

Then, for any efficient adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}$,

$$\left|\Pr\left[\mathcal{B}_\lambda\left(\begin{array}{c}\mathbf{G}, \mathsf{aux},\\ \left\{[\gamma^{(j)}]_2, [\gamma^{(j)}\mathbf{u}^{(j)}]_2\right\}_{j\in[t]}\end{array}\right)\to 1\right] - \Pr\left[\mathcal{B}_\lambda\left(\begin{array}{c}\mathbf{G}, \mathsf{aux},\\ \left\{[\gamma^{(j)}]_2, [\mathbf{v}^{(j)}]_2\right\}_{j\in[t]}\end{array}\right)\to 1\right]\right|, \tag{3.10}$$

is negligible, where the probabilities are taken over the choice of $\mathbb{G}$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\{\mathbf{u}^{(j)}\}_{j\in[t]} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, $\gamma^{(j)} \leftarrow \mathbb{Z}_q$, $\mathbf{v}^{(j)} \leftarrow \mathbb{Z}_q^m$ *for* $j \in [t]$, *and the coin of* $\mathcal{B}_\lambda$.

The lemma is shown for the case where aux is chosen uniformly at random in [AWY20], but the same proof works for our more general setting.

3.7.2 Construction

Here, we show our construction of two input ABE for NC^1 in the standard model. The function class that the scheme supports is exactly the same as that of Sec. 3.6. In the construction, we use a BIPFE scheme $\mathsf{BIPFE} = (\mathsf{BIPFE.Setup}, \mathsf{BIPFEKeyGen}, \mathsf{BIPFE.Enc}, \mathsf{BIPFE.Dec})$. We refer to Sec. 3.4.3 for the definition and instantiations of BIPFE.

$\mathsf{Setup}(1^\lambda)$: 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 in Sec. 3.4.5. It samples a group description $\mathbf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. It then proceeds as follows.

1. Sample BGG + 18 scheme:
   a) Sample $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.
   b) Sample random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n \times m})^{2\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Sample BIPFE instances $\mathsf{BIPFE.msk}_1 \leftarrow \mathsf{BIPFE.Setup}(1^\lambda, 1^{m(\ell+1)+2}, 1^2)$ and $\mathsf{BIPFE.msk}_2 \leftarrow \mathsf{BIPFE.Setup}(1^\lambda, 1^{m\ell}, 1^3)$.
3. Output

$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u}), \quad \mathsf{msk} = \left([1]_1, [1]_2, \mathbf{A}_\tau^{-1}, \{\mathsf{BIPFE.msk}_i\}_{i \in \{1,2\}} \right).$$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: Given input the public parameters pp, master secret key msk and a circuit F, compute BGG + 18 function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.
2. Compute $[\mathbf{A} \| \mathbf{B}_F]_\tau^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and sample $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^\mathsf{T} \leftarrow [\mathbf{A} \| \mathbf{B}_F]_\tau^{-1}(\mathbf{u}^\mathsf{T})$.
3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, b)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit b, encryption for slot 1 is defined as follows:

1. Sample LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and noises $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{e}_L, \mathbf{e}_R \leftarrow (\widetilde{\chi^m})^\ell$, where $\widetilde{\chi^m}$ is defined as in Sec. 3.4.4.
2. Compute

$$\begin{aligned}
&\mathbf{c}_{1,1} := \left(1,\ \mathbf{s}\mathbf{A} + \mathbf{e},\ \mathbf{s}\mathbf{u}^\mathsf{T} + e_0 + \left\lceil \frac{q}{2} \right\rceil \cdot b,\ \mathbf{s}\,(\mathbf{B}_L - \mathbf{x}_1 \otimes \mathbf{G}) + \mathbf{e}_L\right), \\
&\mathbf{c}_{1,2} := \mathbf{0}_{m(\ell+1)+2}, \\
&\mathbf{c}_{2,1} := \mathbf{s}\mathbf{B}_R + \mathbf{e}_R, \quad \mathbf{c}_{2,2} := -\mathbf{1}_\ell \otimes \mathbf{s}\mathbf{G}, \quad \mathbf{c}_{2,3} := \mathbf{0}_{m\ell}
\end{aligned}$$

   where we define $\mathbf{B}_L := [\mathbf{B}_1, \ldots, \mathbf{B}_\ell]$ and $\mathbf{B}_R := [\mathbf{B}_{\ell+1}, \ldots, \mathbf{B}_{2\ell}]$.
3. Set $\mathbf{C}_1 = (\mathbf{c}_{1,1}^\mathsf{T}, \mathbf{c}_{1,2}^\mathsf{T})$ and $\mathbf{C}_2 = (\mathbf{c}_{2,1}^\mathsf{T}, \mathbf{c}_{2,2}^\mathsf{T}, \mathbf{c}_{2,3}^\mathsf{T})$.

4. For $i = 1, 2$, compute $\mathsf{BIPFE.ct}_i := \mathsf{BIPFE.Enc}(\mathsf{BIPFE.msk}_i, [\mathbf{C}_i]_1)$.

5. Output $\mathsf{ct}_1 = (\mathsf{BIPFE.ct}_1, \mathsf{BIPFE.ct}_2)$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Sample $t \leftarrow \mathbb{Z}_q$.

2. Compute

$$\mathbf{d}_{1,1} := t \cdot \mathbf{1}_{m(\ell+1)+2}, \quad \mathbf{d}_{1,2} := \mathbf{0}_{m(\ell+1)+2}$$
$$\mathbf{d}_{2,1} := t \cdot \mathbf{1}_{m\ell}, \ \mathbf{d}_{2,2} := t\,(\mathbf{x}_2 \otimes \mathbf{1}_m)\,, \ \mathbf{d}_{2,3} := \mathbf{0}_{m\ell}.$$

3. Set $\mathbf{D}_1 = (\mathbf{d}_{1,1}^\mathsf{T}, \mathbf{d}_{1,2}^\mathsf{T})$ and $\mathbf{D}_2 = (\mathbf{d}_{2,1}^\mathsf{T}, \mathbf{d}_{2,2}^\mathsf{T}, \mathbf{d}_{2,3}^\mathsf{T})$.

4. For $i = 1, 2$, compute $\mathsf{BIPFE.sk}_i := \mathsf{BIPFE.KeyGen}(\mathsf{BIPFE.msk}_i, [\mathbf{D}]_2)$.

5. Output $\mathsf{ct}_2 = (\mathsf{BIPFE.sk}_1, \mathsf{BIPFE.sk}_2)$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: The decryption algorithm takes as input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1 and ct_2 corresponding to the two attributes $\mathbf{x}_1$ and $\mathbf{x}_2$ and proceeds as follows:

1. Parse the ciphertext:
Parse the ciphertexts as $\mathsf{ct}_1 \to (\mathsf{BIPFE.ct}_1, \mathsf{BIPFE.ct}_2)$ and $\mathsf{ct}_2 \to (\mathsf{BIPFE.sk}_1, \mathsf{BIPFE.sk}_2)$.

2. Decrypt the BIPFE ciphertexts:
Compute $[\mathbf{w}_i]_T = \mathsf{BIPFE.Dec}(\mathsf{BIPFE.sk}_i, \mathsf{BIPFE.ct}_i)$ for $i = 1, 2$.

3. Reorganize the obtained vector:
Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$. Reorganize $[\mathbf{w}_i]_T$ for $i = 1, 2$ to obtain:

$$[v_0, \mathbf{v}_{2\ell+1}, v_{2\ell+2}, \mathbf{v}_{\mathbf{x}_1}]_T := [\mathbf{w}_1]_T, \quad [\mathbf{v}_{\mathbf{x}_2}]_T := [\mathbf{w}_2]_T, \quad [\mathbf{v}_\mathbf{x}]_T := [\mathbf{v}_{\mathbf{x}_1}, \mathbf{v}_{\mathbf{x}_2}]_T,$$

where $v_0 \in \mathbb{Z}_q$, $\mathbf{v}_{2\ell+1} \in \mathbb{Z}_q^m$, $v_{2\ell+2} \in \mathbb{Z}_q$, and $\mathbf{v}_{\mathbf{x}_1}, \mathbf{v}_{\mathbf{x}_2} \in \mathbb{Z}_q^{m\ell}$.

4. Evaluate function on $\mathsf{BGG+18}$ ciphertexts in exponent:
Compute $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

5. Perform BGG + 18 decryption in the exponent:
Form $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$. Then compute

$$[v']_T := [(v_{2\ell+2} - (\mathbf{v}_{2\ell+1}\mathbf{r}_1^\top + \mathbf{v}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top))]_T.$$

6. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
Find $\eta \in [-B, B] \cup [-B + \lceil q/2 \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. To speed up the operation, one can employ the baby-step giant-step algorithm.

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

Correctness: Correctness is argued by observing that vectors $\{\mathbf{w}_i\}_{i\in\{1,2\}}$ form the randomized version of the BGG+ ciphertext w.r.t $\mathbf{s}$ on the exponent. Namely, we have

$$\begin{aligned}\mathbf{w}_1 &= \mathbf{c}_{1,1} \odot \mathbf{d}_{1,1} + \mathbf{c}_{1,2} \odot \mathbf{d}_{1,2} \\ &= t\left(1,\ \mathbf{sA} + \mathbf{e},\ \mathbf{su}^\top + e_0 + \left\lceil \frac{q}{2} \right\rceil \cdot b,\ \mathbf{s}\left(\mathbf{B}_L - \mathbf{x}_1 \otimes \mathbf{G}\right) + \mathbf{e}_L\right)\end{aligned}$$

and

$$\begin{aligned}\mathbf{w}_2 &= \mathbf{c}_{2,1} \odot \mathbf{d}_{2,1} + \mathbf{c}_{2,2} \odot \mathbf{d}_{2,2} + \mathbf{c}_{2,3} \odot \mathbf{d}_{2,3} \\ &= t\left(\mathbf{sB}_R + \mathbf{e}_R\right) - t\left(\mathbf{1}_\ell \otimes \mathbf{sG}\right) \odot \left(\mathbf{x}_2 \otimes \mathbf{1}_m\right) \\ &= t\left(\mathbf{s}\left(\mathbf{B}_R - \mathbf{x}_2 \otimes \mathbf{G}\right) + \mathbf{e}_R\right),\end{aligned}$$

where we used $(\mathbf{1}_\ell \otimes \mathbf{sG}) \odot (\mathbf{x}_2 \otimes \mathbf{1}_m) = \mathbf{s}\,(\mathbf{x}_2 \otimes \mathbf{G})$ in the third equation above. Having established above, the rest of the argument for proving the correctness is exactly the same as Sec. 3.6 and thus omitted.

3.7.3 Security

We prove the security of the construction via following theorem:

Theorem 3.14. *Our* 2ABE *scheme for function class* NC_1 *satisfies strong very selective security assuming that* BIPFE *satisfies function hiding security and the bilinear KOALA*

assumption for certain samplers[8] *and the LWE assumption hold.*

Proof. Consider a PPT adversary A that makes at most $Q_{\mathsf{ct}}(\lambda)$ ciphertext queries (in both slots) and $Q_{\mathsf{kq}}(\lambda)$ key queries during the game. We denote the event that A outputs 1 at the end of $\mathbf{Game}_{\mathrm{x}}$ as E_{x}.

$\mathbf{Game}_0$: This is the real game with $\beta = 0$. At the beginning of the game, the adversary declares its challenge queries $\{(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})\}_{i\in[Q_{\mathsf{ct}}]}$ for the first slot, $\{\mathbf{x}_2^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$ for the second slot, and the key queries $\{F^{(j)}\}_{j\in[Q_{\mathsf{kq}}]}$. Then, the challenger samples the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{u})$ and master secret key $\mathsf{msk} = \left([1]_1, [1]_2, \mathbf{A}_\tau^{-1}, \{\mathsf{BIPFE.msk}_i\}_{i\in\{1,2\}}\right)$ as described in the scheme. Then, it computes the ciphertexts $\mathsf{ct}_1^{(i)} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_0^{(i)})$, $\mathsf{ct}_2^{(i)} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2^{(i)})$, for $i \in [Q_{\mathsf{ct}}]$ and secret keys $\mathsf{sk}^{(j)} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F^{(j)})$ for $j \in [Q_{\mathsf{kq}}]$ and returns $\mathsf{pp}, \{\mathsf{ct}_1^{(i)}, \mathsf{ct}_2^{(i)}\}_i, \{\mathsf{sk}^{(j)}\}_j$ to A.

$\mathbf{Game}_j$: This game is defined for $j \le Q_{\mathsf{ct}}$. In this game, $\mathsf{ct}_1^{(i)}$ is computed as

$$\mathsf{ct}_1^{(i)} \leftarrow \begin{cases} \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_1^{(i)}) & \text{if } i \le j \\ \mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1^{(i)}, b_0^{(i)}) & \text{if } i > j \end{cases}$$

Except for the above, the game is the same as $\mathbf{Game}_0$.

We have that $\mathbf{Game}_{Q_{\mathsf{ct}}}$ equals to the real game with $\beta = 1$. Therefore, we have to show that $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{Q_{\mathsf{ct}}}]|$ is negligible. To do so, it suffices to show that $|\Pr[\mathsf{E}_{j-1}] - \Pr[\mathsf{E}_j]|$ is negligible for all $j \in [Q_{\mathsf{ct}}]$. We introduce the following sequence of games to prove this. In the following, we can assume that the j-th ciphertext for slot 1 is the challenge ciphertext (i.e., $b_0^{(j)} \neq b_1^{(j)}$), since otherwise $\mathbf{Game}_{j-1}$ and $\mathbf{Game}_j$ are equivalent.

$\mathbf{Game}_{j-1,0}$: This game is the same as $\mathbf{Game}_{j-1}$. To fix the notation, we describe how the challenger answers the challenge queries below.

[8] We refer to the proof of Lemma 3.16 for the description of the sampler.

1. Slot 1 ciphertext queries: To answer a slot 1 ciphertext request for $(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})$, the challenger samples $\mathbf{s}^{(i)}$, $\mathbf{e}^{(i)}$, $\mathbf{e}_L^{(i)}$, $\mathbf{e}_R^{(i)}$, and $e_0^{(i)}$ as specified and computes the vectors as below:

$$\mathbf{c}_{1,1}^{(i)} := \left(1,\ \mathbf{s}^{(i)}\mathbf{A} + \mathbf{e}^{(i)},\ \mathbf{s}^{(i)}\mathbf{u}^\mathsf{T} + e_0^{(i)} + \left\lceil \frac{q}{2} \right\rceil \cdot b^{(i)},\ \mathbf{s}^{(i)}\left(\mathbf{B}_L - \mathbf{x}_1^{(i)} \otimes \mathbf{G}\right) + \mathbf{e}_L^{(i)}\right),$$
$$\mathbf{c}_{1,2}^{(i)} = \mathbf{0}_{m(\ell+1)+2}$$

where $b^{(i)} = 1$ if $i \leq j-1$ and $b^{(i)} = 0$ otherwise,

$$\mathbf{c}_{2,1}^{(i)} := \mathbf{s}^{(i)}\mathbf{B}_R + \mathbf{e}_R^{(i)}, \quad \mathbf{c}_{2,2}^{(i)} = -\mathbf{1}_\ell \otimes \mathbf{s}^{(i)}\mathbf{G}, \quad \mathbf{c}_{2,3}^{(i)} := \mathbf{0}_{m\ell}.$$

Then, the challenger sets $\mathbf{C}^{(i)}$ and computes $\mathsf{ct}_1^{(i)}$ as specified using the vectors.

2. Slot 2 ciphertext queries: To answer a slot 2 ciphertext request for $\mathbf{x}_2^{(i)}$, the challenger samples $t^{(i)}$ as specified and computes the vectors as below:

$$\mathbf{d}_{1,1}^{(i)} = t^{(i)}\mathbf{1}_{m(\ell+1)+2}, \quad \mathbf{d}_{1,2}^{(i)} = \mathbf{0}_{m(\ell+1)+2},$$
$$\mathbf{d}_{2,1}^{(i)} = t^{(i)}\mathbf{1}_{m\ell}, \quad \mathbf{d}_{2,2}^{(i)} = t^{(i)}\left(\mathbf{x}_2^{(i)} \otimes \mathbf{1}_m\right), \quad \mathbf{d}_{2,3}^{(i)} = \mathbf{0}_{m\ell}.$$

Then, the challenger sets $\mathbf{D}^{(i)}$ and computes $\mathsf{ct}_2^{(i)}$ as specified using the vectors.

By definition, we have

$$\Pr[\mathsf{E}_{j-1}] = \Pr[\mathsf{E}_{j-1,0}].$$

Game$_{j-1,1}$: In this game, we change how the challenger computes the j-th ciphertext for slot 1 and all the ciphertexts for slot 2. In particular, the challenger sets $\mathbf{C}^{(j)}$ by setting the vectors as

$$\mathbf{c}_{1,1}^{(j)} = \mathbf{0}_{m(\ell+1)+2}, \quad \mathbf{c}_{1,2}^{(j)} = \mathbf{1}_{m(\ell+1)+2},$$
$$\mathbf{c}_{2,1}^{(j)} = \mathbf{c}_{2,2}^{(j)} = \mathbf{0}_{m\ell}, \quad \mathbf{c}_{2,3}^{(j)} = \mathbf{1}_{m\ell}.$$

Furthermore, we change the vectors $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as follows:

$$\mathbf{d}_{1,2}^{(i)} = t^{(i)}\left(1, \mathbf{s}^{(j)}\mathbf{A} + \mathbf{e}^{(j)}, \mathbf{s}^{(j)}\mathbf{u}^\mathsf{T} + e_0^{(j)} + \left\lceil \frac{q}{2} \right\rceil \cdot b_0^{(j)}, \mathbf{s}^{(j)}\left(\mathbf{B}_L - \mathbf{x}_1^{(j)} \otimes \mathbf{G}\right) + \mathbf{e}_L^{(j)}\right),$$
$$\mathbf{d}_{2,3}^{(i)} = t^{(i)}\left(\mathbf{s}^{(j)}\left(\mathbf{B}_R - \mathbf{x}_2^{(i)} \otimes \mathbf{G}\right) + \mathbf{e}_R^{(j)}\right).$$

We note that other terms in $\mathbf{D}_1^{(i)}$ and $\mathbf{D}_2^{(i)}$ are unchanged. We show in Lemma 3.15

that

$$|\Pr[\mathsf{E}_{j-1,0}] - \Pr[\mathsf{E}_{j-1,1}]| \leq \mathrm{negl}(\lambda).$$

Game$_{j-1,2}$**:** In this game, we further change $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as follows:

$$\mathbf{d}_{1,2}^{(i)} = t^{(i)} \left(1, \mathbf{c}^{(i)}, c_0^{(i)}, \mathbf{c}_L^{(i)}\right), \quad \mathbf{d}_{2,3}^{(i)} = t^{(i)} \cdot \mathbf{c}_R^{(i)},$$

where $\mathbf{c}^{(i)} \leftarrow \mathbb{Z}_q^m$, $c_0^{(i)} \leftarrow \mathbb{Z}_q$, $\mathbf{c}_L^{(i)}, \mathbf{c}_R^{(i)} \leftarrow \mathbb{Z}_q^{m\ell}$ are freshly chosen for each i. We show in Lemma 3.16 that

$$|\Pr[\mathsf{E}_{j-1,1}] - \Pr[\mathsf{E}_{j-1,2}]| \leq \mathrm{negl}(\lambda).$$

Game$_{j-1,3}$**:** This game is the same as **Game**$_j$. We show in Lemma 3.17 that

$$|\Pr[\mathsf{E}_{j-1,2}] - \Pr[\mathsf{E}_{j-1,3}]| \leq \mathrm{negl}(\lambda).$$

Indistinguishability of Hybrids. We next argue that consecutive hybrids are indistinguishable.

Lemma 3.15 (Game$_{j-1,0} \approx_c$ **Game**$_{j-1,1}$**).** *We have* $|\Pr[\mathsf{E}_{j-1,0}] - \Pr[\mathsf{E}_{j-1,1}]| \leq \mathrm{negl}(\lambda)$.

Proof. We observe that $\mathbf{C}_1^{(i)} \boxdot \mathbf{D}_1^{(i')}$ and $\mathbf{C}_2^{(i)} \boxdot \mathbf{D}_2^{(i')}$ in **Game**$_{j-1,0}$ are the same as those in **Game**$_{j-1,1}$ for all the combinations of $i, i' \in [Q_{\mathsf{ct}}]$. We can change $\mathbf{C}_1^{(i)}$ and $\mathbf{D}_1^{(i')}$ in **Game**$_{j-1,0}$ to those of **Game**$_{j-1,1}$ in a computationally indistinguishable way by a straightforward reduction to the function privacy of BIPFE. In the reduction, the reduction algorithm samples msk except for $\mathsf{BIPFE.msk}_0$ and simulates BIPFE secret keys and ciphertexts by the oracle queries. We can also change $\mathbf{C}_2^{(i)}$ and $\mathbf{D}_2^{(i')}$ using the security of BIPFE again. ■

Lemma 3.16 (Game$_{j-1,1} \approx_c$ **Game**$_{j-1,2}$**).** *We have* $|\Pr[\mathsf{E}_{j-1,1}] - \Pr[\mathsf{E}_{j-1,2}]| \leq \mathrm{negl}(\lambda)$.

Proof. We construct an adversary B against the bilinear KOALA assumption with respect to certain samplers that are specified later assuming an adversary A that distinguishes the two games. We first fix the challenge queries $\{(\mathbf{x}_1^{(i)}, b_0^{(i)}, b_1^{(i)})\}_{i\in[Q_{\mathsf{ct}}]}$ and $\{\mathbf{x}_2^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$ and secret key queries $\{F^{(i)}\}_{i\in[Q_{\mathsf{kq}}]}$ that maximizes the distinguishing advantage of A. This is possible because the queries are only dependent on the security parameter and the randomness of A, both of which can be hardwired into A in the non-uniform setting.[9]

We first consider the sampling algorithm Samp_0 that works as follows. In the following, let $\mathsf{BGG+18} = (\mathsf{BGG+18.Setup}, \mathsf{BGG+18.KeyGen}, \mathsf{BGG+18.Enc}, \mathsf{BGG+18.Dec})$ be the kpABE scheme by [BGG+14] that is introduced in Sec. 3.4.5.

$\mathsf{Samp}_0(1^\lambda, q)$: Given the security parameter 1^λ and the modulus q, it works as follows.

1. Run $(\mathsf{BGG+18.mpk}, \mathsf{BGG+18.msk}) \leftarrow \mathsf{BGG+18.Setup}(1^\lambda)$ for the circuit class $C_{2\ell(\lambda),d(\lambda)}$ and modulus q.
2. Run $\mathsf{BGG+18.sk}^{(i)} \leftarrow \mathsf{BGG+18.KeyGen}(\mathsf{BGG+18.mpk}, \mathsf{BGG+18.msk}, F^{(i)})$ for $i \in [Q_{\mathsf{kq}}]$.
3. Output $\mathsf{aux} := (\mathsf{BGG+18.mpk}, \{\mathsf{BGG+18.sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$.

We then define Samp_1 as follows:

$\mathsf{Samp}_1(\mathsf{aux}, q)$: Given the auxiliary information $\mathsf{aux} = (\mathsf{BGG+18.mpk}, \{\mathsf{BGG+18.sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$ and the modulus q, it works as follows.

1. Sample $\mathbf{z} \leftarrow \mathbb{Z}_q^n$, $e_0 \leftarrow \chi$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{e}_k \leftarrow \widetilde{\chi^m}$ for $k \in [\ell]$.
2. Compute

$$\psi_{2\ell+1} := \mathbf{z}\mathbf{A} + \mathbf{e} \in \mathbb{Z}_q^m, \quad \psi_{2\ell+2} := \mathbf{z}\mathbf{u}^\mathsf{T} + e_0 + b_0^{(j)}\lceil q/2\rceil \in \mathbb{Z}_q,$$
$$\text{For all } k \in [2\ell],\ b \in \{0,1\}, \psi_{k,b} := \mathbf{z}(\mathbf{B} - b\mathbf{G}) + \mathbf{e}_k \in \mathbb{Z}_q^m$$

3. Set $\mathbf{u}^{(i)} := \left(\psi_{2\ell+1}, \psi_{2\ell+2}, \left\{\psi_{k,x_{1,k}^{(j)}}\right\}_{k\in[\ell]}, \left\{\psi_{k,x_{2,k}^{(i)}}\right\}_{k\in[\ell]}\right)$.

[9] This is the only proof in the chapter where we need the non-uniform reduction algorithm.

4. Output $\{\mathbf{u}^{(i)}\}_{i\in[Q_{\mathsf{ct}}]}$

We can observe that each $\mathbf{u}^{(i)}$ is distributed as a $\mathsf{BGG+18}$ ciphertext for message $b_0^{(j)}$ and attribute $(\mathbf{x}_1^{(j)}, \mathbf{x}_2^{(i)})$, even though when we consider the joint distribution of $\{\mathbf{u}^{(i)}\}_i$, the set of vectors is mutually correlated. This along with the fact that the j-th ciphertext is the challenge ciphertext (i.e., there is no i and i' such that $F^{(i')}(\mathbf{x}_1^{(j)}, \mathbf{x}_2^{i}) = 0$) imply that each $\mathbf{u}^{(i)}$ is pseudorandom given aux by $\mathsf{Sel\text{-}INDr}$ security of $\mathsf{BGG+18}$ (as per Definition 3.5). Therefore, assuming the bilinear KOALA assumption with respect to $\mathsf{GroupGen}$, Samp_0, and Samp'_1, where Samp'_1 is defined as in Lemma 3.13 from Samp_1 above[10], we have that the following distributions are computationally indistinguishable:

$$\left(\begin{array}{c}\mathbf{G}, \mathsf{aux},\\ \left\{[\gamma^{(i)}]_2, [\gamma^{(i)}\mathbf{u}^{(i)}]_2\right\}_{i\in[Q_{\mathsf{ct}}]}\end{array}\right) \approx_c \left(\begin{array}{c}\mathbf{G}, \mathsf{aux},\\ \left\{[\gamma^{(i)}]_2, [\mathbf{v}^{(i)}]_2\right\}_{i\in[Q_{\mathsf{ct}}]}\end{array}\right)$$

where $\mathbb{G}$ is chosen by $\mathsf{GroupGen}(1^\lambda)$, $\mathsf{aux} \leftarrow \mathsf{Samp}_0(q)$, $\{\mathbf{u}^{(i)}\}_{i\in[t]} \leftarrow \mathsf{Samp}_1(q, \mathsf{aux})$, $\gamma^{(i)} \leftarrow \mathbb{Z}_q$, and $\mathbf{v}^{(i)} \leftarrow \mathbb{Z}_q^m$ for $i \in [Q_{\mathsf{ct}}]$.

To complete the proof, we construct B that distinguishes the above distributions given the adversary A. At the beginning, given the input, B first parses $\mathsf{aux} \to (\mathsf{BGG+18.mpk}, \{\mathsf{BGG+18.sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]})$. B then honestly samples $\{\mathsf{BIPFE.msk}_i\}_{i\in\{1,2\}}$ by itself.

Secret Key Queries: It can answer the secret key queries by A by $\{\mathsf{BGG+18.sk}^{(i)}\}_{i\in[Q_{\mathsf{kq}}]}$.

Ciphertext Queries for Slot 1: For answering the ciphertext queries, the simulation for slot 1 is straightforward. Namely, B samples $\mathbf{s}^{(i)}$, $e_0^{(i)}$, $\mathbf{e}^{(i)}$, $\mathbf{e}_L^{(i)}$, and $\mathbf{e}_R^{(i)}$ for $i \neq j$ and generates the ciphertext $\mathsf{ct}_1^{(i)}$ for all $i \in [Q_{\mathsf{ct}}]$ as specified in $\mathbf{Game}_{j-1}$ using $\mathsf{BGG+18.mpk}$. Note that $\mathbf{s}^{(j)}$, $e_0^{(j)}$, $\mathbf{e}^{(j)}$, $\mathbf{e}_L^{(j)}$, and $\mathbf{e}_R^{(j)}$ are not necessary for computing $\mathsf{ct}_1^{(j)}$ and are undefined at this point.

Ciphertext Queries for Slot 2: B also has to answer ciphertext queries for slot 2. To

[10]Thus, matrix $\mathbf{V}$ is defined as Eq.(3.9), where vectors $\{\mathbf{u}_{(i)}\}_i$ in the matrix are the vectors output by the Samp_1 algorithm above.

answer the i-th ciphertext query for slot 2, B first sets terms $\mathbf{d}_{1,2}^{(i)}$ and $\mathbf{d}_{2,3}^{(i)}$ as

$$\left[\mathbf{d}_{1,2}^{(i)}, \mathbf{d}_{2,3}^{(i)}\right]_2 := \left[\gamma^{(i)}, \mathbf{v}^{(i)}\right]_2,$$

where $\mathbf{v}^{(i)}$ is either random or $\mathbf{v}^{(i)} = \gamma^{(i)}\mathbf{u}^{(i)}$. The computation of $[\mathbf{D}^{(i)}]_2$ for the terms other than the above terms is straightforward using $[\gamma^{(i)}]_2$ by implicitly setting $t^{(i)} := \gamma^{(i)}$. It then computes the ciphertext $\mathsf{ct}_2^{(i)}$ as in the honest encryption algorithm.

It can be seen that B simulates $\mathbf{Game}_{j-1,4}$ if $\mathbf{v}^{(i)}$ is random and $\mathbf{Game}_{j-1,3}$ if $\mathbf{v}^{(i)} = \gamma^{(i)}\mathbf{u}^{(i)}$, where B implicitly sets

$$\mathbf{s}^{(j)} := \mathbf{z}, \quad e_0^{(j)} := e_0, \quad \mathbf{e}^{(j)} := \mathbf{e}, \quad \left(\mathbf{e}_L^{(j)}, \mathbf{e}_L^{(j)}\right) := (\mathbf{e}_1, \ldots \mathbf{e}_{2\ell}).$$

Therefore, B distinguishes the distributions if A distinguishes the games. ■

Lemma 3.17 ($\mathbf{Game}_{j-1,2} \approx_c \mathbf{Game}_{j-1,3}$)**.** *We have* $|\Pr[\mathsf{E}_{j-1,2}] - \Pr[\mathsf{E}_{j-1,3}]| \leq \mathrm{negl}(\lambda)$.

Proof. This follows by considering the same sequence of games that is used for showing the indistinguishability of $\mathbf{Game}_{j-1,0}$ and $\mathbf{Game}_{j-1,2}$, but in the reverse order and with the difference that $b_0^{(j)}$ is replaced by $b_1^{(j)}$. The indistinguishability between the games follows from the security of BIPFE, LWE, and the bilinear KOALA assumption for the same sampler. ■
■

3.8 COMPILING k-ABE TO k-PE VIA LOCKABLE OBFUSCATION

In this section we describe our compiler to lift k-input ABE to k-input PE. Namely, we construct k-input predicate encryption using k-input ABE and lockable obfuscation. The conversion preserves $\mathsf{Ada\text{-}IND}$ security. The extension of the conversion that preserves strong security is provided in Section 3.9.

3.8.1 Construction

Our construction uses the following building blocks:

1. A secret key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.

2. A Lockable Obfuscator $\mathsf{LO} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$ with lock space $\mathcal{L} = \{0,1\}^m$ and input space $\mathcal{X} = \{0,1\}^n$.

3. A k-input ABE scheme $\mathsf{kABE} = (\mathsf{kABE.Setup}, \mathsf{kABE.KeyGen}, \mathsf{kABE.Enc}_1, \ldots, \mathsf{kABE.Enc}_k, \mathsf{kABE.Dec})$ in which the message bit is associated with the last slot, $\mathsf{kABE.Enc}_k$. We require $k = O(1)$.

   In the construction below, we require the message space of the SKE scheme to be the same as the lock space $\mathcal{L}$ of the lockable obfuscator scheme LO and the message space of kABE to be the same as the key space of SKE.

We now describe the construction of k-input predicate encryption scheme. Our k-input PE construction has the same attribute space and the function class as the underlying k-input ABE, when we consider the function class of NC_1 circuits or polynomial-size circuits.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, the Setup algorithm does the following:

1. Run $(\mathsf{kABE.msk}, \mathsf{kABE.pp}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$.
2. Run $\mathsf{SKE.Setup}(1^\lambda)$ k times and obtain secret keys $K_1, K_2, \ldots, K_k$.
3. Output $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$ and $\mathsf{pp} = \mathsf{kABE.pp}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, the master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$ and a circuit F, the KeyGen algorithm does the following:

1. Run $\mathsf{kABE.sk}_F \leftarrow \mathsf{kABE.KeyGen}(\mathsf{pp}, \mathsf{kABE.msk}, F)$.
2. Output $\mathsf{sk}_F = \mathsf{kABE.sk}_F$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, m)$: On input the public parameters pp, master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$, attribute $\mathbf{x}_1$ for position 1 and message m, the encryption algorithm does the following:

1. Sample $\gamma_1 \leftarrow \mathcal{L}$ and let $\mathsf{ct}_1^* = \mathsf{SKE.Enc}(K_1, \gamma_1)$

2. Compute $\mathsf{ct}_1 = \mathsf{kABE.Enc}_1(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_1)$.

3. Define a function $f_1[\mathsf{ct}_1, \mathsf{ct}_1^*]$ as in Figure 3.2.

4. Output $\mathsf{ct}'_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}_1^*], m, \gamma_1)$.

$\mathsf{Enc}_i(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_i)$ for $2 \le i \le k$: On input the public parameters pp, master secret key $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$, attribute $\mathbf{x}_i$ for position i, the encryption algorithm does the following:

1. Sample a random value $\gamma_i \leftarrow \mathcal{L}$ and let $\mathsf{ct}_i^* = \mathsf{SKE.Enc}(K_i, \gamma_i)$.

2. Compute $\mathsf{ct}_i = \begin{cases} \mathsf{kABE.Enc}_i(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_i) & \text{for } 2 \le i < k \\ \mathsf{kABE.Enc}_k(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}_k, K_k) & \text{for } i = k \end{cases}$.

3. Define a function $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ as in Figure 3.2.

4. Output $\mathsf{ct}'_i = \mathsf{LO.Obf}(1^\lambda, f_i[\mathsf{ct}_i, \mathsf{ct}_i^*], K_{i-1}, \gamma_i)$.

Circuit $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ **for** $1 \le i \le k$

1. Parse input as $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \tilde{G}_{i+1}, \ldots, \tilde{G}_k, \mathsf{sk}_F)$ where ct_j is regarded as a slot j ciphertext of kABE, $\tilde{G}_j$ is regarded as an obfuscated circuit of LO and sk_F is regarded as a kABE secret key.

2. Compute $K'_i = \begin{cases} \mathsf{LO.Eval}(\tilde{G}_{i+1}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \tilde{G}_{i+2}, \ldots, \tilde{G}_k, \mathsf{sk}_F)) & \text{for } 1 \le i < k \\ \mathsf{kABE.Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) & \text{for } i = k \end{cases}$

3. Outputs $\gamma'_i \leftarrow \mathsf{SKE.Dec}(\mathsf{K}'_i, \mathsf{ct}_i^*)$.

Figure 3.2: Circuit Obfuscated by Slot i Encryption for $1 \le i \le k$

$\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct}'_1, \ldots, \mathsf{ct}'_k)$: On input the secret key sk_F for function F, and kPE ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$, do the following:

1. Parse ct'_1 as an LO obfuscation.

2. Compute and output $\mathsf{LO.Eval}(\mathsf{ct}'_1, (\mathsf{ct}'_2, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F))$.

Correctness. To establish correctness, we first prove the following statement:

Claim 3.18. For $\mathbf{x}_1, \ldots, \mathbf{x}_k$ such that $F(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 0$, and $\mathsf{ct}_i, \mathsf{ct}_i^*, \mathsf{ct}_i'$, for $1 \leq i \leq k$, computed as per the scheme,

$$\text{For } 2 \leq i \leq k,\ \mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_{i-1}.$$

Proof. We can prove this by induction.

Base case: For $i = k$, we show that

$$\mathsf{LO.Eval}(\mathsf{ct}_k', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)) = K_{k-1}$$

Proof: Since, $\mathsf{ct}_k' = \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}_k^*], K_{k-1}, \gamma_k)$, from the functionality of LO, $f_k[\mathsf{ct}_k, \mathsf{ct}_k^*]$ is evaluated on input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)$ in the following steps:

1. $\mathsf{kABE.Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) = \mathsf{K}_k$, from the correctness of $\mathsf{kABE.Dec}$
2. Output $\mathsf{SKE.Dec}(K_k, \mathsf{ct}_k^*) = \gamma_k$, from the correctness of $\mathsf{SKE.Dec}$

Since, the output of function $f_k[\mathsf{ct}_k, \mathsf{ct}_k^*]$ matches the lock value in ct_k', $\mathsf{LO.Eval}(\mathsf{ct}_k', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{k-1}, \mathsf{sk}_F)) = K_{k-1}$, from the correctness of LO.

Inductive Step: We show that for $2 \leq i \leq k-1$, if

$$\mathsf{LO.Eval}(\mathsf{ct}_{i+1}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \mathsf{ct}_{i+2}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_i,$$

then

$$\mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_{i-1}.$$

Proof: Recall that $\mathsf{ct}_i' = \mathsf{LO.Obf}(1^\lambda, f_i[\mathsf{ct}_i, \mathsf{ct}_i^*], K_{i-1}, \gamma_i)$. By LO's functionality, $\mathsf{LO.Eval}(\mathsf{ct}_i', (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F))$ first evaluates $f_i[\mathsf{ct}_i, \mathsf{ct}_i^*]$ on input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}_{i+1}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)$ in the following two steps:

1. $\mathsf{LO.Eval}(\mathsf{ct}_{i+1}', (\mathsf{ct}_1, \ldots, \mathsf{ct}_i, \mathsf{ct}_{i+2}', \ldots, \mathsf{ct}_k', \mathsf{sk}_F)) = K_i$, by the induction hypothesis.
2. Output $\mathsf{SKE.Dec}(K_i, \mathsf{ct}_i^*) = \gamma_i$, from the correctness of SKE.

Since, the function output matches with the lock value,

$\mathsf{LO.Eval}(\mathsf{ct}'_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_{i-1}, \mathsf{ct}'_{i+1}, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F)) = K_{i-1}$ from the correctness of $\mathsf{LO.Eval}$.

■

Finally, we observe that the kPE decryption outputs $\mathsf{LO.Eval}(\mathsf{ct}'_1, (\mathsf{ct}'_2, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F))$, where $\mathsf{ct}'_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}^*_1], m, \gamma_1)$. Hence from the functionality of LO, firstly the function $f_1[\mathsf{ct}_1, \mathsf{ct}^*_1]$ is evaluated on input $(\mathsf{ct}'_2, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F)$ in the following steps:

1. Compute $\mathsf{LO.Eval}(\mathsf{ct}'_2, (\mathsf{ct}_1, \mathsf{ct}'_3, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F)) = K_1$.
2. Output $\mathsf{SKE.Dec}(K_1, \mathsf{ct}^*_1) = \gamma_1$ from the correctness of SKE.

Since, $f_1[\mathsf{ct}_1, \mathsf{ct}^*_1]$ evaluates to γ_1, which is the lock value in ct'_1, from the correctness of $\mathsf{LO.Eval}$, we get $\mathsf{LO.Eval}(\mathsf{ct}'_1, (\mathsf{ct}'_2, \ldots, \mathsf{ct}'_k, \mathsf{sk}_F)) = m$ as desired.

3.8.2 Security

We prove that the above construction satisfies Ada-IND security of Definition 3.12 via the following theorem.

Theorem 3.19. *Assume* LO *is a secure lockable obfuscation scheme as per Definition 3.8, that* kABE *is a secure k input* ABE *scheme as per Definition 3.11 and* SKE *is a secure secret key encryption scheme. Then, the* kPE *construction presented above is secure as per Definition 3.12.*

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

Game$_0$: This is the real world.

Game$_1$: In this world, the SKE key K_k encrypted in the kABE ciphertext ct_k is replaced with $\mathbf{0}$.

For $a = 2$ to $k + 1$ define:

Game$_{a.0}$: In this world,

1. For $j \in [1, k-(a-1)]$, ct'_j is computed as in the real world.
2. For $j = k-(a-2)$,

   a) $\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \text{ (i.e. } a > 2) \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \text{ (i.e. } a = 2) \end{cases}$

   b) $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$

   c) $\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j) & \text{if } j > 1 \text{ (i.e. } a < k+1) \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j) & \text{if } j = 1 \text{ (i.e. } a = k+1) \end{cases}$

3. For $j \in [k-(a-3), k]$, ct'_j is generated using LO simulator. In more detail,

   a) $\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases}$

   b) $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$

   c) $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$

Game$_{a.1}$: This is same as **Game**$_{a.0}$, except the following change: In this world, $\mathsf{ct}'_{k-(a-2)}$ is generated using LO simulator.

Indistinguishability of Hybrids. We now show that the consecutive games are indistinguishable. We let E_x denote the event that the adversary $\mathcal{A}$ outputs correct guess for the challenge bit b at the end of **Game**$_x$.

Claim 3.20. Assume that kABE satisfies $\mathsf{Ada\text{-}IND}$ security (Definition 3.11). Then, **Game**$_0$ and **Game**$_1$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between **Game**$_0$ and **Game**$_1$ with non-

negligible probability then there exists an adversary $\mathcal{B}$ who can break Ada-IND security of kABE using $\mathcal{A}$. The reduction is as follows:

1. The kABE challenger samples $(\mathsf{kABE.msk}, \mathsf{kABE.pp}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$, a bit $b' \leftarrow \{0,1\}$ and sends $\mathsf{kABE.pp}$ to $\mathcal{B}$.

2. Upon receiving the public parameters $\mathsf{kABE.pp}$ from the kABE challenger, $\mathcal{B}$ sets $\mathsf{pp} = \mathsf{kABE.pp}$, samples k SKE keys $K_1, \ldots, K_k$ using $\mathsf{SKE.KeyGen}$ and invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$. $\mathcal{B}$ implicitly sets $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)$.

3. $\mathcal{B}$ then responds to key queries and ciphertext queries from $\mathcal{A}$ as follows:

   Key Queries: $\mathcal{B}$ forwards each key query for a function F to kABE challenger and obtains a secret key $\mathsf{kABE.sk}_F$. $\mathcal{B}$ returns $\mathsf{sk}_F = \mathsf{kABE.sk}_F$ to $\mathcal{A}$.

   Ciphertext Queries: Each ciphertext query from $\mathcal{A}$ is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } 1 < j \leq k\text{)} \end{cases}$$

   On receiving a ciphertext query, $\mathcal{B}$ does the following:

   a) If the query is for slot $1 \leq j \leq k-1$,

      - Samples a random value $\gamma_j \leftarrow \mathcal{L}$ and computes $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \gamma_j)$.
      - Sends $\mathbf{x}^i_{j,b}$, as a ciphertext query to the kABE challenger.
      - The kABE challenger returns a ciphertext $\mathsf{ct}_j = \mathsf{kABE.Enc}_j(\mathsf{kABE.pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b})$ for slot j.
      - $\mathcal{B}$ defines the function $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns
        $$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j) & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j) & \text{otherwise} \end{cases}$$

   b) If the query is for slot k

      - Samples $\gamma_k \leftarrow \mathcal{L}$ and computes $\mathsf{ct}^*_k = \mathsf{SKE.Enc}(K_k, \gamma_k)$.
      - Sends $(\mathbf{x}^i_{k,b}, (\mu^i_0 = K_k, \mu^i_1 = \mathbf{0}))$ as ciphertext query to the kABE challenger.
      - The kABE challenger computes and returns a ciphertext ct_k for slot k,

computed as $\mathsf{ct}_k = \mathsf{kABE.Enc}_k(\mathsf{kABE.pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{k,b}, \mu^i_{b'})$.

- $\mathcal{B}$ defines function $f_k[\mathsf{ct}_k, \mathsf{ct}^*_k]$ and computes and returns

$$\mathsf{ct}'_k \quad = \quad \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}^*_k], K_{k-1}, \gamma_k).$$

4. In the end, $\mathcal{A}$ outputs its guess bit $\hat{b}$. If $b = \hat{b}$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$ to the kABE challenger.

We can observe that if the bit b' chosen by kABE challenger is 0, then $\mathcal{B}$ simulated $\mathbf{Game}_0$, else $\mathbf{Game}_1$ with $\mathcal{A}$. This gives us the advantage of $\mathcal{B}$, against kABE challenger, as $|\Pr(b'' = 1|b' = 0) - \Pr(b'' = 1|b' = 1)| = |\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]|$. Hence, assuming Ada-IND security of kABE, we get

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: Observe that the key queries made by $\mathcal{B}$ to the kABE challenger are the same key queries as made by $\mathcal{A}$ to $\mathcal{B}$. Also the attribute in each ciphertext query by $\mathcal{B}$ to kABE challenger is taken from the corresponding ciphertext query by $\mathcal{A}$. Hence, the admissibility of $\mathcal{A}$ implies admissibility of $\mathcal{B}$. ■

Claim 3.21. Assume that SKE is a CPA secure encryption scheme. Then $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_{2.0}]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$ with non-negligible probability, then there exists an adversary $\mathcal{B}$ who can break CPA security of SKE using $\mathcal{A}$. The reduction is as follows:

1. The SKE challenger samples $K \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and a bit $b' \leftarrow \{0, 1\}$ and invokes $\mathcal{B}$.

2. Upon being challenged by SKE challenger, $\mathcal{B}$ does the following:

   a) Samples $(\mathsf{kABE.pp}, \mathsf{kABE.msk}) \leftarrow \mathsf{kABE.Setup}(1^\lambda)$ and SKE keys $K_1, \ldots, K_{k-1}$. Sets $\mathsf{pp} = \mathsf{kABE.pp}$, $\mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_{k-1}, K_k)$,

where $\mathcal{B}$ implicitly sets K_k to be the secret key K sampled by the SKE challenger.

b) Samples a bit b and invokes $\mathcal{A}$ with pp.

c) For each key query for any function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{kABE.KeyGen}(\mathsf{pp}, \mathsf{kABE.msk}, F)$.

d) To answer each ciphertext query which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } 1 < j \leq k\text{),} \end{cases}$$

$\mathcal{B}$ does the following:

i. If the query is for slot $j < k$, $\mathcal{B}$ samples γ_j and computes ct_j and ct^*_j on its own as $\mathsf{ct}_j = \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b})$, $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \gamma_j)$. Defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}^*_1], m^i_b, \gamma_1) & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

ii. If the query is for slot k

A. $\mathcal{B}$ computes $\mathsf{ct}_k = \mathsf{kABE.Enc}_k(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{k,b}, \mathbf{0})$

B. Samples $\gamma_k \leftarrow \mathcal{L}$ and sends $\mu^i_0 = \gamma_k$ and $\mu^i_1 = \mathbf{0}$ as challenge messages to the SKE challenger.

C. SKE challenger returns $\mathsf{ct}^*_k = \mathsf{SKE.Enc}(K_k, \mu^i_{b'})$.

D. $\mathcal{B}$ defines function $f_k[\mathsf{ct}_k, \mathsf{ct}^*_k]$ and returns

$$\mathsf{ct}'_k = \mathsf{LO.Obf}(1^\lambda, f_k[\mathsf{ct}_k, \mathsf{ct}^*_k], K_{k-1}, \gamma_k)$$

to $\mathcal{A}$.

e) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$ to the SKE challenger.

We can observe that if $b' = 0$ then $\mathcal{B}$ simulated $\mathbf{Game}_1$, else $\mathbf{Game}_{2.0}$. Hence, if $\mathcal{A}$ distinguishes between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$, with non negligible probability then $\mathcal{B}$ also

wins against the SKE challenger. Assuming CPA security of SKE, we get

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_{2.0}]| \leq \mathrm{negl}(\lambda).$$

■

Claim 3.22. Assume that LO is a secure lockable obfuscation scheme (Definition 3.8). Then for $2 \leq a \leq k+1$, $\mathbf{Game}_{a.0}$ and $\mathbf{Game}_{a.1}$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_{a.0}] - \Pr[\mathsf{E}_{a.1}]| \leq \mathrm{negl}(\lambda).$$

Proof. Recall that in both the hybrids,

- For $j \in [1, k-(a-1)]$, ct'_j is computed as in the real world.
- For $j \in [k-(a-3), k]$, ct'_j is generated using LO simulator.
- For $j = k-(a-2)$, ct_j and ct^*_j are computed as:

$$\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}), & \text{if } j < k, \text{(i.e. } a > 2) \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}), & \text{if } j = k, \text{(i.e. } a = 2), \end{cases}$$

$$\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$$

The only difference between the two hybrids is in the generation of $\mathsf{ct}'_{k-(a-2)}$ as following. Let $j = k-(a-2)$.

In $\mathbf{Game}_{a.0}$,

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b, \gamma_j), & \text{if } j = 1, \text{(i.e. } a = k+1) \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

In $\mathbf{Game}_{a.1}$,

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|m^i_b|}), & \text{if } j = 1, \text{(i.e. } a = k+1) \\ \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|}), & \text{otherwise} \end{cases}$$

We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_{a.0}$ and $\mathbf{Game}_{a.1}$ then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs, thus breaking the security of LO. The reduction is as follows:

1. The LO challenger samples $b' \leftarrow \{0,1\}$ and starts the game with $\mathcal{B}$. Upon being challenged by the LO challenger, $\mathcal{B}$ does the following:

   a) Samples public parameters and master secret key for kPE as $(\mathsf{pp}, \mathsf{msk} = (\mathsf{kABE.msk}, K_1, \ldots, K_k)) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.

   b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries, to which $\mathcal{B}$ responds as following.

      i. For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.

      ii. To answer each ciphertext query which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{j,0}, \mathbf{x}^i_{j,1}) & \text{(for slot } j > 1\text{),} \end{cases}$$

      $\mathcal{B}$ does the following:

      A. If the query is for slot $j \in [1, k-(a-1)]$,

         - Samples $\gamma_j \leftarrow \mathcal{L}$.
         - Computes ct_j and ct^*_j using msk and $\mathbf{x}^i_{j,b}$ as attribute.
         - Defines function $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ and returns

$$\mathsf{ct}'_j = \begin{cases} \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}^*_1], m^i_b, \gamma_1), & \text{if } j = 1 \\ \mathsf{LO.Obf}(1^\lambda, f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}, \gamma_j), & \text{otherwise} \end{cases}$$

      B. If the query is for slot $j \in [k-(a-3), k]$

         - Computes
           $\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases}$,
           $\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$ and defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$.
         - Returns $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$.

C. If the query is for slot $j = k - (a-2)$

- Computes
$\mathsf{ct}_j = \begin{cases} \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}) & \text{if } j < k \\ \mathsf{kABE.Enc}_j(\mathsf{pp}, \mathsf{kABE.msk}, \mathbf{x}^i_{j,b}, \mathbf{0}) & \text{if } j = k \end{cases}$,
$\mathsf{ct}^*_j = \mathsf{SKE.Enc}(K_j, \mathbf{0})$ and defines $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$.

- If $j = 1$ (i.e. $a = k+1$), sends $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], m^i_b$, else sends $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j], K_{j-1}$ to the LO challenger and receives either an LO obfuscated or a simulated program ct'_j from the LO challenger.

- Returns ct'_j to $\mathcal{A}$.

c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

We can observe that if the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_{a.0}$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_{a.1}$ with $\mathcal{A}$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_{a.0}] - \Pr[\mathsf{E}_{a.1}]| \leq \mathrm{negl}(\lambda).$$

■

Claim 3.23. Assume that SKE is a CPA secure encryption scheme. Then for $2 \leq a \leq k$, $\mathbf{Game}_{a.1}$ and $\mathbf{Game}_{a+1.0}$ are computationally indistinguishable . That is,

$$|\Pr[\mathsf{E}_{a.1}] - \Pr[\mathsf{E}_{a+1.0}]| \leq \mathrm{negl}(\lambda).$$

Proof. The only difference between the two hybrids is in the computation of $\mathsf{ct}^*_{k-(a-1)}$. In $\mathbf{Game}_{a.1}$, $\mathsf{ct}^*_{k-(a-1)} = \mathsf{SKE.Enc}(K_{k-(a-1)}, \gamma_{k-(a-1)})$, while in $\mathbf{Game}_{a+1.0}$, $\mathsf{ct}^*_{k-(a-1)} = \mathsf{SKE.Enc}(K_{k-(a-1)}, \mathbf{0})$. Hence the indistinguishability of the two hybrids follows from the CPA security of SKE. The reduction is similar to that in the proof of indistinguishability between $\mathbf{Game}_1$ and $\mathbf{Game}_{2.0}$. ■

Claim 3.24. $\Pr[\mathsf{E}_{k+1.1}] - \frac{1}{2} = 0$

Proof. In $\mathbf{Game}_{k+1.1}$, all the LO, circuits returned as kPE ciphertexts, are simulated using LO simulator as $\mathsf{ct}'_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_1[\mathsf{ct}_1,\mathsf{ct}^*_1]|}, 1^{|m^i_b|})$ and $\mathsf{ct}'_j = \mathsf{LO.Sim}(1^\lambda, 1^{|f_j[\mathsf{ct}_j,\mathsf{ct}^*_j]|}, 1^{|K_{j-1}|})$ for $2 \leq j \leq k$. Hence, they depend only on the lengths of functions $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$, length of message and length of SKE keys. Length of $f_j[\mathsf{ct}_j, \mathsf{ct}^*_j]$ further depends only on the length of attributes and messages. Since, these lengths are fixed for the scheme, $\{\mathsf{ct}'_j\}_{j\in[k]}$ completely hide the bit b. Hence, $\mathcal{A}$ can do nothing better than a pure guess for bit b in $\mathbf{Game}_{k+1.1}$. ■
■

Applications. The conversion above can be applied to all the multi-input ABE schemes in this work. Here, we focus on the applications to the candidate two input ABE scheme from lattices in Sec. 3.11 and the candidate three input ABE scheme in Sec. 3.10. The other schemes will be discussed in Sec. 3.9 because they satisfy strong (very selective) security and thus we can apply the conversion in Sec. 3.9. A nice property of the PE scheme obtained from the two input ABE scheme in Sec. 3.11 is that it can handle any polynomial-size circuits. Besides, we can expect that it is post-quantum secure, because it does not use pairings and only uses lattice tools. By applying the conversion to the three input ABE scheme in Sec. 3.10, we can obtain a three-input PE scheme that can handle NC_1 circuits.

3.9 TWO-INPUT PE WITH STRONGER SECURITY

In this section we describe our compiler to lift 2-input ABE to 2-input PE that preserves strong security. The conversion uses lockable obfuscation similarly to Sec. 3.8. Unlike the conversion in Sec. 3.8, we do not know how to extend it to general arity k and it is set to be $k = 2$ here. The construction uses the following building blocks:

1. Two instances of 2-input ABE scheme. In one instance the message is associated with encryption for position 2, while in the other instance, the message is associated with the encryption for position 1. We represent the two instances as $\mathsf{2ABE} = (\mathsf{2ABE.Setup}, \mathsf{2ABE.KeyGen}, \mathsf{2ABE.Enc}_1, \mathsf{2ABE.Enc}_2, \mathsf{2ABE.Dec})$

and $\mathsf{2ABE}'$ = $(\mathsf{2ABE}'.\mathsf{Setup}, \mathsf{2ABE}'.\mathsf{KeyGen}, \mathsf{2ABE}'.\mathsf{Enc}_1, \mathsf{2ABE}'.\mathsf{Enc}_2, \mathsf{2ABE}'.\mathsf{Dec})$.

2. A Lockable Obfuscator $\mathsf{Obf} = (\mathsf{LO.Obf}, \mathsf{LO.Eval})$.

3.9.1 Construction

Our two-input PE construction has the same attribute space and the function class as the underlying two-input ABE, when we consider the function class of NC_1 circuits or polynomial-size circuits.

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the Setup algorithm does the following:

1. Run $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and $(\mathsf{2ABE}'.\mathsf{msk}, \mathsf{2ABE}'.\mathsf{pp}) \leftarrow \mathsf{2ABE}'.\mathsf{Setup}(1^\lambda)$.
2. Output $\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, the master secret key msk and a circuit F, the keygen algorithm does the following:

1. Parse msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.
2. Run $\mathsf{2ABE.sk}_F \leftarrow \mathsf{2ABE.KeyGen}(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, F)$ and $\mathsf{2ABE}'.\mathsf{sk}_F \leftarrow \mathsf{2ABE}'.\mathsf{KeyGen}(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, F)$.
3. Output $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F)$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, m)$: On input the public parameters, pp, master secret key msk, attribute $\mathbf{x}_1$ for position 1 and message m, the encryption algorithm does the following:

1. Parses msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and pp as $(\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.
2. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_1)$.
3. Sample $\alpha \leftarrow \mathcal{M}$ and compute $\mathsf{ct}'_1 = \mathsf{2ABE}'.\mathsf{Enc}_1(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}_1, \alpha)$.
4. Define a function $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$, with $\mathsf{ct}_1, \mathsf{ct}'_1$ being hardwired (Figure 3.3).

5. Output $\mathsf{ct}_1'' = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}_1'], m, \alpha)$.

Circuit $f_1[\mathsf{ct}_1, \mathsf{ct}_1']$

1. Parse input as $(\tilde{G}, \mathsf{sk}, \mathsf{sk}')$ where $\tilde{G}$ is regarded as an obfuscated circuit of LO, and sk and sk' are regarded as secret keys of $\mathsf{2ABE}$ and $\mathsf{2ABE}'$ respectively.
2. Compute $r \leftarrow \mathsf{LO.Eval}(\tilde{G}, (\mathsf{ct}_1, \mathsf{sk}))$.
3. Output $\alpha' = \mathsf{2ABE}'.\mathsf{Dec}(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{sk}', \mathsf{ct}_1', r)$.

Figure 3.3: Circuit Obfuscated by Slot 1 Encryption

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$:

1. Parse msk as $(\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})$ and pp as $(\mathsf{2ABE.pp}, \mathsf{2ABE}'.\mathsf{pp})$.
2. Sample $\beta \leftarrow \mathcal{M}$.
3. Compute $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_2, \beta)$.
4. Compute $\mathsf{ct}_2' = \mathsf{2ABE.Enc}_2'(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}_2)$.
5. Define a function $f_2[\mathsf{ct}_2]$, with ct_2 being hardwired, as in Figure 3.4.
6. Output $\mathsf{ct}_2'' = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}_2', \beta)$.

Circuit $f_2[\mathsf{ct}_2]$

1. Parse input as $(\mathsf{ct}_1, \mathsf{sk})$ where ct_1 is regarded as a ciphertext of $\mathsf{2ABE}$ for the first slot and sk is regarded a secret key of $\mathsf{2ABE}$.
2. Output $\beta' \leftarrow \mathsf{2ABE.Dec}(\mathsf{2ABE.pp}, \mathsf{sk}, \mathsf{ct}_1, \mathsf{ct}_2)$.

Figure 3.4: Circuit Obfuscated by Slot 2 Encryption

$\mathsf{Dec}(\mathsf{sk}_F, \mathsf{ct}_1'', \mathsf{ct}_2'')$: On input the secret key sk_F for function F, and $\mathsf{2PE}$ ciphertexts ct_1'' and ct_2'', do the following:

1. Parse sk_F as $(\mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F)$.
2. Output $\mathsf{LO.Eval}(\mathsf{ct}_1'', (\mathsf{ct}_2'', \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F))$.

Correctness. Recall that $\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m, \alpha)$. We claim that

$$f_1[\mathsf{ct}_1, \mathsf{ct}'_1](\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F) = \alpha.$$

This may be argued via the following steps:

1. Recall that $\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta)$ and $f_2[\mathsf{ct}_2](\mathsf{ct}_1, \mathsf{2ABE.sk}_F) = \mathsf{2ABE.Dec}(\mathsf{2ABE.pp}, \mathsf{2ABE.sk}_F, \mathsf{ct}_1, \mathsf{ct}_2) = \beta$. The second equality follows by correctness of 2ABE and the fact that ct_1 and ct_2 encrypt β under attributes $\mathbf{x}_1, \mathbf{x}_2$. Since ct''_2 has lock value β and message value ct'_2, we have by correctness of LO that $\mathsf{LO.Eval}(\mathsf{ct}''_2, (\mathsf{ct}_1, \mathsf{2ABE.sk}_F)) = \mathsf{ct}'_2$.

2. Next, following the description of $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ (Figure 3.3), we evaluate $\mathsf{2ABE}'.\mathsf{Dec}(\mathsf{2ABE}'.\mathsf{sk}_F, \mathsf{ct}'_1, \mathsf{ct}'_2)$ and recover α by correctness of 2ABE′ decryption and the construction of ct'_1 and ct'_2 as encryptions of α under attributes $\mathbf{x}_1, \mathbf{x}_2$.

Thus, we get that $f_1[\mathsf{ct}_1, \mathsf{ct}'_1](\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F) = \alpha$. Now, by correctness of LO, we have that $\mathsf{LO.Eval}(\mathsf{ct}''_1, (\mathsf{ct}''_2, \mathsf{2ABE.sk}_F, \mathsf{2ABE}'.\mathsf{sk}_F)) = m$ as desired. This concludes the proof.

3.9.2 Security

We prove security via the following theorem.

Theorem 3.25. *Assume* LO *is a secure lockable obfuscation scheme as per Definition 3.8 and that* 2ABE *and* 2ABE′ *are secure two input* ABE *schemes satisfying strong security as in Definition 3.13 (resp., strong very selective security as in Definition 3.15). Then, the* 2PE *construction presented above satisfies strong security as per Definition 3.14 (resp., strong very selective security as in Definition 3.15).*

Proof. This proof is more complex than that of Theorem 3.19, because the adversary can make queries for decrypting keys, in which case contents of obfuscated circuits can be revealed. However, as we argue, this leakage does not compromise the security of messages that must remain hidden, because for their corresponding 2ABE ciphertexts, the protecting obfuscators will remain "locked". Moreover, the "unlocked" LO output 2ABE ciphertexts ct'_2 which cannot be used to decrypt slot 1 ciphertexts by admissibility

of the adversary, and hence do not compromise security of the hidden instances. This is in contrast to the previous scheme, where a global secret SKE key K was being output after successful inner 2ABE decryption.

We focus on the case of strong security below. The case of strong very selective security is similar and simpler. The proof proceeds via a sequence of games between the challenger and a PPT adversary $\mathcal{A}$.

Game$_0$: This is the real world.

Game$_1$: This world differs from the previous in the way slot 2 ciphertext queries are answered. Let us recall that each ciphertext query is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2).} \end{cases}$$

Let $\mathcal{S}$ be the set of all those slot 2 ciphertext queries in which $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$. Then in this world, for queries in $\mathcal{S}$, we replace the value β encrypted in 2ABE ciphertext, ct_2 with $\mathbf{0}$.

Game$_2$: This world differs from the previous in the following ways. In this world, in response to queries in set $\mathcal{S}$, ct''_2 is simulated using the LO simulator.

Game$_3$: This world differs from the previous in the following ways. Let $\mathcal{S}'$ be the set of slot 1 ciphertext queries satisfying one of the following two conditions: (i) $\mathbf{x}^i_{1,0} \neq \mathbf{x}^i_{1,1}$ (ii) $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 \neq m^i_1)$. In this world, ct'_1 encrypts $\mathbf{0}$ instead of α for all queries in $\mathcal{S}'$.

Game$_4$: This world differs from the previous in the following ways. In this world, in response to queries in set $\mathcal{S}'$, ct''_1 is simulated using the LO simulator.

Indistinguishability of Hybrids. We now show that the consecutive hybrids are indistinguishable. We let E_x denote the event that the adversary $\mathcal{A}$ outputs correct guess for the challenge bit b at the end of $\mathbf{Game}_x$.

Claim 3.26. Assume that 2ABE satisfies strong Ada-IND security (Definition 3.13). Then, $\mathbf{Game}_0$ and $\mathbf{Game}_1$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can break strong Ada-IND security of 2ABE using $\mathcal{A}$. The reduction is as follows:

1. The 2ABE challenger samples $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and $b' \leftarrow \{0,1\}$ and sends $\mathsf{2ABE.pp}$ to $\mathcal{B}$.

2. Upon receiving the public parameters $\mathsf{2ABE.pp}$ from 2ABE challenger, $\mathcal{B}$ does the following.

   a) Samples $(\mathsf{2ABE'.msk}, \mathsf{2ABE'.pp}) \leftarrow \mathsf{2ABE'.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$. It implicitly sets the master secret key as $(\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE.msk'})$.

   b) Invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$.

3. $\mathcal{B}$ then responds to key queries and challenge queries from $\mathcal{A}$ as follows:

   Key Queries:

   a) For each key query for a function F, from $\mathcal{A}$, $\mathcal{B}$ makes a key query for F to 2ABE challenger and receives $\mathsf{2ABE.sk}_F$ from the challenger.

   b) Computes $\mathsf{2ABE'.sk}_F \leftarrow \mathsf{2ABE'.KeyGen}(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, F)$.

   c) Sets $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$ and returns it to $\mathcal{A}$.

   Ciphertext Queries: Each ciphertext query from $\mathcal{A}$ is of the form

   $$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

   Upon receiving such a query, $\mathcal{B}$ does the following:

a) For slot 1 queries:

i. Samples $\alpha \leftarrow \mathcal{M}$.

ii. Sends $\mathbf{x}^i_{1,b}$ to $\mathsf{2ABE}$ challenger as slot 1 ciphertext query, to which the $\mathsf{2ABE}$ challenger replies with ct_1.

iii. Computes $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1b}, \alpha)$

iv. Defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

v. Returns ct''_1 to $\mathcal{A}$.

b) For slot 2 queries:

i. Computes $\mathsf{ct}'_2 = \mathsf{2ABE'.Enc}_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{2,b})$

ii. Samples $\beta \leftarrow \mathcal{M}$.

iii. If $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$, then sets $\mu^i_0 = \beta, \mu^i_1 = \mathbf{0}$, else sets $\mu^i_0 = \beta, \mu^i_1 = \beta$.

iv. Sends $\mathbf{x}^i_{2,b}, (\mu^i_0, \mu^i_1)$ as ciphertext query for slot 2 to the $\mathsf{2ABE}$ challenger.

v. The $\mathsf{2ABE}$ challenger computes and returns slot 2 ciphertext as

$$\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mu^i_{b'}),$$

where b' is the coin chosen by the challenger, which is fixed throughout the game.

vi. $\mathcal{B}$ defines $f_2[\mathsf{ct}_2]$ and computes

$$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

vii. Returns ct''_2 to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to the $\mathsf{2ABE}$ challenger.

We can observe that if $b' = 0$, then $\mathcal{B}$ simulated $\mathbf{Game}_0$, else $\mathbf{Game}_1$. Hence, if $\mathcal{A}$ can distinguish between the two hybrids, then $\mathcal{B}$ can win against $\mathsf{2ABE}$ challenger.

Assuming strong Ada-IND security of 2ABE, we get

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: We show that if $\mathcal{A}$ is admissible for strong 2PE security then $\mathcal{B}$ is also admissible for strong 2ABE security. Observe that the key queries issued by $\mathcal{B}$ to the 2ABE challenger are the same key queries as issued by $\mathcal{A}$ to $\mathcal{B}$. Consider the challenge queries issued by $\mathcal{B}$. If for some function F for which key query has been made, $F(\mathbf{x}_{1,b}^{j_1}, \mathbf{x}_{2,b}^{j_2}) = 1$ then we need to ensure that $\mu_0^{j_2} = \mu_1^{j_2}$ (since 2ABE encrypts message in slot 2, the message equality condition is required for query index j_2, i.e. corresponding to slot 2). Since, the ciphertext queries with the same attributes are issued by $\mathcal{A}$ to $\mathcal{B}$, admissibility of $\mathcal{A}$ demands that $\mathbf{x}_{2,0}^{j_2} = \mathbf{x}_{2,1}^{j_2}$. But, when $\mathbf{x}_{2,0}^{j_2} = \mathbf{x}_{2,1}^{j_2}$, $\mathcal{B}$ takes $\mu_0^{j_2} = \mu_1^{j_2} = \beta$, as desired. ■

Claim 3.27. Assume that LO is a secure lockable obfuscation scheme (Definition 3.8). Then $\mathbf{Game}_1$ and $\mathbf{Game}_2$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_1$ and $\mathbf{Game}_2$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs using $\mathcal{A}$, thus breaking the security of LO. The reduction is as follows:

1. Upon being challenged by LO challenger, $\mathcal{B}$ does the following:

   a) Samples public parameters and master secret for 2PE as $(\mathsf{pp}, \mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.

   b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries to which $\mathcal{B}$ responds as follows:

      Key Queries:
      For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns

$\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.

Ciphertext Queries: To answer each ciphertext query, which is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

$\mathcal{B}$ does the following:

i. For slot 1 queries:

   A. Samples $\alpha \leftarrow \mathcal{M}$.

   B. Computes

   $$\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{1,b})$$

   and

   $$\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \alpha)$$

   .

   C. Defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

   $$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

   D. Sends ct''_1 to $\mathcal{A}$.

ii. For slot 2 queries:

   A. Computes $\mathsf{ct}'_2 = \mathsf{2ABE'.Enc}_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{2,b})$.

   B. If $\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1}$,

   - Samples $\beta \leftarrow \mathcal{M}$
   - Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
   - Defines $f_2[\mathsf{ct}_2]$ and computes

   $$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

   C. Else if $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$,

   - Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$, defines $f_2[\mathsf{ct}_2]$ and sends it along with ct'_2 to the LO challenger.

- The LO challenger returns either an obfuscated circuit or a simulated circuit which $\mathcal{B}$ sets as ct_2''.

D. Sends ct_2'' to $\mathcal{A}$.

c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

We can observe that if the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_1$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated $\mathbf{Game}_2$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathrm{negl}(\lambda).$$

■

Claim 3.28. Assume that 2ABE′ satisfies strong Ada-IND security (Definition 3.13). Then, $\mathbf{Game}_2$ and $\mathbf{Game}_3$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can break strong Ada-IND security of 2ABE′ using $\mathcal{A}$. The reduction is as follows:

1. The 2ABE′ challenger samples $(\mathsf{2ABE'.msk}, \mathsf{2ABE'.pp}) \leftarrow \mathsf{2ABE'.Setup}(1^\lambda)$ and $b' \leftarrow \{0,1\}$ and sends 2ABE′.pp to $\mathcal{B}$.

2. Upon receiving the public parameters 2ABE′.pp from 2ABE′ challenger, $\mathcal{B}$ does the following.

   a) Samples $(\mathsf{2ABE.msk}, \mathsf{2ABE.pp}) \leftarrow \mathsf{2ABE.Setup}(1^\lambda)$ and sets $\mathsf{pp} = (\mathsf{2ABE.pp}, \mathsf{2ABE'.pp})$. $\mathcal{B}$ implicitly sets $\mathsf{msk} = (\mathsf{2ABE.msk}, \mathsf{2ABE'.msk})$.

   b) Invokes $\mathcal{A}$ with pp as public parameters and chooses a bit $b \leftarrow \{0,1\}$.

3. $\mathcal{B}$ then responds to key queries and ciphertext queries from $\mathcal{A}$ as follows:

Key Queries:

a) Upon receiving a key query for function F from $\mathcal{A}$, $\mathcal{B}$ makes a key query for F to $\mathsf{2ABE'}$ challenger and receives $\mathsf{2ABE'.sk}_F$ from $\mathsf{2ABE'}$ challenger.

b) Computes $\mathsf{2ABE.sk}_F \leftarrow \mathsf{2ABE.KeyGen}(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, F)$.

c) Sets $\mathsf{sk}_F = (\mathsf{2ABE.sk}_F, \mathsf{2ABE'.sk}_F)$ and returns it to $\mathcal{A}$.

Ciphertext Queries: Each ciphertext query i from $\mathcal{A}$ is of the form

$$\begin{cases} (\mathbf{x}^i_{1,0}, \mathbf{x}^i_{1,1}), (m^i_0, m^i_1) & \text{(for slot 1), or} \\ (\mathbf{x}^i_{2,0}, \mathbf{x}^i_{2,1}) & \text{(for slot 2),} \end{cases}$$

Upon receiving such a query, $\mathcal{B}$ does the following:

a) For slot 1 queries:

i. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{1,b})$

ii. Samples $\alpha \leftarrow \mathcal{M}$.

iii. If the $\mathsf{query}_i \in \mathcal{S}'$, i.e.
$(\mathbf{x}^i_{1,0} \neq \mathbf{x}^i_{1,1})$ OR $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 \neq m^i_1))$

- Sets $\mu^i_0 = \alpha$ and $\mu^i_1 = \mathbf{0}$ and sends ciphertext query $(\mathbf{x}^i_{1,b}, (\mu^i_0, \mu^i_1))$ to the $\mathsf{2ABE'}$ challenger.

- The $\mathsf{2ABE'}$ challenger computes and returns slot 1 ciphertext as

$$\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \mu^i_{b'}).$$

iv. Else, if the $\mathsf{query}_i \notin \mathcal{S}'$, i.e.
$(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 = m^i_1)$

- Sets $\mu^i_0 = \alpha$ and $\mu^i_1 = \alpha$ and sends ciphertext query $(\mathbf{x}^i_{1,b}, (\mu^i_0, \mu^i_1))$ to the $\mathsf{2ABE'}$ challenger.

- The $\mathsf{2ABE'}$ challenger computes and returns slot 1 ciphertext as

$$\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \mu^i_{b'}),$$

where b' is the coin chosen by the challenger, which is fixed throughout the game.

v. $\mathcal{B}$ defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

vi. Sends ct''_1 to $\mathcal{A}$.

b) For slot 2 queries:

i. Sends ciphertext query $\mathbf{x}^i_{2,b}$ for slot 2 to the $\mathsf{2ABE}'$ challenger. The $\mathsf{2ABE}'$ challenger computes and returns slot 2 ciphertext as

$$\mathsf{ct}'_2 = \mathsf{2ABE}'.\mathsf{Enc}_2(\mathsf{2ABE}'.\mathsf{pp}, \mathsf{2ABE}'.\mathsf{msk}, \mathbf{x}^i_{2,b}).$$

ii. If the $\mathsf{query}_i \in \mathcal{S}$, i.e. $(\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1})$

- Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$.
- Defines $f_2[\mathsf{ct}_2]$ and simulates $\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|})$.

iii. If the $\mathsf{query}_i \notin \mathcal{S}$, i.e. $(\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1})$

- Samples $\beta \leftarrow \mathcal{M}$ and computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
- Defines $f_2[\mathsf{ct}_2]$ and computes $\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta)$.

iv. Sends ct''_2 to $\mathcal{A}$.

4. In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to the $\mathsf{2ABE}'$ challenger.

We can observe that if $b' = 0$, then $\mathcal{B}$ simulated $\mathbf{Game}_2$, else $\mathbf{Game}_3$. Hence, if $\mathcal{A}$ distinguishes between the two hybrids, then $\mathcal{B}$ wins against $\mathsf{2ABE}'$ challenger.

Assuming strong $\mathsf{Ada\text{-}IND}$ security of $\mathsf{2ABE}'$, we get

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathrm{negl}(\lambda).$$

Admissibility of $\mathcal{B}$: We show that if $\mathcal{A}$ is admissible for strong $\mathsf{2PE}$ security then $\mathcal{B}$ is also admissible for strong $\mathsf{2ABE}'$ security. Observe that the key queries issued by $\mathcal{B}$ are the same key queries as issued by $\mathcal{A}$. Now consider the challenge queries issued by $\mathcal{B}$.

If there is a function F for which key has been queried, such that $F(\mathbf{x}_{1,b}^{j_1}, \mathbf{x}_{2,b}^{j_2}) = 1$ then we need to ensure that $\mu_0^{j_1} = \mu_1^{j_1}$ (since, $\mathsf{2ABE}'$ encrypts message in slot 1, the message equality condition is required for query index j_1). Since, the challenge queries for the same attributes are issued by $\mathcal{A}$ to $\mathcal{B}$, admissibility of $\mathcal{A}$ demands that $\mathbf{x}_{1,0}^{j_1} = \mathbf{x}_{1,1}^{j_1}$ and $m_0^{j_1} = m_1^{j_1}$. But, in this case, $\mathcal{B}$ takes $\mu_0^{j_1} = \mu_1^{j_1} = \alpha$, as desired. ■

Claim 3.29. Assume that LO is a secure lockable obfuscation scheme (Definition 3.8). Then $\mathbf{Game}_3$ and $\mathbf{Game}_4$ are computationally indistinguishable. That is,

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathrm{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ can distinguish between $\mathbf{Game}_3$ and $\mathbf{Game}_4$ with non-negligible probability then there exists an adversary $\mathcal{B}$ who can distinguish between LO obfuscated programs and simulated programs using $\mathcal{A}$, thus breaking the security of LO. The reduction is as follows:

1. Upon being challenged by LO challenger, $\mathcal{B}$ does the following:

   a) Samples public parameters and master secret for $\mathsf{2PE}$ as $(\mathsf{pp}, \mathsf{msk}{=}(\mathsf{2ABE.msk}, \mathsf{2ABE}'.\mathsf{msk})) \leftarrow \mathsf{Setup}(1^\lambda)$ and invokes $\mathcal{A}$ with pp. $\mathcal{B}$ also samples a bit b.

   b) $\mathcal{A}$ issues polynomially many key queries and ciphertext queries to which $\mathcal{B}$ responds as follows:

      Key Queries:
      For each key query for a function F from $\mathcal{A}$, $\mathcal{B}$ returns $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$ to $\mathcal{A}$.

      Ciphertext Queries: To answer each ciphertext query, which is of the form

      $$\begin{cases} (\mathbf{x}_{1,0}^i, \mathbf{x}_{1,1}^i), (m_0^i, m_1^i) & \text{(for slot 1), or} \\ (\mathbf{x}_{2,0}^i, \mathbf{x}_{2,1}^i) & \text{(for slot 2),} \end{cases}$$

      $\mathcal{B}$ does the following:

      i. For slot 1 queries:

         A. Computes $\mathsf{ct}_1 = \mathsf{2ABE.Enc}_1(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}_{1,b}^i)$.

B. If $(\mathbf{x}^i_{1,0} = \mathbf{x}^i_{1,1})$ and $(m^i_0 = m^i_1)$,

- Samples $\alpha \leftarrow \mathcal{M}$.
- Computes $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \alpha)$.
- Computes

$$\mathsf{ct}''_1 = \mathsf{LO.Obf}(1^\lambda, f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b, \alpha).$$

C. Else,

- Computes $\mathsf{ct}'_1 = \mathsf{2ABE'.Enc}_1(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{1,b}, \mathbf{0})$ and defines $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$.
- Sends $f_1[\mathsf{ct}_1, \mathsf{ct}'_1], m^i_b$ to LO challenger.
- The LO challenger returns either an obfuscated or a simulated program, which $\mathcal{B}$ sets as ct''_1.

D. $\mathcal{B}$ sends ct''_1 to $\mathcal{A}$.

ii. For slot 2 queries:

A. Computes $\mathsf{ct}'_2 = \mathsf{2ABE'.Enc}_2(\mathsf{2ABE'.pp}, \mathsf{2ABE'.msk}, \mathbf{x}^i_{2,b})$.

B. If $\mathbf{x}^i_{2,0} = \mathbf{x}^i_{2,1}$,

- Samples $\beta \leftarrow \mathcal{M}$
- Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \beta)$.
- Defines $f_2[\mathsf{ct}_2]$ and computes

$$\mathsf{ct}''_2 = \mathsf{LO.Obf}(1^\lambda, f_2[\mathsf{ct}_2], \mathsf{ct}'_2, \beta).$$

C. Else if $\mathbf{x}^i_{2,0} \neq \mathbf{x}^i_{2,1}$,

- Computes $\mathsf{ct}_2 = \mathsf{2ABE.Enc}_2(\mathsf{2ABE.pp}, \mathsf{2ABE.msk}, \mathbf{x}^i_{2,b}, \mathbf{0})$.
- Defines $f_2[\mathsf{ct}_2]$ and simulates

$$\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|}).$$

D. Sends ct''_2 to $\mathcal{A}$.

c) In the end, $\mathcal{A}$ outputs a bit $\hat{b}$. If $\hat{b} = b$, then $\mathcal{B}$ returns $b'' = 1$, else $b'' = 0$, to LO challenger.

If the LO challenger returned obfuscated programs, then $\mathcal{B}$ simulated **Game**$_3$, else if LO challenger returned simulated programs, then $\mathcal{B}$ simulated **Game**$_4$. Hence, if $\mathcal{A}$ distinguishes between the two games, then so does $\mathcal{B}$ between obfuscated and simulated programs. Assuming LO is secure, we get

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathrm{negl}(\lambda).$$

■

Claim 3.30. $\Pr[\mathsf{E}_4] - \frac{1}{2} = 0$

Proof. We argue that the adversary cannot obtain any information of b in **Game**$_4$. We first observe that the only possible way for the adversary to learn information of b is to make challenge ciphertext queries, since decrypting ciphertexts do not convey any information of b. However, responses to challenge ciphertext queries does not convey any information of b either as we see below. In **Game**$_4$, a challenge ciphertext for slot 1 is computed as $\mathsf{ct}''_1 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_1[\mathsf{ct}_1, \mathsf{ct}'_1]|}, 1^{|m^i_b|})$. The distribution of ct''_1 depends only on the lengths of function $f_1[\mathsf{ct}_1, \mathsf{ct}'_1]$ and message m^i_b. We observe that $|f_1[\mathsf{ct}_1, \mathsf{ct}'_1]|$ further depends only on the lengths of ct_1 and ct'_1, which in turn depends only on the lengths of the underlying message and attribute. Similarly, challenge ciphertext for slot 2 is computed as $\mathsf{ct}''_2 = \mathsf{LO.Sim}(1^\lambda, 1^{|f_2[\mathsf{ct}_2]|}, 1^{|\mathsf{ct}'_2|})$. We can see that ct''_2 does not convey any information of b because of the same reason as above. ■

■

Applications. By applying the above conversion to two input ABE scheme with strong security in Sec. 3.6, we obtain a candidate construction of two input PE scheme with strong security. A caveat here is that the resulting scheme cannot necessarily be proven secure under LWE in the bilinear generic group model as one might expect. The problem here is that our conversion uses the decryption algorithm of the underlying two input ABE

scheme in a non-black box way, which especially uses the code of the group operations. To claim the security of the resulting scheme, we heuristically assume that the two-input ABE scheme in Sec. 3.6 is strongly secure even in the standard model if we implement the bilinear generic group model with concrete well-chosen bilinear group and then apply the above conversion. We note that this kind of heuristic instantiation is widely used in the context of cryptographic hash functions and bilinear maps. We also mention that we can apply the above conversion to the two input ABE scheme in Sec. 3.7. Since the scheme is proven secure in the standard model, the construction does not suffer from the above problem.

3.10 THREE-INPUT ABE FROM PAIRINGS AND LATTICES

In this section, we provide a candidate construction for 3ABE using the structure of [BV22] and [AY20] as discussed in Section 3.1. Leveraging ideas from the Brakerski-Vaikuntanathan construction [BV22], we also obtain a candidate for 2ABE for P – we provide this construction in Section 3.11. Our 3ABE scheme supports NC_1 circuits. More formally, it supports attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$ and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{C_{3\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \mathrm{poly}(\lambda)$ and $d(\lambda) = O(\log \lambda)$, where $C_{3\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $3\ell(\lambda)$ and depth at most $d(\lambda)$.

3.10.1 Construction

The construction is defined as follows:

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, $k = k(\lambda)$, noise distribution χ, $\hat{\chi}$ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, $\tau'_0 = \tau'_0(\lambda)$, $\tau_t = \tau_t(\lambda)$ and $B = B(\lambda)$ as specified in Sec. 3.10.1. It samples a group description $\mathbf{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, [1]_1, [1]_2)$. It then sets $L := (5\ell + 1)m + 1$ and proceeds as follows.

1. Samples BGG + 18 scheme:

a) Samples $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

b) Samples random matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{3\ell}) \leftarrow (\mathbb{Z}_q^{n \times m})^{3\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

2. Samples $w_0 \leftarrow \mathbb{Z}_q^*$, $\mathbf{W} \leftarrow (\mathbb{Z}_q^*)^{k \times L}$.

3. Samples BV scheme:

a) Samples $\mathbf{C}$ along with its trapdoor $\mathbf{C}_{\tau_0'}^{-1}$ as
$(\mathbf{C}, \mathbf{C}_{\tau_0'}^{-1}) \leftarrow \mathsf{TrapGen}(1^{2(\ell+1)n}, 1^k, q)$, where
$\mathbf{C}^\mathsf{T} = (\mathbf{C}_{2\ell+1,0} \| \mathbf{C}_{2\ell+1,1} \| \ldots \| \mathbf{C}_{3\ell,0} \| \mathbf{C}_{3\ell,1} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2}) \in (\mathbb{Z}_q^{k \times n})^{2(\ell+1)}$.

4. Outputs

$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u}), \ \ \mathsf{msk} = \left(\mathbf{A}_{\tau_0}^{-1}, \mathbf{C}_{\tau_0'}^{-1}, w_0, \mathbf{W} \right).$$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, master secret key msk and a circuit F, compute $\mathsf{BGG+18}$ function key for circuit F as follows:

1. Compute $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.

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

3. Output the secret key $\mathsf{sk}_F := \mathbf{r}$.

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, \mu)$: On input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_1$, message bit μ, encryption for slot 1 is defined as follows:

1. Set $\mathbf{m} = \left\lfloor \frac{q}{K} \right\rceil \mu(1, \ldots, 1) \in \mathbb{Z}_q^k$. We define $K = 2\tau_t\sqrt{n}k$. .

2. Samples $\mathsf{LWEsecret}$ $\mathbf{S} \leftarrow \mathbb{Z}_q^{k \times n}$ and error terms $\mathbf{e}_0 \leftarrow \chi^k$, $\mathbf{E} \leftarrow \chi^{k \times m}$, $\mathbf{E}_{i,x_{1,i}} \leftarrow \hat{\chi}^{k \times m}$, for $i \in [\ell]$, and $\mathbf{E}_{i,b} \leftarrow \hat{\chi}^{k \times m}$, for $i \in [\ell+1, 3\ell]$ and $b \in \{0,1\}$.

3. For $i \in [\ell]$, computes

$$\psi_{i,x_{1,i}} := \mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}} \in \mathbb{Z}_q^{k \times m}.$$

4. For $i \in [\ell+1, 3\ell]$, $b \in \{0,1\}$, computes

$$\psi_{i,b} := \mathbf{S}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{E}_{i,b} \in \mathbb{Z}_q^{k \times m}.$$

5. Computes $\psi_{3\ell+1} := \mathbf{SA} + \mathbf{E} \in \mathbb{Z}_q^{k\times m}$ and $\psi_{3\ell+2}^{\mathsf{T}} := \mathbf{Su}^{\mathsf{T}} + \mathbf{e}_0^{\mathsf{T}} \in \mathbb{Z}_q^{k\times 1}$.

6. Sample $\widehat{\mathbf{S}}_{3\ell+1} \leftarrow \mathbb{Z}_q^{n\times m}$, $\widehat{\mathbf{s}}_{3\ell+2} \leftarrow \mathbb{Z}_q^{n}$, $\{\widehat{\mathbf{S}}_{2\ell+i,b}\}_{i\in[\ell],b\in\{0,1\}} \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$, $\widehat{\mathbf{E}} \leftarrow \chi^{k\times m}$, $\widehat{\mathbf{e}}_0 \leftarrow \chi^k$, $\widehat{\mathbf{E}}_{2\ell+i,b} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell], b \in \{0,1\}$.

7. Compute all possible "BV encodings" for slot 3 attribute $\mathbf{x}_3$ and construct $\widehat{\mathbf{C}}_1$ as follows:

$$\widehat{\mathbf{C}}_1 = (\{\psi_{i,x_{1i}}\}_{i\in[\ell]}, \{\psi_{i,b}\}_{\substack{i\in[\ell+1,2\ell],\\ b\in\{0,1\}}}, \{\mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}\}_{\substack{i\in[2\ell+1,3\ell],\\ b\in\{0,1\}}},$$
$$\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1}, \mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}_{3\ell+2}^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}} + \psi_{3\ell+2}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}}) \in \mathbb{Z}_q^{k\times L}$$

   Here, we assume that the entries of $\widehat{\mathbf{C}}_1$ are vectorized in some fixed order.

8. Sample $t_{\mathbf{x}_1} \leftarrow \mathbb{Z}_q^*$ and

9. Output $\mathsf{ct}_1 = ([t_{\mathbf{x}_1} w_0]_1, [t_{\mathbf{x}_1}\widehat{\mathbf{C}}_1 \odot \mathbf{W}]_1$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: On input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Set $\widehat{\mathbf{C}}_2 = (\mathbf{1}_{k\times\ell m}, \{\hat{\psi}_{\ell+i,x_{2,i}}\}_{i\in[\ell]}, \mathbf{1}_{k\times 2\ell m}, \mathbf{1}_{k\times m}, \mathbf{1}_{k\times 1})$, where

$$\hat{\psi}_{\ell+i,b} := \begin{cases} \mathbf{1}_m \in \mathbb{Z}_q^m & \text{if } b = x_{2,i} \\ \mathbf{0}_m \in \mathbb{Z}_q^m & \text{if } b \neq x_{2,i} \end{cases} \quad \text{for } i \in [\ell] \text{ and } b \in \{0,1\}.$$

2. Sample $t_{\mathbf{x}_2} \leftarrow \mathbb{Z}_q^*$ and output $\mathsf{ct}_2 = ([t_{\mathbf{x}_2}/w_0]_2, [t_{\mathbf{x}_2}\widehat{\mathbf{C}}_2 \oslash \mathbf{W}]_2$.

$\mathsf{Enc}_3(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_3)$: Given input the public parameters pp, master secret key msk, attribute vector $\mathbf{x}_3$, encryption for slot 3 is defined as follows:

1. Compute $[(\mathbf{C}_{2\ell+1,\mathbf{x}_{3,1}} \| \ldots \| \mathbf{C}_{3\ell,\mathbf{x}_{3,\ell}} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2})^{\mathsf{T}}]_{\tau_t}^{-1}$ from $\mathbf{C}_{\tau_0'}^{-1}$ and sample short vector $\mathbf{t}_{\mathbf{x}_3}$ such that
   $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,\mathbf{x}_{3,i}} = \mathbf{0}$ for all $i \in [\ell]$, $\mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1} = \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2} = \mathbf{0}$, as
   $\mathbf{t}_{\mathbf{x}_3}^{\mathsf{T}} \leftarrow [(\mathbf{C}_{2\ell+1,\mathbf{x}_{3,1}} \| \ldots \| \mathbf{C}_{3\ell,\mathbf{x}_{3,\ell}} \| \mathbf{C}_{3\ell+1} \| \mathbf{C}_{3\ell+2})^{\mathsf{T}}]_{\tau_t}^{-1}(\mathbf{0}^{\mathsf{T}})$.

2. Output $\mathsf{ct}_3 = \mathbf{t}_{\mathbf{x}_3}$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$: On input the public parameters pp, the secret key sk_F for circuit F and ciphertexts ct_1, ct_2 and ct_3 corresponding to the three attributes $\mathbf{x}_1$,

$\mathbf{x}_2$ and $\mathbf{x}_3$, the decryption algorithm proceeds as follows:

1. Takes the coordinate-wise pairing between ciphertexts for slot 1 and slot 2:
   Computes $[v_0]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2}]_T$ and $[\mathbf{V}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \widehat{\mathbf{C}}_1 \odot \widehat{\mathbf{C}}_2]_T$ as $e(\mathsf{ct}_1, \mathsf{ct}_2)$.

2. Expands obtained matrix:
   Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Expands $[\mathbf{V}]_T$ to obtain:

   $[\mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi_{i,x_i}]_T$ for $i \in [\ell]$, $[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi'_{i,b}]_T$, where
   $\psi'_{i,b} = \begin{cases} \psi_{i,x_i} & \text{if } b = x_i \\ \mathbf{0} & \text{if } b = 1 - x_i \end{cases}$, for $i \in [\ell + 1, 2\ell], b \in \{0, 1\}$.

   $[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\psi_{i,b} + \mathbf{C}_{i,b} \widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b})]_T$ for $i \in [2\ell + 1, 3\ell], b \in \{0, 1\}$,
   $[\mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+1} \widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1})]_T$,
   $[\mathbf{v}^\mathsf{T}_{3\ell+2}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+2} \widehat{\mathbf{s}}^\mathsf{T}_{3\ell+2} + \widehat{\mathbf{e}}^\mathsf{T}_0 + \psi^\mathsf{T}_{3\ell+2} + \mathbf{m}^\mathsf{T})]_T$.

3. Recovers $\mathsf{BGG+18}$ ciphertext components for third slot:

   Let us denote $\mathbf{V}_{i,x_i}$ as $\mathbf{V}_i$ for $i \in [2\ell + 1, 3\ell]$.
   Computes $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i})]_T$ for $i \in [2\ell + 1, 3\ell]$,
   $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi_{3\ell+1} + \widehat{\mathbf{E}})]_T$ and $[\mathbf{t}_{\mathbf{x}_3} \mathbf{v}^\mathsf{T}_{3\ell+2}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \mathbf{t}_{\mathbf{x}_3} (\psi^\mathsf{T}_{3\ell+2} + \mathbf{m}^\mathsf{T} + \widehat{\mathbf{e}}^\mathsf{T}_0)]_T$.
   (because $\mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{i,x_i} = \mathbf{0}$ for $i \in [2\ell + 1, 3\ell]$, $\mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{3\ell+1} = \mathbf{t}_{\mathbf{x}_3} \mathbf{C}_{3\ell+2} = \mathbf{0}$)

4. Computes function to be applied on $\mathsf{BGG+18}$ ciphertexts:

   Computes $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

5. Performs $\mathsf{BGG+18}$ decryption in the exponent:

   a) Let us denote $\mathbf{V}_{i,x_i}$ as $\mathbf{V}_i$ for $i \in [\ell + 1, 2\ell]$.

   b) Computes $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_i]_T$ for $i \in [2\ell]$.

   c) Forms $[\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_\mathbf{x}]_T = [\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_1 \| \ldots \| \mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell}]_T$, $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}^m_q, \mathbf{r}_2 \in \mathbb{Z}^m_q)$.

   d) Then computes
   $[v']_T := \left[\left(\mathbf{t}_{\mathbf{x}_3} \mathbf{v}^\mathsf{T}_{3\ell+2} - \left(\mathbf{t}_{\mathbf{x}_3} \mathbf{V}_{3\ell+1} \mathbf{r}^\mathsf{T}_1 + \mathbf{t}_{\mathbf{x}_3} \mathbf{V}_\mathbf{x} \widehat{\mathbf{H}}_{F,\mathbf{x}} \mathbf{r}^\mathsf{T}_2 \right) \right) \right]_T$

6. Recover exponent via brute force if $F(\mathbf{x}) = 0$:
   After simplification, for $F(\mathbf{x}) = 0$, we get $v' = t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{t}_{\mathbf{x}_3} \mathbf{m}^\mathsf{T} + e')$, where e' is

the combined error. Find $\eta \in [-B, B] \cup [-B - \lceil q/2 \rceil, B - \lceil q/K \rceil] \cup [-B + \lceil q/K \rceil, B + \lceil q/2 \rceil]$ such that $[v_0]_T^\eta = [v']_T$ by brute-force search. If there is no such η, output $\perp$. In the correctness, we show that η can be found in polynomial steps. To speed up the operation, one can employ the baby-step giant-step algorithm.

7. Output 0 if $\eta \in [-B, B]$ and 1, otherwise.

Parameters We choose the parameters for the 3-ABE scheme as follows:

$$
\begin{aligned}
&m = n^{1.1} \log q, && k = \theta(n\ell \log q), && q = 2^{\Theta(\lambda)} \\
&\tau_0 = n \log q \log m, && \tau = m^{3.1} \ell \cdot 2^{O(d)} && \tau_0' = \omega(\sqrt{2n(\ell+1) \log q \log k}), \\
&\chi = \mathsf{SampZ}(3\sqrt{n}), && \hat{\chi} = \mathsf{SampZ}(6\sqrt{n} m^2), && B = \ell m^5 n^3 k \tau \tau_t \cdot 2^{O(d)}.
\end{aligned}
$$

We can set τ_t to be arbitrary polynomial such that $\tau_t > \tau_0'$. The parameter n may be chosen as $n = \lambda^c$ for some constant $c > 1$.

Correctness To see correctness, we first make following observations:

1. Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, then

$$
\begin{aligned}
\widehat{\mathbf{C}}_1 \odot \widehat{\mathbf{C}}_2 = (\{\psi_{i,x_i}\}_{i \in [\ell]}, \{\psi'_{i,b}\}_{i \in [\ell+1, 2\ell], b \in \{0,1\}}, \{\mathbf{C}_{i,b} \widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}\}_{\substack{i \in [2\ell+1, 3\ell], \\ b \in \{0,1\}}}, \\
\mathbf{C}_{3\ell+1} \widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1}, \mathbf{C}_{3\ell+2} \widehat{\mathbf{s}}_{3\ell+2}^\mathsf{T} + \widehat{\mathbf{e}}_0^\mathsf{T} + \psi_{3\ell+2}^\mathsf{T} + \mathbf{m}^\mathsf{T}),
\end{aligned}
$$

   where

$$
\psi'_{i,b} = \begin{cases} \psi_{i,x_i} & \text{if } b = x_i \\ \mathbf{0} & \text{if } b = 1 - x_i \end{cases}, \text{ for } i \in [\ell+1, 2\ell], b \in \{0,1\}.
$$

   Hence, on expanding $\mathbf{V}$, the decryptor obtains

$$
\begin{aligned}
&[\mathbf{V}_i]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} \psi_{i,x_i}]_T \text{ for } i \in [2\ell], \\
&[\mathbf{V}_{i,b}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{i,b} \widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b})]_T, \text{ for } i \in [2\ell+1, 3\ell], b \in \{0,1\}, \\
&[\mathbf{V}_{3\ell+1}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+1} \widehat{\mathbf{S}}_{3\ell+1} + \widehat{\mathbf{E}} + \psi_{3\ell+1})]_T, \\
&[\mathbf{v}_{3\ell+2}^\mathsf{T}]_T = [t_{\mathbf{x}_1} t_{\mathbf{x}_2} (\mathbf{C}_{3\ell+2} \widehat{\mathbf{s}}_{3\ell+2}^\mathsf{T} + \widehat{\mathbf{e}}_0^\mathsf{T} + \psi_{3\ell+2}^\mathsf{T} + \mathbf{m}^\mathsf{T})]
\end{aligned}
$$

   Here, recall that we represent $\mathbf{V}_{i,x_i}$ by $\mathbf{V}_i$, for $i \in [\ell+1, 2\ell]$.

2. Recovering $\{\psi_{2\ell+i,x_{3,i}}\}_{i\in[\ell]}$, $\psi_{3\ell+1}$ and $\psi_{3\ell+2}$:

   For $i \in [\ell]$,

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{2\ell+i,x_{3,i}} &= t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\psi_{2\ell+i,x_{3,i}} + \widehat{\mathbf{E}}_{2\ell+i,x_{3,i}}) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}}\widehat{\mathbf{S}}_{2\ell+i,x_{3,i}}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\psi_{2\ell+i,x_{3,i}} + \widehat{\mathbf{E}}_{2\ell+i,x_{3,i}}) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{2\ell+i,x_{3,i}} = 0).
\end{aligned}$$

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{3\ell+1} &= t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\psi_{3\ell+1} + \widehat{\mathbf{E}}) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1}\widehat{\mathbf{S}}_{3\ell+1}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\psi_{3\ell+1} + \widehat{\mathbf{E}}) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+1} = 0) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{SA} + \mathbf{E} + \widehat{\mathbf{E}}).
\end{aligned}$$

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{v}_{3\ell+2}^\top &= t_{\mathbf{x}_1}t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\psi_{3\ell+2}^\top + \mathbf{m}^\top + \widehat{\mathbf{e}}_0^\top) + \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2}\widehat{\mathbf{s}}_{3\ell+2}^\top) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\psi_{3\ell+2}^\top + \mathbf{m}^\top + \widehat{\mathbf{e}}_0^\top) \text{ (because } \mathbf{t}_{\mathbf{x}_3}\mathbf{C}_{3\ell+2} = 0) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{Su}^\top + \mathbf{m}^\top + (\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top)).
\end{aligned}$$

   Representing $\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{i,x_i}$ by $\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_i$ for $i \in [2\ell+1, 3\ell]$ gives us, for $i \in [2\ell+1, 3\ell]$,

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_i &= \mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{i,x_i} \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\psi_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}(\mathbf{B}_i - x_i\mathbf{G}) + \mathbf{E}_{i,x_i} + \widehat{\mathbf{E}}_{i,x_i}).
\end{aligned}$$

3. Next, observe that:

$$\begin{aligned}
\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{\mathbf{x}} &= \mathbf{t}_{\mathbf{x}_3}(\mathbf{V}_1, \ldots, \mathbf{V}_{2\ell}, \mathbf{V}_{2\ell+1}, \ldots, \mathbf{V}_{3\ell}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}(\mathbf{B}_1 - x_1\mathbf{G}) + \mathbf{E}_{1,x_1}, \ldots, \mathbf{S}(\mathbf{B}_{2\ell} - x_{2\ell}\mathbf{G}) + \mathbf{E}_{2\ell,x_{2\ell}}, \\
&\quad \mathbf{S}(\mathbf{B}_{2\ell+1} - x_{2\ell+1}\mathbf{G}) + \mathbf{E}_{2\ell+1,x_{2\ell+1}} + \widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}}, \ldots, \mathbf{S}(\mathbf{B}_{3\ell} - x_{3\ell}\mathbf{G}) + \mathbf{E}_{3\ell,x_{3\ell}} + \widehat{\mathbf{E}}_{3\ell,x_{3\ell}}) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}\mathbf{S}((\mathbf{B}_1, \ldots, \mathbf{B}_{3\ell}) - (x_1\mathbf{G}, \ldots, x_{3\ell}\mathbf{G})) \\
&\quad + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}((\mathbf{E}_{1,x_1}, \ldots, \mathbf{E}_{3\ell,x_{3\ell}}) + (\mathbf{0}_{k\times 2\ell m}, \widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}}, \ldots, \widehat{\mathbf{E}}_{3\ell,x_{3\ell}})) \\
&= t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}\mathbf{S}(\mathbf{B} - \mathbf{x}\otimes\mathbf{G}) + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3}), \\
&\quad (\text{ where } \mathbf{E}_{\mathbf{x}} = (\mathbf{E}_{1,x_1}, \ldots, \mathbf{E}_{3\ell,x_{3\ell}}) \text{ and } \widehat{\mathbf{E}}_{\mathbf{x}_3} = (\mathbf{0}_{k\times 2\ell m}, \widehat{\mathbf{E}}_{2\ell+1,x_{2\ell+1}}, \ldots, \widehat{\mathbf{E}}_{3\ell,x_{3\ell}})).
\end{aligned}$$

4. Performing BGG + 18 evaluation and decryption in the exponent yields:

$$\begin{aligned}
[v']_T &= [(\mathbf{t}_{\mathbf{x}_3}\mathbf{v}_{3\ell+2}^\top - (\mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{3\ell+1}\mathbf{r}_1^\top + \mathbf{t}_{\mathbf{x}_3}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top))]_T \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{Su}^\top + \mathbf{m}^\top + \mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top) - t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{SA} + \mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top \\
&\quad - t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}(\mathbf{B} - \mathbf{x}\otimes\mathbf{G}) + \mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top]_T \\
&= [t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{Su}^\top + \mathbf{m}^\top - \mathbf{S}(\mathbf{Ar}_1^\top + (\mathbf{BH}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^\top) \\
&\quad + t_{\mathbf{x}_1}t_{\mathbf{x}_2}\mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\top + \widehat{\mathbf{e}}_0^\top - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\top - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\top)]_T \\
&\quad (\because (\mathbf{B} - \mathbf{x}\otimes\mathbf{G})\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathbf{BH}_F - F(\mathbf{x})\mathbf{G} \text{ (Lemma 3.2))}
\end{aligned}$$

Replacing $\mathbf{BH}_F$ by $\mathbf{B}_F$, $(\mathbf{r}_1, \mathbf{r}_2)$ by $\mathbf{r}$, $\mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\mathsf{T} + \widehat{\mathbf{e}}_0^\mathsf{T} - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\mathsf{T} - (\mathbf{E}_\mathbf{x} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\mathsf{T})$ by e' and for $F(\mathbf{x}) = 0$, we get:

$$\begin{aligned}
[v']_T &= [t_{\mathbf{x}_1} t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\mathsf{T} + \mathbf{m}^\mathsf{T} - \mathbf{S}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\mathsf{T}) + e')]_T \\
&= [t_{\mathbf{x}_1} t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}(\mathbf{S}\mathbf{u}^\mathsf{T} + \mathbf{m}^\mathsf{T} - \mathbf{S}\mathbf{u}^\mathsf{T}) + e')]_T \quad (\because (\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^\mathsf{T} = \mathbf{u}^\mathsf{T}) \\
&= [t_{\mathbf{x}_1} t_{\mathbf{x}_2}(\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} + e')]_T = [v_0]_T^{(\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} + e')}
\end{aligned}$$

Thus, by brute force search we get $\eta = \mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} + e'$.

5. Bounding error e':
Recall that we set $\chi = \mathsf{SampZ}(3\sqrt{n})$, $\hat{\chi} = \mathsf{SampZ}(6\sqrt{n}m^2)$. By the definition of SampZ, we have $\|\mathbf{e}_0\|_\infty, \|\widehat{\mathbf{e}}_0\|_\infty \leq 3n$, $\|\mathbf{E}\|_\infty, \|\widehat{\mathbf{E}}\|_\infty \leq 3n$. and $\|\mathbf{E}_{i,b}\|_\infty, \|\widehat{\mathbf{E}}_{i,b}\|_\infty \leq 6nm^2$ for $i \in [3\ell]$ and $b \in \{0,1\}$, $\|\mathbf{r}\|_\infty \leq \sqrt{n}\tau$, $\|\mathbf{t}_{\mathbf{x}_3}\|_\infty \leq \sqrt{n}\tau_t$, and $\|\widehat{\mathbf{H}}_{F,\mathbf{x}}\|_\infty \leq m \cdot 2^{O(d)}$, where the last inequality follows from Lemma 3.2. Thus, we have

$$\begin{aligned}
e' &= \mathbf{t}_{\mathbf{x}_3}(\mathbf{e}_0^\mathsf{T} + \widehat{\mathbf{e}}_0^\mathsf{T} - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^\mathsf{T} - (\mathbf{E}_\mathbf{x} + \widehat{\mathbf{E}}_{\mathbf{x}_3})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^\mathsf{T}) \\
&\leq k\sqrt{n}\tau_t(6n + 6n^{1.5}m\tau + 24\ell m^5 n^{1.5}\tau \cdot 2^{O(d)}) \\
&= O(\ell m^5 n^2 k\tau\tau_t \cdot 2^{O(d)}) \leq B
\end{aligned}$$

by our choice of B.

6. Bounding $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T}$:
When message bit $b = 0$, then $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T}$=0. For $b = 1$, $\lfloor q/K \rceil \leq |\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T}| \leq \tau_t\sqrt{n}k \cdot \lfloor q/K \rceil$ (where $K = 2\tau_t\sqrt{n} \cdot k$), unless $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} = 0$. Thus, for $b = 0$, $\eta \in [-B, B]$ and for ($b = 1$, and $B < \lfloor \frac{q}{2K} \rceil$), $\eta \in [-B - \lceil q/2 \rceil, -\lceil q/K \rceil + B] \cup [\lceil q/K \rceil - B, B + \lceil q/2 \rceil]$, unless $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} = 0$. Observe that $|\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T}| \in \{0, \lceil \frac{q}{K} \rceil, 2 \cdot \lceil \frac{q}{K} \rceil, \ldots, \tau_t\sqrt{n}k \cdot \lceil \frac{q}{K} \rceil\}$. Thus, η can take only $2B + 4B\tau_t\sqrt{n}k$ different values. Since, B, k, τ_t are polynomially bounded, η can be found by brute force search in polynomial steps.

The probability that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} = 0$ is non-negligible but bounded away from 1 and hence this may be amplified as discussed below.

Amplifying Correctness. Above, we set $\mathbf{m} = \mu \cdot (q/K)(1, \ldots, 1)$. Note that for correctness, we require that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} \neq 0$. However, since we are constrained to sample $\mathbf{t}_{\mathbf{x}_3}$ polynomially bounded (since the message must be recovered from the exponent), the probability that $\mathbf{t}_{\mathbf{x}_3}\mathbf{m}^\mathsf{T} = 0$ is non-negligible, leading to error in correctness. A simple method to amplify correctness is to simply run the scheme in parallel λ times, and output 0 only if all instances output 0. This (standard) trick allows to make the correctness error

exponentially small, although with the disadvantage of reducing efficiency. We note that we can do better by replacing the vector $\mathbf{u}$ with λ vectors $\mathbf{u}_1, \ldots, \mathbf{u}_\lambda$ and providing λ secret keys $\mathbf{r}_1, \ldots, \mathbf{r}_\lambda$ corresponding to each one. Similarly, we can provide λ encodings of the message bit, decrypt each one of them and output 0 only if all instances output 0. However, we choose not to clutter the (already complex) formal description with this added complication for ease of exposition.

3.10.2 Discussion of Security

Compared to [BV22], we require slightly different security requirements for the encodings (even though neither works formalize this). First, we need the encoding to retain security even if some of the masks are stripped off, as long as only one encoding for the same position is revealed. We expect this to be secure since these stripped off encodings are fresh BGG + 18 encodings and should be secure by BGG + 18 security. Second, in their case, only a single BGG + 18 secret key is generated per each instance of BGG + 18, which is sampled afresh for each ciphertext, while in our case, we use the same BGG + 18 instance throughout the system and generate multiple secret keys for it. On the other hand, in our case, the encodings all live in the exponent, unlike their case, where they live "downstairs". Hence, the attacker gets restricted to only linear attacks by GGM whereas the attacker has more freedom in their construction.

3.11 TWO-INPUT ABE FOR POLYNOMIAL CIRCUITS USING BV22

In this section, we construct candidate two input ABE scheme using the structure of [BV22] scheme. Unlike other schemes in this chapter, the construction below does not employ pairings and thus is expected to be post-quantum secure. Besides, it can support polynomial-size circuits with any depth. Formally, it supports attribute space $A_\lambda = \{0,1\}^{\ell(\lambda)}$ and any circuit class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ that is subclass of $\{C_{2\ell(\lambda),d(\lambda)}\}_\lambda$ with arbitrary $\ell(\lambda) \leq \text{poly}(\lambda)$ and $d(\lambda) \leq \text{poly}(\lambda)$, where $C_{2\ell(\lambda),d(\lambda)}$ is a set of circuits with input length $2\ell(\lambda)$ and depth at most $d(\lambda)$.

3.11.1 Construction

The construction is defined as follows:

$\mathsf{Setup}(1^\lambda)$: On input 1^λ, the setup algorithm defines the parameters $n = n(\lambda)$, $m = m(\lambda)$, $k = k(\lambda)$, noise distribution χ, $\hat{\chi}$ over $\mathbb{Z}$, $\tau_0 = \tau_0(\lambda)$, $\tau = \tau(\lambda)$, $\tau'_0 = \tau'_0(\lambda)$, $\tau_t = \tau_t(\lambda)$ and $B = B(\lambda)$ as specified in Sec. 3.10.1. Let ℓ be the length of the attributes and d be the maximum depth of circuits. Then the setup algorithm does the following.

1. Samples $\mathsf{BGG+18}$ master secret and public keys:
   a) Samples $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ such that $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.
   b) Samples random matrices $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{2\ell}) \leftarrow (\mathbb{Z}_q^{n\times m})^{2\ell}$ and a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.
2. Samples $(\mathbf{C}, \mathbf{C}_{\tau'_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^{2n(\ell+1)}, 1^k, q)$, where
$$\mathbf{C}^\mathsf{T} = (\mathbf{C}_{\ell+1,0} \| \mathbf{C}_{\ell+1,1} \| \ldots \| \mathbf{C}_{2\ell,0} \| \mathbf{C}_{2\ell,1} \| \mathbf{C}_{2\ell+1} \| \mathbf{C}_{2\ell+2}) \in (\mathbb{Z}_q^{k\times n})^{2\ell+2}.$$
3. Outputs
$$\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u}), \ \mathsf{msk} = (\mathbf{A}_{\tau_0}^{-1}, \mathbf{C}_{\tau'_0}^{-1}).$$

$\mathsf{Enc}_1(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_1, \mu)$: On input the public parameters pp, master secret msk, attribute vector $\mathbf{x}_1$ and message bit μ, encryption for slot 1 does the following:

1. Sets $\mathbf{m} = \left\lfloor \frac{q}{K} \right\rceil \mu(1, \ldots, 1) \in \mathbb{Z}_q^k$. We define $K = 2\tau_t\sqrt{n}k$.
2. Samples a random secret matrix $\mathbf{S} \leftarrow \mathbb{Z}_q^{k\times n}$ and error vectors/matrices as $\mathbf{e}_0 \leftarrow \chi^k$, $\mathbf{E} \leftarrow \chi^{k\times m}$, $\mathbf{E}_{i,x_{1,i}} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell]$, and $\mathbf{E}_{i,b} \leftarrow \hat{\chi}^{k\times m}$ for $i \in [\ell+1, 2\ell]$ and $b \in \{0,1\}$.
3. Computes
$$\psi_{i,x_{1,i}} = \mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{E}_{i,x_{1,i}} \text{ for } i \in [\ell],$$
$$\psi_{i,b} = \mathbf{S}(\mathbf{B}_i - b\mathbf{G}) + \mathbf{E}_{i,b} \text{ for } i \in [\ell+1, 2\ell], b \in \{0,1\},$$
$$\psi_{2\ell+1} = \mathbf{SA} + \mathbf{E}, \ \psi_{2\ell+2}^\mathsf{T} = \mathbf{Su}^\mathsf{T} + \mathbf{e}_0^\mathsf{T}.$$

4. For $i \in [\ell+1, 2\ell], b \in \{0,1\}$, samples

$$\widehat{\mathbf{S}}_{i,b} \leftarrow \mathbb{Z}_q^{n\times m},\ \widehat{\mathbf{S}}_{2\ell+1} \leftarrow \mathbb{Z}_q^{n\times m},\ \widehat{\mathbf{s}}_{2\ell+2} \leftarrow \mathbb{Z}_q^{n},$$

$$\widehat{\mathbf{E}}_{i,b} \leftarrow \hat{\chi}^{k\times m},\ \widehat{\mathbf{E}} \leftarrow \chi^{k\times m},\ \widehat{\mathbf{e}}_0 \leftarrow \chi^{k}.$$

5. Computes $\widehat{\psi}_{i,b} = \mathbf{C}_{i,b}\widehat{\mathbf{S}}_{i,b} + \widehat{\mathbf{E}}_{i,b} + \psi_{i,b}$ for $i \in [\ell+1, 2\ell], b \in \{0,1\}$,

$$\widehat{\psi}_{2\ell+1} = \mathbf{C}_{2\ell+1}\widehat{\mathbf{S}}_{2\ell+1} + \widehat{\mathbf{E}} + \psi_{2\ell+1},\ \ \widehat{\psi}_{2\ell+2}^{\mathsf{T}} = \mathbf{C}_{2\ell+2}\widehat{\mathbf{s}}_{2\ell+2}^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}} + \psi_{2\ell+2}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}}.$$

6. Outputs $\mathsf{ct}_1 = (\{\psi_{i,x_{1,i}}\}_{i\in[\ell]}, \{\widehat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \widehat{\psi}_{2\ell+1}, \widehat{\psi}_{2\ell+2})$.

$\mathsf{Enc}_2(\mathsf{pp}, \mathsf{msk}, \mathbf{x}_2)$: On input the public parameters pp, master secret key msk and an attribute vector $\mathbf{x}_2$, encryption for slot 2 is defined as follows:

1. Computes $[(\mathbf{C}_{2\ell+1}\|\mathbf{C}_{2\ell+2}\|\mathbf{C}_{\ell+1,x_{2,1}}\|\ldots\|\mathbf{C}_{2\ell,x_{2,\ell}})^{\mathsf{T}}]_{\tau_t}^{-1}$ from $\mathbf{C}_{\tau_0'}^{-1}$ and samples a short vector $\mathbf{t}_{\mathbf{x}_2}$ such that

$$\mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+1}\|\mathbf{C}_{2\ell+2}\|\mathbf{C}_{\ell+1,x_{2,1}}\|\ldots\|\mathbf{C}_{2\ell,x_{2,\ell}}) = \mathbf{0} \mod q \text{ as}$$

$$\mathbf{t}_{\mathbf{x}_2}^{\mathsf{T}} \leftarrow [(\mathbf{C}_{2\ell+1}\|\mathbf{C}_{2\ell+2}\|\mathbf{C}_{\ell+1,x_{2,1}}\|\ldots\|\mathbf{C}_{2\ell,x_{2,\ell}})^{\mathsf{T}}]_{\tau_t}^{-1}(\mathbf{0}).$$

2. Returns $\mathsf{ct}_2 = \mathbf{t}_{\mathbf{x}_2}$.

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$: On input the public parameters pp, master secret key msk and a function F, the keygen algorithm does the following.

1. Generates $\mathsf{BGG+18}$ function key:

   a) Computes $\mathbf{H}_F = \mathsf{EvalF}(\mathbf{B}, F)$ and $\mathbf{B}_F = \mathbf{B}\mathbf{H}_F$.

   b) Computes $[\mathbf{A}\|\mathbf{B}_F]_{\tau}^{-1}$ from $\mathbf{A}_{\tau_0}^{-1}$ and samples $\mathbf{r} \in \mathbb{Z}^{2m}$ as $\mathbf{r}^{\mathsf{T}} \leftarrow [\mathbf{A}\|\mathbf{B}_F]_{\tau}^{-1}(\mathbf{u}^{\mathsf{T}})$.

2. Returns $\mathsf{sk}_F = \mathbf{r}$.

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_1, \mathsf{ct}_2)$: On input the public parameters pp, key $\mathsf{sk}_F = \mathbf{r}$, and slot 1 and slot 2 ciphertexts ct_1, ct_2, the decryption algorithm does the following.

1. Parses the public parameters pp as

$$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{u})$$

and the ciphertexts for slot 1 and slot 2 as

$$\mathsf{ct}_1 = (\{\psi_{i,x_{1,i}}\}_{i\in[\ell]}, \{\widehat{\psi}_{i,b}\}_{i\in[\ell+1,2\ell],b\in\{0,1\}}, \widehat{\psi}_{2\ell+1}, \widehat{\psi}_{2\ell+2}),\ \mathsf{ct}_2 = \mathbf{t}_{\mathbf{x}_2}.$$

2. Computes

$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V} = \mathbf{t}_{\mathbf{x}_2}(\psi_{1,x_{1,1}}\| \ldots \|\psi_{\ell,x_{1,\ell}}\|\widehat{\psi}_{\ell+1,x_{2,1}}\| \ldots \|\widehat{\psi}_{2\ell,x_{2,\ell}}\|\widehat{\psi}_{2\ell+1}\|\widehat{\psi}^{\mathsf{T}}_{2\ell+2}).$$

3. Expands $\mathbf{t}_{\mathbf{x}_2}\mathbf{V}$ to obtain

$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{i,x_{1,i}} = \mathbf{t}_{\mathbf{x}_2}\psi_{i,x_{1,i}} \text{ for } i \in [\ell], \quad \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\ell+i,x_{2,i}} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{\ell+i,x_{2,i}} \text{ for } i \in [\ell],$$
$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+1}, \qquad \mathbf{t}_{\mathbf{x}_2}\mathbf{v}^{\mathsf{T}}_{2\ell+2} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}^{\mathsf{T}}_{2\ell+2}.$$

4. Forms $\mathbf{r} = (\mathbf{r}_1 \in \mathbb{Z}_q^m, \mathbf{r}_2 \in \mathbb{Z}_q^m)$ and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.

Let,

$$(\mathbf{V}_{1,x_1}\| \ldots \|\mathbf{V}_{2\ell,x_{2\ell}}) = \mathbf{V}_{\mathbf{x}}.$$

5. Computes $\widehat{\mathbf{H}}_{F,\mathbf{x}} = \mathsf{EvalFX}(F, \mathbf{x}, \mathbf{B})$.

6. Computes

$$v = (\mathbf{t}_{\mathbf{x}_2}\mathbf{v}^{\mathsf{T}}_{2\ell+2} - \left(\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1}\mathbf{r}_1^{\mathsf{T}} + \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^{\mathsf{T}}\right)).$$

7. Outputs 0 if $v \in [-B, B]$ and 1, otherwise.

Correctness: To see correctness, we first make following observations:

1. Let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$.

On expanding $\mathbf{t}_{\mathbf{x}_2}\mathbf{V}$, the decryptor obtains, for $i \in [\ell]$,

$$\begin{aligned}\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{i,x_{1,i}} &= \mathbf{t}_{\mathbf{x}_2}\psi_{i,x_{1,i}} \\ &= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B}_i - x_{1,i}\mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}\mathbf{E}_{i,x_{1,i}}.\end{aligned}$$

$$\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\ell+i,x_{2,i}} = \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{\ell+i,x_{2,i}}$$

$$
\begin{aligned}
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{\ell+i,\mathbf{x}_{2,i}}\widehat{\mathbf{S}}_{\ell+i,x_{2,i}} + \widehat{\mathbf{E}}_{\ell+i,x_{2,i}} + \psi_{\ell+i,\mathbf{x}_{2,i}}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B}_{\ell+i} - x_{2,i}\mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E}_{\ell+i,x_{2,i}} + \widehat{\mathbf{E}}_{\ell+i,x_{2,i}}).
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1} &= \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+1} \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+1}\widehat{\mathbf{S}}_{2\ell+1} + \widehat{\mathbf{E}} + \psi_{2\ell+1}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}\mathbf{A} + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E} + \widehat{\mathbf{E}}).
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^{\mathsf{T}} &= \mathbf{t}_{\mathbf{x}_2}\widehat{\psi}_{2\ell+2}^{\mathsf{T}} \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{C}_{2\ell+2}\widehat{\mathbf{s}}_{2\ell+2}^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}} + \psi_{2\ell+2}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}}) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}}) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}}).
\end{aligned}
$$

2. Next, observe that:

$$
\begin{aligned}
\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}} &= \mathbf{t}_{\mathbf{x}_2}(\mathbf{V}_1, \ldots, \mathbf{V}_\ell, \mathbf{V}_{\ell+1}, \ldots, \mathbf{V}_{2\ell}) \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{S}(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{t}_{\mathbf{x}_2}(\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2}), \\
&\quad \text{where } \mathbf{E}_{\mathbf{x}} = (\mathbf{E}_{1,x_1}, \ldots, \mathbf{E}_{2\ell,x_{2\ell}}) \text{ and } \widehat{\mathbf{E}}_{\mathbf{x}_2} = (\mathbf{0}_{k\times\ell m}, \widehat{\mathbf{E}}_{\ell+1,x_{\ell+1}}, \ldots, \widehat{\mathbf{E}}_{2\ell,x_{2\ell}}).
\end{aligned}
$$

3. Finally, we get,

$$
\begin{aligned}
v &= \mathbf{t}_{\mathbf{x}_2}\mathbf{v}_{2\ell+2}^{\mathsf{T}} - (\mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{2\ell+1}\mathbf{r}_1^{\mathsf{T}} + \mathbf{t}_{\mathbf{x}_2}\mathbf{V}_{\mathbf{x}}\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^{\mathsf{T}}) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}} - \mathbf{S}(\mathbf{A}\mathbf{r}_1^{\mathsf{T}} + (\mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G})\mathbf{r}_2^{\mathsf{T}})) \\
&+ \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}} - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^{\mathsf{T}} - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^{\mathsf{T}}) \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}} - \mathbf{S}(\mathbf{A}\|\mathbf{B}_F)\mathbf{r}^{\mathsf{T}}) + e' \\
&= \mathbf{t}_{\mathbf{x}_2}(\mathbf{S}\mathbf{u}^{\mathsf{T}} + \mathbf{m}^{\mathsf{T}} - \mathbf{S}\mathbf{u}^{\mathsf{T}}) + e' \\
&= \mathbf{t}_{\mathbf{x}_2}\mathbf{m}^{\mathsf{T}} + e' \\
&\text{where } e' = \mathbf{t}_{\mathbf{x}_2}(\mathbf{e}_0^{\mathsf{T}} + \widehat{\mathbf{e}}_0^{\mathsf{T}} - (\mathbf{E} + \widehat{\mathbf{E}})\mathbf{r}_1^{\mathsf{T}} - (\mathbf{E}_{\mathbf{x}} + \widehat{\mathbf{E}}_{\mathbf{x}_2})\widehat{\mathbf{H}}_{F,\mathbf{x}}\mathbf{r}_2^{\mathsf{T}}).
\end{aligned}
$$

4. As discussed in section 3.10.1, the error e' is bounded by $O(\ell m^5 n^2 k\tau\tau_t 2^{O(d)})$, and $\mathbf{t}_{\mathbf{x}}\mathbf{m}^{\mathsf{T}} = 0$ when $b = 0$ and $\mathbf{t}_{\mathbf{x}}\mathbf{m}^{\mathsf{T}} \leq \tau_t\sqrt{n}k \cdot \lceil \frac{q}{2\tau_t\sqrt{nk}} \rceil$ for $b = 1$. Thus, for $b = 0$, $v \in [-B, B]$ and for ($b = 1$, and $B < \frac{q}{4k\sqrt{n}\tau_t}$), $v \notin [-B, B]$ unless $\mathbf{t}_{\mathbf{x}}\mathbf{m}^{\mathsf{T}} = 0$. The probability that $\mathbf{t}_{\mathbf{x}}\mathbf{m}^{\mathsf{T}} = 0$ is non-negligible but bounded away from 1 and hence this may be amplified as discussed in section 3.10.1.

CHAPTER 4

CONSTANT INPUT ATTRIBUTE BASED ENCRYPTION FROM EVASIVE AND TENSOR LWE

4.1 INTRODUCTION

In this chapter, we extend the results of Chapter 3 from arity 2 (and heuristic construction for arity 3) to *any* constant arity. Our constructions in this chapter are based on recently introduced lattice based assumptions of evasive LWE [Wee22; Tsa22] and tensor LWE [Wee22] and have the potential of being quantum secure.

Related Work.

As discussed in Chapter 3, there are very few studies on MIABE and MIPE. Here, we compare the results in this chapter to those in Chapter 3 and the related work by Francati *et al.* [FFMV23].

Comparison with the results in Chapter 3 and in [FFMV23]: The constructions in Chapter 3 support only arity 2 for NC_1 and uses LWE and pairings and are not secure against quantum attacks. The security is proven in generic group model (GGM) or is based on non-falsifiable KOALA assumption. The constructions in this chapter are based only on lattice based assumptions and the security is proven in the standard model and hence has the possibility of being secure against quantum attacks. The assumptions are new and need more study to understand them better and build confidence in their plausibility. Francati et al. [FFMV23] provided multi-input PE (hence also ABE) schemes for the restricted functionality of conjunctions of (bounded) polynomial depth from LWE. Notably, one of their constructions can support polynomial arity unlike our results in Chapter 3 and this chapter, which is a plus. On the other hand, their security model does not support collusions, which is typically the main technical challenge

in constructing ABE and PE even in the single input setting. As another plus, when restricted to constant (though not polynomial) arity, their constructions can support user corruption, which our constructions, in Chapter 3 and this chapter, cannot. However we support a much more expressive function class which is not restricted to conjunctions.

We briefly mention the stronger notion of multi-input *functional encryption* (miFE) [GGG+14], which, as discussed in the previous chapter, generalizes multi-input ABE and PE. In contrast to MIABE and MIPE, miFE has been studied extensively, and admits constructions for various functionalities from a variety of assumptions [GGG+14; AJ15; AGRW17; DOT18; ACF+18; CDG+18a; Tom19; ABKW19; ABG19; LT19; AGT21b; AGT21a; AGT22]. However, since multi-input FE for NC_1 implies indistinguishability obfuscation (iO) [BGI+01; GGH+13], it remains an important area of study to instantiate weaker notions such as MIABE and MIPE from assumptions not known to imply iO. This is particularly important in the post quantum regime, where constructions of iO are still based on strong, ill-understood assumptions which are often broken [Agr19; APM20; WW21; GP21; DQV+21; AJS23]. Several prior works therefore focus on instantiating iO based constructions from weaker assumptions [AY20; AWY20; Wee22; Tsa22; VWW23; AKYY23], a direction also followed by this work.

4.2 OUR RESULTS

As discussed, current known results for MIABE schemes are quite restricted. In this chapter, we significantly extend the reach of multi-input ABE schemes by providing the first construction of MIABE for the function class NC_1 for *any* constant arity from the recently introduced evasive LWE assumption [Wee22; Tsa22]. Our construction can be extended to support the function class P by using evasive and a suitable strengthening of tensor LWE. For the special case of arity 2, we need only the assumptions introduced by Wee, i.e. evasive LWE for NC_1 and evasive plus tensor LWE for P (i.e. we do not need

Paper	Arity	Functionality	Corruption	Collusion	Assumption
[FFMV23]	Poly	Conjunctions in P	No	No	LWE
[FFMV23]	Constant	Conjunctions in P	Yes	No	LWE
Chapter 3	2	NC_1	No	Yes	Koala and LWE
Chapter 3	2	P	No	Yes	Heuristic
This Chapter	2	P	No	Yes	Evasive and Tensor LWE
This Chapter	Constant	NC_1	No	Yes	Evasive LWE
This	Constant	P	No	Yes	Evasive and strong Tensor LWE

Table 4.1: Comparison with Related Work in MIPE. Note that KOALA is a non-standard knowledge type assumption and "heuristic" means that there is no proof of security.

to strengthen tensor LWE).[1]

In more detail, our construction supports k encryptors, for any constant k, where each encryptor uses the master secret key msk to encode its input $(\mathbf{x}_i, m_i)$, the key generator computes a key sk_f for a function $f \in \mathsf{NC}_1$ (or P at the cost of a stronger assumption) and the decryptor can recover $(m_1, \ldots, m_k)$ if and only if $f(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 1$. We prove security in the standard indistinguishability game defined in Chapter 3 from the aforementioned assumptions. Using the compiler from Chapter 3, the MIABE schemes can be upgraded to multi-input predicate encryption schemes for the same arity and function class. Along the way, we show that the tensor LWE assumption can be reduced to standard LWE in a special case which was not known before. This adds confidence to the plausibility of the assumption and may be of wider interest.

We defer details about our strengthening of tensor LWE for P as well as the new implication discussed above to the technical overview (Section 4.3) since stating them formally will require heavy notation which we do not want to introduce here. We provide a comparison with known results in Table 4.1.

[1] Actually, our definition of evasive LWE is slightly different from that defined in [Wee22]. Please refer to Assumption 4.6 and the related discussion.

Perspective: Connection to Witness Encryption. Witness encryption (WE) is defined for some NP language L with a corresponding witness relation R. In WE, an encryptor encrypts a message m to a particular problem instance x. The decryptor can recover m if $x \in L$ and it knows a witness w such that $R(x, w) = 1$. Security posits that a ciphertext hides the message m so long as $x \notin L$. Brakerski et al. [BJK+18] showed that MIABE for polynomial arity implies witness encryption – this may explain in part why constructions of MIABE have been so elusive. Even for smaller arity, there are nontrivial implications – for instance, the arity 2 MIABE for NC_1 in Chapter 3 implies a compression factor of $1/3$ for witness encryption, which may be considered surprising. In the other direction, it is well known that in the single input setting, witness encryption implies attribute based encryption [GGH+13]. It is completely unclear however, how to generalize this implication to the multi-input setting – in the setting of single input, the ABE ciphertext contains a WE ciphertext for an NP statement that embeds the attribute. If the attributes are distributed amongst multiple parties, the above approach fails and appears challenging to extend. Thus, MIABE implies new results in WE but not the other way around – indeed, in MIABE, all encryptors must choose their randomness independently to construct a ciphertext for their respective slot, whereas in WE, there is only one encryptor who constructs the ciphertexts for all slots, making it possible to choose correlated randomness across slots. As we will see, this creates a major technical hurdle in designing MIABE, which is not present in WE. Also note that MIABE can subsequently be strengthened to MIPE using lockable obfuscation, as discussed above.

We also note that single input ABE is the strongest application of the stated definition of WE in [GGH+13]. Since the definition of WE given in [GGH+13] only hides the message in the ciphertext when the statement is not in the language, the notion is insufficient to give any meaningful security guarantee when the statement is actually believed to be true but the witness is not known, such as solutions to some of the Clay Institute Millennium Prize Problems, as discussed in [GGH+13]. Hence, we believe that the primitives of MIABE and MIPE deserve to be studied even from assumptions that are already known

to imply WE, such as evasive LWE [Tsa22; VWW23].

4.3 TECHNICAL OVERVIEW

We first briefly recap the techniques used in Chapter 3 and discuss the difficulty in extending the techniques in Chapter 3 to the construction of any constant arity.

Recap of Chapter 3. As we discussed before, the main difficulty in building an MIABE scheme is simultaneously fulfilling two opposing requirements: (1) each encryptor should be able to generate its own ciphertexts independently, (2) these independently generated ciphertexts should permit some kind of "joining" that lets them be viewed as multiple components of a single ABE ciphertext, such that decryption can proceed as in the single input setting. To achieve joining of ciphertext components, existing single input schemes generate multiple ciphertext components using common randomness. However, evidently, two independent sources, each generating an unbounded number of ciphertexts (say Q_1 and Q_2 respectively) cannot even store, much less embed, $Q_1 \cdot Q_2$ random strings in the ciphertexts they compute (even if they share a common PRF key).

In the two-input setting, we solve this conundrum by using the beautiful synergy between the algebraic structure offered by lattice based single input ABE schemes and pairing based constructions. This synergy was first discovered and harnessed by Agrawal and Yamada [AY20] in the context of broadcast encryption (a.k.a succinct single input ciphertext policy ABE for NC_1). We notice that the same synergy can be beneficial for the two-input *key* policy ABE setting, albeit for different reasons.

In more detail, in Chapter 3, we achieve the joining of ciphertexts via common randomness by letting each party embed fresh randomness in the exponent of a pairing based group for each ciphertext it computes. Now, party 1 (respectively 2) has Q_1 (respectively Q_2) random elements embedded in its Q_1 (respectively Q_2) ciphertexts. Using the pairing

operation, the dercryptor can compute $Q_1 \cdot Q_2$ elements by pairwise multiplication in the exponent. In more detail, for each input, party 1 samples randomness t_1 and encodes it in $\mathbf{G}_1$, party 2 samples randomness t_2 and encodes it in $\mathbf{G}_2$, where $\mathbf{G} : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbb{G}_T$ is a pairing group with prime order q. Now these ciphertexts may be combined to form a new ciphertext with respect to the randomness $t_1 t_2$ on $\mathbf{G}_T$. This allows to uniquely separate every pair of ciphertexts, since each pair (i, j) where $i \in [Q_1]$ and $j \in [Q_2]$, will have unique randomness $t_1^i t_2^j$. We have by security of pairings that these $Q_1 \cdot Q_2$ correlated terms are indistinguishable from random in the exponent. This allows for generating the requisite randomness and solving the difficulty described above.

Fruitful interplay of pairings and lattices. However, generating joint randomness is not the final goal – the ciphertexts generated using the above joining procedure must behave like an ABE! Note that, having relied on a pairing, whatever we have obtained must live in the exponent of a group. Also note that, pairing based ABE schemes have been rendered unhelpful by this point, since the single multiplication afforded by the pairing has been used up and can no longer participate in the design of the ABE. Here, similarly to [AY20; AWY20], we are rescued by the serendipitously well-fitting structure of a lattice based ABE scheme constructed by Boneh et al [BGG+14]. In [BGG+14] (henceforth BGG + 18), decryption works as follows: (i) homomorphically compute the circuit f on ciphertext encodings – this step is *linear* even for $f \in \mathsf{P}$, (ii) perform a product of the ciphertext matrix and secret key vector, (iii) round the recovered value to recover the message. Hence, the first two steps can be performed "upstairs" in the exponent and the last step may be performed "downstairs" by recovering the exponent brute force.

Structure of BGG + 18. Let us recall the structure of the BGG + 18 scheme, since this forms the starting point of our construction. As observed in multiple works, in BGG + 18, the ciphertext for an attribute $\mathbf{x} \in [\ell]$ in BGG + 18 is computed by

first generating LWE encodings for all possible values of the attribute $\mathbf{x}$, namely, $\{\psi_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ and then choosing $\{\psi_{i,x_i}\}_{i\in[\ell]}$ where x_i is the i-th bit of attribute $\mathbf{x}$. Here, $\psi_{i,b} = \mathbf{s}(\mathbf{A}_i - x_{i,b}\cdot\mathbf{G}) + \mathsf{noise}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ are public matrices, $\mathbf{s} \in \mathbb{Z}_q^n$ is freshly chosen randomness, and $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ is the special "gadget" matrix which admits a public trapdoor (details not important here). Here, and in the remainder of this overview, we use noise to denote freshly and independently sampled noise terms of appropriate dimension, for each sample. Choosing components based on $\mathbf{x}$ and concatenating the samples yields $\mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G}) + \mathsf{noise}$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times \ell m}$ denotes the concatenation of $\{\mathbf{A}_i\}_{i\in[\ell]}$.

To evaluate a circuit $f \in \mathsf{P}$, BGG + 18 observe that there exists an efficiently computable low norm matrix, denoted by $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$, so that the right multiplication of $(\mathbf{A} - \mathbf{x}\otimes\mathbf{G})$ by $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$ yields a quantity of the form $\mathbf{A}_f - f(\mathbf{x})\mathbf{G}$ – since the matrix is low norm, this can be right multiplied to $\mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G}) + \mathsf{noise}$ to obtain approximately $\mathbf{s}(\mathbf{A}_f - f(\mathbf{x})\mathbf{G})$ without blowing up the noise. The decryption key for a function f is a low norm vector which, loosely speaking, is used in a matrix vector product that allows canceling the masking term $\mathbf{s}\mathbf{A}_f$ when $f(\mathbf{x}) = 0$, and this in turn allows to recover the message.

Circling back to the construction in Chapter 3, the first encryptor can (roughly speaking) compute $[t_1\cdot\psi_{\mathbf{x}}]_1$, $[t_1]_1$, the second encryptor can compute $[t_2\cdot\psi_{\mathbf{y}}]_2$, $[t_2]_2$ and the decryptor can compute $[t_1t_2\psi_{\mathbf{x}\|\mathbf{y}}]_T$, $[t_1t_2]_T$. Note that randomization by t_1t_2 is absolutely essential for security, else the adversary can potentially recover terms like $\mathbf{s}(\mathbf{A} - \mathbf{x}\otimes\mathbf{G}) + \mathsf{noise}$ and $\mathbf{s}(\mathbf{A} - \overline{\mathbf{x}}\otimes\mathbf{G}) + \mathsf{noise}$ in the exponent, which allows canceling $\mathbf{s}\mathbf{A}$ by subtraction and leads to a complete break of security. Next, the circuit f can be evaluated in the exponent as described above by right multiplication with a low norm matrix and the secret key can be applied by the matrix vector product to obtain the (scaled) message plus some noise in the exponent. The noise growth can be suitably bounded for the circuit class NC_1, and given $[t_1t_2]_T$, one can recover the message using brute force discrete log computation.

While the construction in Chapter 3 is an important first step towards constructing MIABE schemes, it is evident that going beyond degree two is difficult while relying on pairings. While we do consider arity 3, in Chapter 3, by additionally relying on ideas from a clever lattice based scheme by Brakerski and Vaikuntanathan [BV22], this scheme is heuristic, i.e. does not have a proof based on any clean assumption. Thus, it is completely unclear how to go beyond arity 2 using the techniques developed in Chapter 3, even for NC_1. A natural idea to overcome the barrier of 2 is to rely on lattices in lieu of pairings.

Towards Lattice Based Constructions. Taking a step back, a promising direction would be to consider the lattice adaptation of the Agrawal-Yamada broadcast encryption scheme [AY20] recently proposed by Wee [Wee22]. This construction makes important progress in identifying a clean assumption in the lattice regime that captures the functionality provided by the pairing without relying on bilinear groups, and can be used to construct advanced primitives like broadcast encryption and witness encryption without relying on iO (or the messy assumptions needed to build iO in the post quantum regime). In more detail, Wee [Wee22] suggested two new assumptions – the evasive LWE and tensor LWE and used these to construct ciphertext polict ABE schemes with optimal parameters. We describe his approach next.

Overview of Wee's approach. The main idea of Wee is to cleverly replace the randomization in the exponent by tensoring on the ground. In more detail, Wee observes that the transformation of $(\mathbf{A} - \mathbf{x} \otimes \mathbf{G})$ to $(\mathbf{A}_f - f(\mathbf{x})\mathbf{G})$ via right multiplication by $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$ is preserved under tensoring with random low norm vectors $\mathbf{r}$. To see this, note that

$$\mathbf{s}(\mathbf{A} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise} = \underbrace{\mathbf{s}(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})}_{\text{Randomized secret}} \mathbf{A} + \mathsf{noise}$$

where the latter quantity can be seen as BGG + 18 ciphertext with a tensored LWE secret. This easily implies that homomorphism is preserved even with tensoring as

desired. Hence, one can homomorphically evaluate f on $(\mathbf{A} - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}^\top$ to obtain $(\mathbf{A}_f - f(\mathbf{x})\mathbf{G}) \otimes \mathbf{r}^\top$ via right multiplication by $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$.

Importantly, Wee shows that a very natural adaptation of [AY20], obtained by replacing randomization in the exponent by tensoring can be shown secure under a new and elegant assumption, which he calls *evasive* LWE. To support NC_1, he shows that evasive LWE suffices, while to support P, one additionally needs another new assumption, which he calls *tensor* LWE. The formulation of a relatively simple and general assumption in the lattice regime that allows to give a proof for a very natural construction of succint ciphertext policy ABE is a very important contribution which is likely to influence many future lattice constructions, including ours. We describe these assumptions next.

Evasive LWE. The evasive LWE assumption, introduced by Wee [Wee22] (and independently Tsabary [Tsa22]), is a strengthening of the LWE assumption which says that certain extra information, namely Gaussian preimages to LWE public matrices, can only be used in a "semi-honest" way. Recall that the LWE assumption says that

$$(\mathbf{B}, \mathbf{sB} + \mathbf{e}) \approx_c (\mathbf{B}, \mathbf{c})$$

where $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$ for some low norm "noise" distribution χ and $\mathbf{c} \leftarrow \mathbb{Z}_q^m$. Intuitively, the evasive LWE assumption says that if the adversary is additionally given some low norm matrix $\mathbf{K}$ such that $\mathbf{BK} = \mathbf{P}$, which we denote as $\mathbf{B}^{-1}(\mathbf{P})$ (as in the literature, see for instance [Wee22]), for some efficiently sampleable matrix $\mathbf{P}$, then the adversary can exploit this extra information only via the limited means of computing the product $(\mathbf{sB} + \mathbf{e}) \cdot \mathbf{B}^{-1}(\mathbf{P}) \approx \mathbf{sP}$ and trying to distinguish this from uniform. The assumption says that this is the only additional capability that the adversary obtains, besides its existing strategies for breaking LWE.

Evidently, the distribution of $\mathbf{P}$ here is of crucial importance – for instance, if $\mathbf{P} = \mathbf{0}$, then $\mathbf{B}^{-1}(\mathbf{P})$ is a trapdoor for $\mathbf{B}$ and can be used to easily break LWE. On the other extreme,

if $\mathbf{P}$ is chosen uniformly, then this assumption reduces to standard LWE. The "playing ground" of evasive LWE is in the middle – namely, when it holds that

$$(\mathbf{B}, \mathbf{P}, \mathbf{sB} + \mathbf{e}, \mathbf{sP} + \mathbf{e}') \approx_c (\mathbf{B}, \mathbf{P}, \mathbf{c}, \mathbf{c}')$$

then

$$(\mathbf{B}, \mathbf{sB} + \mathbf{e}, \mathbf{B}^{-1}(\mathbf{P})) \approx_c (\mathbf{B}, \mathbf{c}, \mathbf{B}^{-1}(\mathbf{P})).$$

Here, the former condition is referred to as the PRE condition and the latter as POST. The actual assumption used by the scheme is more complex and includes more LWE samples that use the same secret $\mathbf{s}$ as well as some (carefully chosen) auxiliary information aux. To formalize the PRE condition, the assumption must specify an efficient sampler Samp which outputs the correlated LWE matrices. We defer the formalization to Section 4.5; here we only remark that the assumption captures in the lattice setting, the guarantees provided by the generic group model for pairings, namely the intuition that an adversary can only use legitimate operations to learn anything. It is therefore very natural (in hidsight) that this assumption should be able to replace the reliance on the generic group model in the constructions of [AY20; AWY20].

Tensor LWE. The tensor LWE assumption states that correlated BGG + 18 samples tensored with different random vectors remain pseudorandom. In more detail, for all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, it posits that

$$\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{A} - \mathbf{x}_i \otimes \mathbf{G}) + \mathbf{e}_i,\ \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times \ell m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{mn}, \mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}, \mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}, \mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

Note that there are no Gaussian preimages in the above assumption. In our work, we show that for the special case where $\mathbf{x}_i = 0\ \forall i \in [Q]$, tensor LWE reduces to standard LWE (Lemma 4.10). In more detail, let Adv be an attacker for Tensor LWE with $\mathbf{x}_i = \mathbf{0}$

for all $i \in [Q]$. Adv is given either $\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$ or $\mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$. We prove that under the LWE assumption, Adv has a negligible probability of distinguishing the left hand side from the right hand side. This implication was not known before, and increases our confidence in the assumption, which is new and not so well studied. Please see Lemma 4.10 for details.

Generalizing Tensor LWE. While tensor LWE as stated by Wee suffices for our construction of 2-ABE for P, for extending the arity to any constant k, we require a strengthening of this assumption. In more detail, we require that for all $\mathbf{x}_{j_1,\ldots,j_k} \in \{0,1\}^\ell$ indexed by $j_1, \ldots, j_k \in [Q]$, it holds that:

$$\begin{aligned} & \mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_{1,j_1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_{k,j_k}^\mathsf{T})(\mathbf{A} - \mathbf{x}_{j_1,\ldots,j_k} \otimes \mathbf{G}) + \mathbf{e}_{j_1,\ldots,j_k}, \mathbf{r}_{i,j_i}\right\}_{i\in[k],j_1,\ldots,j_k\in[Q]} \\ \approx_c \quad & \mathbf{A}, \left\{\mathbf{c}_{i,j_i}, \mathbf{r}_{i,j_i}\right\}_{i\in[k],j_1,\ldots,j_k\in[Q]} \end{aligned}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times\ell m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{nm^k}, \mathbf{e}_{j_1,\ldots,j_k} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}, \mathbf{r}_{i,j_i} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}, \mathbf{c}_{i,j_i} \leftarrow \mathbb{Z}_q^{\ell m}$.

It is easy to see that the generalized tensor LWE yields Wee's version of tensor LWE for $k = 1$.

Two Input ABE **from evasive and tensor** LWE**.** As a warmup, we first describe our construction of MIABE for arity 2. For NC_1, our construction can be proven secure by relying solely on evasive LWE while for P, we additionally need tensor LWE. We will show subsequently how to generalize this to any constant arity. In this chapter, we consider a modified syntax of MIABE where there is only a single encryption slot which is public key, and multiple key generation slots, which require the master secret key. This syntax better fits our construction and easily implies the standard definition of MIABE which has multiple encryptors that have as input the master secret key, and a single key generator who also requires the master secret key – please see Section 4.4.1 for details.

Given the above discussion, a natural approach to construct MIABE schemes from lattices

is to try adapting the ideas in Chapter 3 by replacing the use of pairings with tensoring, analogously to Wee's approach of adapting the Agrawal-Yamada broadcast encryption scheme to lattices in Wee. We show that in the end, this approach indeed can be made to work, but via several failed attempts which require new techniques to overcome, and a complex security proof, which requires proving several new lemmas. Below, we outline the pathway to our final construction, detailing the hurdles we encounter and the ideas towards their resolution.

Attempt 1. We attempt to design a scheme using tensor based randomization from Wee to instantiate the template of Chapter 3. We sketch the construction at a high level below. We suppress dimensions for ease of readability in this overview.

1. The master public key is $(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{B}, \mathbf{u})$ where $\mathbf{A}_1, \mathbf{A}_2, \mathbf{B}$ are sampled uniformly and $\mathbf{u}$ is sampled from the discrete Gaussian distribution. The master key is a trapdoor for $\mathbf{A}_0$ and a trapdoor for $\mathbf{B}$.

2. The encryptor, given input $(\mathbf{x}, \mu)$ where $\mathbf{x}$ is the attribute and μ is the message, samples randomness $\mathbf{s}$ along with requisite noise terms and computes

$$\underbrace{\mathbf{s}\mathbf{A}_0 + \mathsf{noise}}_{\mathbf{c}_0}, \quad \underbrace{\mathbf{s}\big((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}\big) + \mathsf{noise}}_{\mathbf{c}_1}, \quad \underbrace{\mathbf{s}(\mathbf{G}\mathbf{u}^\top \otimes \mathbf{I}) + \mathsf{noise}}_{\mathbf{c}_2}, \quad \underbrace{\mathbf{s}\mathbf{B} + \mathsf{noise}}_{\mathbf{c}_3}$$

   if $\mu = 0$ and else samples random elements of appropriate dimensions if $\mu = 1$. Note that the encryption procedure is public key.

3. The first key generator (to be interpreted as the second encryptor), given input msk and attribute $\mathbf{y}$ samples Gaussian random vector $\mathbf{r}$ and computes

$$\mathsf{sk}_{\mathbf{y}} = \mathbf{B}^{-1}\big((\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}\big), \mathbf{r}^\mathsf{T}$$

   It outputs this as the secret key for $\mathbf{y}$. Note that the randomizer $\mathbf{r}$ is used to prevent collusion attacks – in its absence, an attacker can obtain samples corresponding to $\mathbf{y}$ and $\overline{\mathbf{y}}$ (i.e. complement of $\mathbf{y}$) and launch attack as discussed earlier.

4. The second key generator, given msk and function f as input computes $\mathsf{sk}_f = (\mathbf{A}_0 \| \mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^\top)$ and outputs this as the secret key for f.

5. The decryptor does the following:

a) *Computing ciphertext component for second attribute:* It combines the ciphertext $\mathbf{c}_3$ with the first secret key $\mathsf{sk}_{\mathbf{y}}$ to obtain $\mathbf{s}\big((\mathbf{A}_2 - \mathbf{y}\otimes\mathbf{G})\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$.

b) *Randomizing ciphertext component for first attribute:* From $\mathbf{c}_1$ and $\mathsf{sk}_{\mathbf{y}}$, it computes $\big(\mathbf{s}\big((\mathbf{A}_1 - \mathbf{x}\otimes\mathbf{G})\otimes\mathbf{I}\big) + \mathsf{noise}\big)(\mathbf{I}\otimes\mathbf{r}^{\mathsf{T}}) = \mathbf{s}\big((\mathbf{A}_1 - \mathbf{x}\otimes\mathbf{G})\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$

c) *Producing a complete* BGG + 18 *ciphertext:* Concatenating the results of the previous two steps, we get

$$\mathbf{s}\big((\mathbf{A}_1\|\mathbf{A}_2) - (\mathbf{x}\|\mathbf{y})\otimes\mathbf{G})\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$$

Note that this looks exactly like a BGG + 18 sample except for the tensoring with $\mathbf{r}^{\mathsf{T}}$. As discussed above, Wee shows that homomorphic computation is preserved under right tensoring with $\mathbf{r}^{\mathsf{T}}$.

d) BGG + 18 *Homomorphic evaluation:* Computing the circuit f homomorphically on this BGG + 18 sample, we obtain

$$\mathbf{s}\big((\mathbf{A}_f - f(\mathbf{x},\mathbf{y})\mathbf{G})\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$$

If $f(\mathbf{x},\mathbf{y}) = 0$, then we get $\mathbf{s}\big(\mathbf{A}_f\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$. Concatenating with the ciphertext component $\mathbf{c}_0$, we get

$$\mathbf{s}\big(\mathbf{A}_0\|\mathbf{A}_f)\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$$

e) *Applying* BGG + 18 *secret key.* By right multiplying the second slot secret key $(\mathbf{A}_0\|\mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^{\top})\otimes\mathbf{I}$ to this, we get

$$\mathbf{s}\big(\mathbf{G}\mathbf{u}^{\top}\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$$

f) BGG + 18 *decryption with tensoring.* Multiplying $\mathbf{c}_2$ with $\mathbf{I}\otimes\mathbf{r}^{\mathsf{T}}$, we get $\mathbf{s}\big(\mathbf{G}\mathbf{u}^{\top}\otimes\mathbf{r}^{\mathsf{T}}\big) + \mathsf{noise}$. Subtracting from the output of the previous step, we get a small value when $\mu = 0$. Thus, we recover μ when $f(\mathbf{x},\mathbf{y}) = 0$.

The above scheme provides functionality and does not appear to have any immediate attacks. However, we are unable to prove security of this scheme based on the evasive/tensor LWE assumption. This is because the evasive LWE assumption accommodates Gaussian preimages for fixed matrices, namely terms of the form $\mathbf{B}^{-1}(\mathbf{P})$, where $\mathbf{B}$ is a random matrix and $\mathbf{P}$ is structured, but does not know how to handle terms such as $(\mathbf{A}_0\|\mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^{\mathsf{T}})$. Since $\mathbf{A}_f$ is highly structured, this is incompatible with the assumption.

Attempt 2. To handle this barrier, in our next attempt, we use an idea by Wee to remove the problematic term $(\mathbf{A}_0\|\mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^\mathsf{T})$. Note that the purpose of this term is to create an LWE sample with secret $\mathbf{s}$ and matrix $\mathbf{A}_f$. In more detail, as shown in step 5e, the term $\mathbf{s}\big(\mathbf{A}_0\|\mathbf{A}_f) \otimes \mathbf{r}^\mathsf{T}\big) + \mathsf{noise}$ obtained by homomorphic evaluation is combined together with the secret key in the second slot $(\mathbf{A}_0\|\mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^\mathsf{T})$ to obtain $\mathbf{s}(\mathbf{G}\mathbf{u}^\top \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$. As shown in step 5f, this term is then used to unmask the ramdomized $\mathbf{c}_2$, i.e. $\mathbf{s}\big(\mathbf{G}\mathbf{u}^\top \otimes \mathbf{r}^\mathsf{T}\big) + \mathsf{noise}$ by subtraction to recover μ.

So as to do away with the requirement of revealing $(\mathbf{A}_0\|\mathbf{A}_f)^{-1}(\mathbf{G}\mathbf{u}^\mathsf{T})$, we provide an alternate route to recover μ. We change the second slot secret key sk_f to $\mathbf{B}^{-1}(\mathbf{A}_f\mathbf{u} \otimes \mathbf{I})$, and use this together with the term $\mathbf{s}\mathbf{B} + \mathsf{noise}$ provided in the ciphertext to obtain $\mathbf{s}(\mathbf{A}_f\mathbf{u} \otimes \mathbf{I}) + \mathsf{noise}$. This allows us to cancel the mask $\mathbf{s}\mathbf{A}_f$ obtained via homomorphic evaluation and brings us closer to relying only on evasive and tensor LWE.

Below, we detail only the modifications we make to our previous attempt:

1. The encryptor, given input $(\mathbf{x}, \mu)$ where $\mathbf{x}$ is the attribute and μ is the message, samples randomness $\mathbf{s}$ along with requisite noise terms and computes

$$\underbrace{\cancel{\mathbf{s}\mathbf{A}_0 + \mathsf{noise}}}_{\mathbf{c}_0},\ \underbrace{\mathbf{s}\big((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}\big) + \mathsf{noise}}_{\mathbf{c}_1},\ \underbrace{\cancel{\mathbf{s}(\mathbf{G}\mathbf{u}^\top \otimes \mathbf{I}) + \mathsf{noise}}}_{\mathbf{c}_2},\ \underbrace{\mathbf{s}\mathbf{B} + \mathsf{noise}}_{\mathbf{c}_3}$$

   if $\mu = 0$ else samples random elements of appropriate dimensions if $\mu = 1$.

2. The second key generator, given msk and function f computes $\mathsf{sk}_f = \mathbf{B}^{-1}(\mathbf{A}_f\mathbf{u}^\top \otimes \mathbf{I})$. At this junction, we let $\mathbf{u}$ be chosen independently by each user instead of fixing it in the public parameters to prevent the adversary from requesting keys for correlated functions and obtaining correlated LWE samples of the form $\mathbf{s}\mathbf{A}_f\mathbf{u}^\mathsf{T} + \mathsf{noise}$ with the same $\mathbf{u}$ and same $\mathbf{s}$.

3. During decryption,

   a) BGG + 18 homomorphic evaluation is simplified. We only compute the circuit f homomorphically on this BGG + 18 sample, to obtain

$$\mathbf{s}\big((\mathbf{A}_f - f(\mathbf{x}, \mathbf{y})\mathbf{G}) \otimes \mathbf{r}^\mathsf{T}\big) + \mathsf{noise}$$

   If $f(\mathbf{x}, \mathbf{y}) = 0$, then we get $\mathbf{s}\big(\mathbf{A}_f \otimes \mathbf{r}^\mathsf{T}\big) + \mathsf{noise}$. There is no need to concatenate with $\mathbf{c}_0$ (this is no longer even provided) but we must right multiply by $(\mathbf{u}^\top \otimes \mathbf{I})$

to obtain $\mathbf{s}\left(\mathbf{A}_f\mathbf{u}^\top \otimes \mathbf{r}^\mathsf{T}\right) + \mathsf{noise}$. Recall that $\mathbf{u}$ is low norm, hence does not blow up the noise.

b) The second slot key $\mathbf{B}^{-1}(\mathbf{A}_f\mathbf{u}^\top \otimes \mathbf{I})$ is right multiplied to $\mathbf{c}_3$ to get $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\top \otimes \mathbf{I}) + \mathsf{noise}$. By right multiplying with $(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})$, we now recover the masking term $\mathbf{s}\left(\mathbf{A}_f\mathbf{u} \otimes \mathbf{r}^\mathsf{T}\right) + \mathsf{noise}$ which can be subtracted from the output of the previous step. If this is small, learn that $\mu = 0$.

Importantly, at this point, we can hope to use evasive LWE to "get rid" of the preimages $\mathbf{B}^{-1}\left((\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}\right)$ and $\mathbf{B}^{-1}(\mathbf{A}_f\mathbf{u}^\top \otimes \mathbf{I})$ from the distribution seen by the adversary. This essentially reduces the task of proving the security of the scheme to that of proving the pseudorandomness of the terms

$$\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathsf{noise}, \quad \mathbf{s}((\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}, \quad \mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$$

Unfortunately, we are still not done, even by relying additionally on tensor LWE. This is because tensor LWE only posits pseudorandomness of LWE samples with respect to secret $\mathbf{s}(\mathbf{I} \otimes \mathbf{r})$. In particular, the presence of the terms $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathsf{noise}$ and $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$ cannot be handled by invoking tensor LWE since they do not have the right form (in particular no $\mathbf{r}$ term appears in these). Therefore, we must handle these next.

Attempt 3. Let us first explain how to deal with the first term $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathsf{noise}$. As in Wee, the idea is to "mask" the problematic term, in this case, $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathsf{noise}$, with a pseudorandom term $\mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathsf{noise}$ such that there is a way to provide an "unmasking" term using which, we can recover a simulatable term $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ but nothing else is revealed [2].

In more detail, we make the following changes:

1. We replace $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathsf{noise}$ by $\mathbf{c} = \mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathsf{noise}$.
2. Next, we put some terms so that the ciphertext along with the first slot of the secret

[2] The informed reader may notice the similarity with randomized encodings [AIK04] and pair/predicate encodings [Att14; Wee14].

key jointly generates $\mathbf{d} := \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$, which is an "unmasking" term.

3. To obtain the desired term, we compute $\mathbf{c}(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d} = \mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}) + \mathsf{noise}$.

Furthermore, it is easy to show that $\mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathsf{noise}$ is pseudorandom by LWE (since $\mathbf{s}_0$ is a fresh randomness introduced only for this specific purpose), which implies that $\mathbf{c}$ is also pseudorandom. This allows us to conclude that $\mathbf{d}$ does not reveal anything more than the desired term, since $\mathbf{c}$ and the desired term determine $\mathbf{d}$.

At this stage, the scheme looks like the following, where for brevity we again omit to mention components that are unchanged.

1. The encryptor computes $\mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0) \begin{pmatrix} (\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I} \end{pmatrix} + \mathsf{noise}$ and $\mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0)\mathbf{B} + \mathsf{noise}$ for $\mu = 0$ (and random elements for $\mu = 1$).

2. The first slot key is $\mathsf{sk}_\mathbf{y} \leftarrow \mathbf{B}^{-1} \begin{pmatrix} (\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T} & \\ & \mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T} \end{pmatrix}$

3. The second slot key is $\mathsf{sk}_f \leftarrow \mathbf{B}^{-1} \begin{pmatrix} \mathbf{A}_f \mathbf{u}^\mathsf{T} \otimes \mathbf{I} \\ \mathbf{0} \end{pmatrix}$ and $\mathbf{u}$. This key is essentially unchanged except padding the inner matrix with zeroes to account for the longer secret.

4. Now, from the ciphertext component $\mathbf{c}_2$ and the first slot key, we get terms $\mathbf{s}(\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ and $\mathbf{d} = \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$. The second term $\mathbf{d}$ is the new term that we will make use of as described above.

5. Now, we compute $\mathbf{c}(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d} = \mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}) + \mathsf{noise}$. Using pseudorandomness of $\mathbf{c}$, we can argue that $\mathbf{d}$ did not reveal anything except $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}) + \mathsf{noise}$.

At this stage, we obtained a term that tensor LWE can handle, namely $\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}) + \mathsf{noise}$.

Attempt 4. Next, we must deal with the second problematic term $\mathbf{s}(\mathbf{A}_f \mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$. It is tempting to try the same strategy as above but unfortunately, this does not work. To see why, let us try to replace $\mathbf{s}(\mathbf{A}_f \mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$ with $\mathbf{s}(\mathbf{A}_f \mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{I}) + \mathsf{noise}$,

where $\mathbf{D}$ is some fixed matrix. We can then modify the scheme so that the ciphertext along with the first slot secret key generate the unmasking term $\mathbf{s}_1(\mathbf{D} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$. Similarly to the above, this allows us to derive the desired term $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ which can be handled by tensor LWE. One may hope that this suffices to prove security.

However, we run into another problem, namely, that of collusion resistance. In particular, an adversary may make multiple key queries for the second slot and use the same ciphertext and first slot key for decryption. These allow her to recover $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T}\otimes\mathbf{I})+\mathbf{s}_1(\mathbf{D}\otimes\mathbf{r}^\mathsf{T})+\mathsf{noise}$ and $\mathbf{s}(\mathbf{A}_{f'}\mathbf{u}'^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ for different f and f'. Even though we want to hide two terms $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I})$ and $\mathbf{s}(\mathbf{A}_{f'}\mathbf{u}'^\mathsf{T} \otimes \mathbf{I})$, there is only a single masking term $\mathbf{s}_1(\mathbf{D} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$, since $\mathbf{s}_1$ would be chosen by the encryptor and $\mathbf{r}$ by the first slot key – this is clearly problematic.

To fix this, we ensure that the masking term is randomized by a user specific randomness corresponding to the second slot key. Namely, we replace $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{I}) + \mathsf{noise}$ with $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) + \mathsf{noise}$, where $\mathbf{t}$ is user specific randomness. We then use the ideas discussed previously to ensure that the ciphertext and second slot key generate $\mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) + \mathsf{noise}$. This mask is removed similarly to the previous case and we may obtain $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$.

Attempt 5. Unfortunately, this still does not suffice. Recall that we wanted to generate the term $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ in order to invoke tensor LWE, which the above term does not let us do. To achieve this, we replace $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}) + \mathsf{noise}$ with $\mathbf{s}(\mathbf{A}_f\mathbf{u}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t} \otimes \mathbf{I}) + \mathsf{noise}$, i.e., we added some space to further randomize the masking term with $\mathbf{r}^\mathsf{T}$. We then let the ciphertext and secret keys for both slots jointly generate $\mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$.

To do so, we do the following:

1. Include $\mathbf{s}_1\mathbf{B} + \mathsf{noise}$ in the ciphertext and $\mathbf{B}^{-1}(\mathbf{C} \otimes \mathbf{r}^\mathsf{T})$ in the first slot key.

Multiplying them yields $\mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$.

2. Include $\mathbf{C}^{-1}(\mathbf{D} \otimes \mathbf{t}^\mathsf{T})$ in the second slot key.

Putting these together enables us to recover the masking term as:

$$\begin{aligned}(\mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}) \cdot \mathbf{C}^{-1}(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) &= \mathbf{s}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})\mathbf{C} \cdot \mathbf{C}^{-1}(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) + \mathsf{noise} \\ &= \mathbf{s}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) + \mathsf{noise} \\ &= \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}\end{aligned}$$

The above term contains randomness $\mathbf{s}_1$ chosen by the encryptor, $\mathbf{r}$ chosen by the first slot key and $\mathbf{t}$ chosen by the second slot key. Intuitively, this randomness triple separates the triple of ciphertext, first key and second key, from any other triple even if some components of the triple are reused. This allows to separate the "thread" of computation corresponding to a given triple, from all other threads, and hopefully allows us to prove security. This brings us to our final scheme.

We provide the complete construction below. The vector $\mathbf{u}$ above is now changed to a matrix $\mathbf{U}$ for syntactic reasons.

1. Set $\mathsf{mpk} = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{B}, \mathbf{C}, \mathbf{D})$, and msk as trapdoors for $\mathbf{B}$ and $\mathbf{C}$.

2. To encrypt a message μ against attribute $\mathbf{x}$, do the following. If $\mu = 0$, do:

   a) Compute $\mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0)\begin{pmatrix}(\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I}\end{pmatrix} + \mathsf{noise}$

   b) Compute $\mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1)\mathbf{B} + \mathsf{noise}$

   c) Output $\mathsf{ct}_\mathbf{x} = (\mathbf{c}_1, \mathbf{c}_2)$

   If $\mu = 1$, output random elements in the appropriate space.

3. To compute the first slot key for attribute $\mathbf{y}$, sample

$$\mathsf{sk}_\mathbf{y} \leftarrow \mathbf{B}^{-1}\begin{pmatrix}(\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T} & & \\ & \mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T} & \\ & & \mathbf{C} \otimes \mathbf{r}^\mathsf{T}\end{pmatrix}$$

4. To compute the second slot key for function f, sample $\mathbf{U}, \mathbf{t}$ and compute

$$\mathsf{sk}_f \leftarrow \mathbf{B}^{-1}\begin{pmatrix}\mathbf{A}_f\mathbf{U}\otimes\mathbf{I}\\ \mathbf{0}\\ \mathbf{D}\otimes\mathbf{t}^\mathsf{T}\otimes\mathbf{I}\end{pmatrix}, \quad \mathbf{C}^{-1}\left(\mathbf{D}\otimes\mathbf{t}^\mathsf{T}\right), \quad \mathbf{U}, \quad \mathbf{t}$$

To decrypt, first compute $\mathbf{d}_1 = \mathbf{c}_1(\mathbf{I}\otimes\mathbf{r}^\mathsf{T})$, $(\mathbf{d}_2,\ \mathbf{d}_3,\ \mathbf{d}_4) = \mathbf{c}_2\cdot\mathsf{sk}_\mathbf{y}$, $\mathbf{d}_5 = \mathbf{c}_2\cdot\mathsf{sk}_{f,1}$, $\mathbf{d}_6 = \mathbf{d}_5(\mathbf{I}\otimes\mathbf{r}^\mathsf{T})$ and $\mathbf{d}_7 = \mathbf{d}_4\cdot\mathsf{sk}_{f,2}$. Then compute $\mathbf{d}_8 = \mathbf{d}_1 - \mathbf{d}_3$, $\mathbf{d}_9 = (\mathbf{d}_8\|\mathbf{d}_2)\widehat{\mathbf{H}}_{(\mathbf{A}_1\|\mathbf{A}_2),f,(\mathbf{x}\|\mathbf{y})}\mathbf{U}$, $\mathbf{d}_{10} = \mathbf{d}_6 - \mathbf{d}_7$. Finally, if $\mathbf{d}_{10} - \mathbf{d}_9 \approx 0$, then output 0, else 1. To see the correctness, observe:

$$\mathbf{d}_1 = \mathbf{s}((\mathbf{A}_1 - \mathbf{x}\otimes\mathbf{G})\otimes\mathbf{r}^\mathsf{T}) + \mathbf{s}_0(\mathbf{A}_0\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise},$$

$$\underline{\mathbf{d}_2 = \mathbf{s}((\mathbf{A}_2 - \mathbf{y}\otimes\mathbf{G})\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise}},$$

$$\mathbf{d}_3 = \mathbf{s}_0(\mathbf{A}_0\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise}, \quad \mathbf{d}_4 = \mathbf{s}_1(\mathbf{C}\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise},$$

$$\mathbf{d}_5 = \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{I}) + \mathbf{s}_1(\mathbf{D}\otimes\mathbf{t}^\mathsf{T}\otimes\mathbf{I}) + \mathsf{noise}, \quad \mathbf{d}_6 = \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{r}^\mathsf{T}) + \mathbf{s}_1(\mathbf{D}\otimes\mathbf{t}^\mathsf{T}\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise},$$

$$\mathbf{d}_7 = \mathbf{d}_4\cdot\mathbf{C}^{-1}(\mathbf{D}\otimes\mathbf{t}^\mathsf{T}) = \mathbf{s}_1(\mathbf{D}\otimes\mathbf{t}^\mathsf{T}\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise}, \quad \underline{\mathbf{d}_8 = \mathbf{s}((\mathbf{A}_1 - \mathbf{x}\otimes\mathbf{G})\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise}}$$

$$\mathbf{d}_9 = \mathbf{s}((\mathbf{A}_f - f(\mathbf{x},\mathbf{y})\mathbf{G})\otimes\mathbf{r}^\mathsf{T})\mathbf{U} + \mathsf{noise}, \quad \mathbf{d}_{10} = \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{r}^\mathsf{T}) + \mathsf{noise}$$

If $f(\mathbf{x},\mathbf{y}) = 0$, then $\mathbf{d}_{10} - \mathbf{d}_9 = \mathsf{noise}$ when $\mu = 0$, else it is large. Above, the underlined terms $\mathbf{d}_2, \mathbf{d}_8$ mimic the ciphertext components of single input $\mathsf{BGG+18}$, computed as if with shared randomness by a single party holding both $\mathbf{x}$ and $\mathbf{y}$. Note that all the machinery developed above was to be able to simulate the single party setting in the two party setting, where the ciphertexts are produced using independent randomness.

Proof Sketch. For ease of exposition, we sketch the proof for the case where only a single key is generated for both the slots. First, we observe that we need to invoke evasive LWE twice, once to handle terms $\mathbf{B}^{-1}(\cdot)$ and once for $\mathbf{C}^{-1}(\cdot)$. Of these, the first application is standard, following Wee while the second one requires more care as it uses a structured LWE, as in [VWW23].

Having removed Gaussian preimages with respect to $\mathbf{B}$ and $\mathbf{C}$, we are required to show pseudorandomness of the following terms:

$$\mathbf{c}_1 = \mathbf{s}((\mathbf{A}_1 - \mathbf{x}\otimes\mathbf{G})\otimes\mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0\otimes\mathbf{I}) + \mathsf{noise}, \quad \mathbf{c}_2 = (\mathbf{s},\mathbf{s}_0,\mathbf{s}_1)\mathbf{B} + \mathsf{noise},$$

$$\mathbf{c}_3 = \mathbf{s}((\mathbf{A}_2 - \mathbf{y} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise} \quad \mathbf{c}_4 = \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise},$$

$$\mathbf{c}_5 = \mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}, \quad \mathbf{c}_6 = \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise}$$

$$\mathbf{c}_7 = \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$$

Above, note that $\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5$ are generated using the secret key for the first slot and the ciphertext, $\mathbf{c}_6$ is generated using the ciphertext and secret key of the second slot, and $\mathbf{c}_7$ is generated using evasive LWE with structured secret, namely by combining $\mathbf{C}^{-1}(\mathbf{D} \otimes \mathbf{t}^\mathsf{T})$ and $\mathbf{c}_5 = \mathbf{s}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})\mathbf{C} + \mathsf{noise}$. This yields $\mathbf{s}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T})(\mathbf{D} \otimes \mathbf{t}^\mathsf{T}) + \mathsf{noise}$ which is equal to $\mathbf{c}_7$.

We now proceed to sketch the hybrid structure of the proof.

Game 0: This is the real game.

Game 1: Express $\mathbf{c}_4$ in terms of $\mathbf{c}_1$ and a term that tensor LWE can handle:

$$\mathbf{c}_4 = \mathbf{c}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \underbrace{\left(\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}\right)}_{\mathbf{c}_{4'}}$$

The only difference between Game 0 and Game 1 is the distribution of the noise term which can be handled by noting that $\mathbf{c}_1(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) \approx \mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T})$ and using the standard smudging lemma (Lemma 4.5).

Game 2: We now change $\mathbf{c}_1$ and $\mathbf{c}_2$ to random by using the power of LWE with secret $\mathbf{s}_0$.

Game 3: Now, we express $\mathbf{c}_7$ in terms of $\mathbf{c}_6$ and a term which is friendly with tensor LWE:

$$\mathbf{c}_7 = \mathbf{c}_6(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \underbrace{\mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}}_{\mathbf{c}'_7}$$

Again, the change follows using the smudging lemma.

Game 4: Change $\mathbf{c}_5$ and $\mathbf{c}_6$ to random. Note that $\mathbf{c}_6(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) \approx \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$ and $\mathbf{c}_5 = \mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}$. Hence, it suffices to show pseudorandomness of

$$\mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathsf{noise}, \quad \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{I}) + \mathsf{noise})$$

We argue this via a new lemma by using only (standard) LWE.

Game 5: At this point it remains to argue that $\mathbf{c}_3$, $\mathbf{c}'_4$ and $\mathbf{c}'_7$ are pseudorandom.

These constitute:

$$\mathbf{s}(\mathbf{I}\otimes\mathbf{r}^\top)\big((\mathbf{A}_1\|\mathbf{A}_2)-(\mathbf{x}\|\mathbf{y})\otimes\mathbf{G}\big)+\mathsf{noise},\quad \mathbf{s}(\mathbf{I}\otimes\mathbf{r}^\top)(\mathbf{A}_f\mathbf{U})+\mathsf{noise}$$

and we can directly plug in the tensor LWE assumption to argue this.

Please see Section 4.6 for the detailed proof.

Extension to Constant Arity. Next, we outline how to extend the above idea to the setting of constant arity. The basic idea is to let the secret key for slot $i \in [k]$ generate

$$\mathbf{s}((\mathbf{A}-\mathbf{x}_i\otimes\mathbf{G})\otimes\mathbf{I}\otimes\mathbf{r}_i^\top\otimes\mathbf{I})+\underbrace{\mathbf{s}_i(\mathbf{D}\otimes\mathbf{I}\otimes\mathbf{r}_i^\top\otimes\mathbf{I})+\mathsf{noise}}_{\text{masking term}}$$

where $\mathbf{r}_i$ is the user specific randomness associated with the secret key for the i-th slot.

In addition, we also prepare other terms so that the ciphertext and secret keys can collaboratively generate the unmasking terms as:

$$\mathbf{s}_i(\mathbf{D}\otimes\mathbf{r}_1^\top\otimes\cdots\otimes\mathbf{r}_k^\top)+\mathsf{noise}\quad\forall i\in[k]$$

Given the unmasking term, the decryptor can obtain

$$\mathbf{s}((\mathbf{A}_i-\mathbf{x}_i\otimes\mathbf{G})\otimes\mathbf{r}_1^\top\otimes\cdots\otimes\mathbf{r}_k^\top)+\mathsf{noise}$$

A similar strategy also works for masking $\mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{I})$ and we can show that the adversary can only obtain

$$\mathbf{s}((\mathbf{A}-\mathbf{x}\otimes\mathbf{G})\otimes\mathbf{r}_1^\top\otimes\cdots\otimes\mathbf{r}_k^\top)+\mathsf{noise},\quad \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{r}_1^\top\otimes\cdots\otimes\mathbf{r}_k^\top)+\mathsf{noise}$$

which are LWE samples w.r.t randomness $\mathbf{s}(\mathbf{I}\otimes\mathbf{r}_1^\top\otimes\cdots\otimes\mathbf{r}_k^\top)$. We refer the reader to Section 4.7 for the complete construction.

On Circuit Depth. As discussed above, for our MIABE for NC_1, we rely only on evasive LWE, even for constant arity. For our MIABE for P, we require evasive and tensor

LWE for arity 2, but for general k, we need to generalize tensor LWE as discussed above.

To remove the need for (any) tensor LWE in the restricted case of NC_1 circuits, we use low norm $\mathbf{A}_i$ and switch out $\mathbf{G}$ for $\mathbf{I}$, as suggested by Wee. We also leverage the observation by Wee, that a weaker version of homomorphic computation is still possible in this setting. In addition, we show that when $\mathbf{A}_i$ and $\mathbf{G}$ are changed as above, LWE samples w.r.t $\mathbf{x}$ obtained by combining ciphertexts and secret keys are indistinguishable from those that are computed using fresh randomness for all combinations of ciphertexts and secret keys.

In more detail, let $\mathbf{i} = (i_1, \ldots, i_k)$ denote the ciphertext queries in the k slots which are being combined for decryption. Then, we show that

$$\left\{\mathbf{s}\left((\mathbf{A}-\mathbf{x}^{\mathbf{i}}\otimes\mathbf{I})\otimes{\mathbf{r}_1^{i_1}}^{\mathsf{T}}\otimes\ldots\otimes{\mathbf{r}_1^{i_k}}^{\mathsf{T}}\right)+\mathsf{noise}\right\}_{i_1,\ldots,i_k\in[Q]} \approx_c \left\{\mathbf{s}_{i_1,\ldots,i_k}\left(\mathbf{A}-\mathbf{x}^{\mathbf{i}}\otimes\mathbf{I}\right)+\mathsf{noise}\right\}_{i_1,\ldots,i_k\in[Q]}$$

where $\mathbf{s}_{i_1,\ldots,i_k}$ is a unique, freshly sampled secret for the combination $\mathbf{i} = (i_1, \ldots, i_k)$. Intuitively, the shortness of $\mathbf{A}$ and $\mathbf{I}$ is used to argue that:

$$\mathbf{s}\left((\mathbf{A}-\mathbf{x}^{\mathbf{i}}\otimes\mathbf{I})\otimes{\mathbf{r}_1^{i_1}}^{\mathsf{T}}\otimes\ldots\otimes{\mathbf{r}_k^{i_k}}^{\mathsf{T}}\right)+\mathsf{noise} \approx_c \left(\mathbf{s}(\mathbf{I}\otimes{\mathbf{r}_1^{i_1}}^{\mathsf{T}}\otimes\ldots\otimes{\mathbf{r}_k^{i_k}}^{\mathsf{T}})+\mathsf{noise}\right)\left(\mathbf{A}-\mathbf{x}^{\mathbf{i}}\otimes\mathbf{I}\right)+\mathsf{noise}$$

which in turn allows to express $\mathbf{s}(\mathbf{I}\otimes{\mathbf{r}_1^{i_1}}^{\mathsf{T}}\otimes\ldots\otimes{\mathbf{r}_k^{i_k}}^{\mathsf{T}})+\mathsf{noise}$ as $\mathbf{s}_{i_1,\ldots,i_k}$ by iteratively separating out ${\mathbf{r}_j^{i_j}}^{\mathsf{T}}$, and adding noise to obtain a fresh secret[3]. Please see Section 4.7 for details.

Organisation of the chapter. The rest of the chapter is organised as follows. In Section 4.4, we provide the preliminaries used in this chapter. In Section 4.5, we discuss evasive and tensor LWE assumptions and define new implications of tensor LWE. We construct 2ABE for P from evasive and tensor LWE in Section 4.6. In Section 4.7, we provide the constructions for constant arity, where we construct MIABE for NC_1 and for P in Sections 4.7.1 and 4.7.3, respectively.

[3]The informed reader may be reminded of the Naor-Reingold argument [NR97] used to construct a PRF from DDH or its lattice analogue [BPR12].

4.4 PRELIMINARIES

Notation used in this chapter. We begin by defining the notation used in this chapter. We use the same notation as in Chapter 3, unless otherwise stated. By default, a vector is a row vector in this chapter. In addition, we use the following notation. For any two vectors $\mathbf{x}$ and $\mathbf{y}$ (resp. matrices $\mathbf{X}$, $\mathbf{Y}$), $\mathbf{x}\|\mathbf{y}$ (resp. $\mathbf{X}\|\mathbf{Y}$) represents horizontal concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$ (resp. matrices $\mathbf{X}$ and $\mathbf{Y}$). For any $n > 0$, $\mathbf{I}_n$ represents an identity matrix of size n. When $n = m$, we denote $\mathbf{I}_m$ by only $\mathbf{I}$ and $\mathbf{I}^{\otimes i}$ denotes $\mathbf{I}_{m^i} = \underbrace{\mathbf{I} \otimes \cdots \otimes \mathbf{I}}_{i \text{ times}}$ for any integer i.

For two distributions $\mathcal{D}_1$, $\mathcal{D}_2$ the notation $\mathcal{D}_1 \approx_c \mathcal{D}_2$ (resp. $\mathcal{D}_1 \approx_s \mathcal{D}_2$) are defined in the same way as in Chapter 3. In addition, we write $\mathcal{D}_1 \equiv \mathcal{D}_2$ when $\mathcal{D}_1$ and $\mathcal{D}_2$ are perfectly indistinguishable.

4.4.1 Multi-Input Attribute Based Encryption

Following Chapter 3, we define multi-input Attribute Based Encryption (ABE) below. As described in the introduction, we use a modified syntax in this chapter to better align with the syntax in [Wee22]. A k-input ABE scheme is parametrized over an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ and function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where each function maps $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$ to $\{0,1\}$. Such a scheme is described by procedures $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}_1, \ldots, \mathsf{KeyGen}_{k-1}, \mathsf{KeyGen}_k, \mathsf{Dec})$ with the following syntax:

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

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0, \mu) \to \mathsf{ct}_{\mathbf{x}_0,\mu}$**:** The encryption algorithm takes as input the master public key mpk, an attribute $\mathbf{x}_0 \in A_\lambda$, and message $\mu \in \{0,1\}$, and outputs a ciphertext $\mathsf{ct}_{\mathbf{x}_0,\mu}$. The attribute string $\mathbf{x}_0$ is also included as part of the ciphertext.

$\mathsf{KeyGen}_i(\mathsf{msk}, \mathbf{x}_i) \to \mathsf{sk}_{i,\mathbf{x}_i}$ **for** $1 \le i \le k-1$**:** The KeyGen algorithm for the i^{th} slot

where $i \in [k-1]$, takes as input the master secret key msk, and an attribute $\mathbf{x}_i \in A_\lambda$ and outputs a key for slot i, $\mathsf{sk}_{i,\mathbf{x}_i}$. Again, we assume that the attribute string $\mathbf{x}_i$ is included as part of the secret key.

$\mathsf{KeyGen}_k(\mathsf{msk}, f) \to \mathsf{sk}_{k,f}$**:** The KeyGen algorithm for slot k takes as input the master secret key msk and a function $f \in \mathcal{F}_\lambda$ and outputs a key $\mathsf{sk}_{k,f}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}_{\mathbf{x}_0,\mu}, \mathsf{sk}_{1,\mathbf{x}_1}, \ldots, \mathsf{sk}_{k-1,\mathbf{x}_{k-1}}, \mathsf{sk}_{k,f}) \to \mu'$**:** The decryption algorithm takes as input a ciphertext $\mathsf{ct}_{\mathbf{x}_0,\mu}$, k keys $\mathsf{sk}_{1,\mathbf{x}_1}, \ldots, \mathsf{sk}_{k-1,\mathbf{x}_{k-1}}$, and $\mathsf{sk}_{k,f}$ and outputs a string μ'.

Next, we define correctness and security. For ease of notation, we drop the subscript λ in what follows.

Correctness: For every $\lambda \in \mathbb{N}, \mu \in \{0,1\}, \mathbf{x}_0, \ldots, \mathbf{x}_{k-1} \in A, f \in \mathcal{F}$, it holds that if $f(\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}) = 0$,[4] then

$$\Pr\left[\mathsf{Dec}\left(\begin{array}{c}\mathsf{mpk},\ \ \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0, \mu), \\ \mathsf{KeyGen}(\mathsf{msk}, \mathbf{x}_1), \ldots, \mathsf{KeyGen}_{k-1}(\mathsf{msk}, \mathbf{x}_{k-1}), \mathsf{KeyGen}_k(\mathsf{msk}, f)\end{array}\right) = \mu\right] = 1 - \mathrm{negl}(\lambda)$$

where the probability is over the choice of $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and over the internal randomness of Enc and $\mathsf{KeyGen}_1, \ldots, \mathsf{KeyGen}_k$.

Definition 4.1 ($\mathsf{Ada\text{-}IND}$ security for $\mathsf{k\text{-}ABE}$)**.** For a $\mathsf{k\text{-}ABE}$ scheme $\mathsf{k\text{-}ABE} = \{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}_1, \ldots, \mathsf{KeyGen}_{k-1}, \mathsf{KeyGen}_k, \mathsf{Dec}\}$, for an attribute space $\{(A_\lambda)^k\}_{\lambda\in\mathbb{N}}$, function space $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and an adversary A, we define the $\mathsf{Ada\text{-}IND}$ security game as follows.

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

[4] We follow the convention in lattice based cryptography where the decryption condition is reversed with respect to the output of the function.

2. **Query phase:** During the game, A adaptively makes the following queries, in an arbitrary order.

   a) **Key Queries:** A makes polynomial number of key queries for each slot, say $p = p(\lambda)$. As a j-th query for slot i, $\mathcal{A}$ chooses

   $$\begin{cases} \mathbf{x}_{i,j} & \text{if } i \in [k-1] \\ f_j & \text{if } i = k, \end{cases}$$

   where $\mathbf{x}_{i,j} \in A_\lambda$ and $f_j \in \mathcal{F}_\lambda$. The challenger computes

   $$\begin{cases} \mathsf{sk}_{i,\mathbf{x}_{i,j}} = \mathsf{KeyGen}_i(\mathsf{msk}, \mathbf{x}_{i,j}) & \text{if } i \in [k-1] \\ \mathsf{sk}_{f_j} = \mathsf{KeyGen}_k(\mathsf{msk}, f_j) & \text{if } i = k \end{cases}$$

   and returns it to $\mathcal{A}$.

   b) **Challenge Query:** A issues a challenge query for encryption. A declares $(\mathbf{x}_0, (\mu_0, \mu_1))$ to the challenger, where $\mathbf{x}_0 \in A_\lambda$ is an attribute and $(\mu_0, \mu_1) \in \{0,1\} \times \{0,1\}$ is the pair of messages. Then, the challenger samples $\beta \leftarrow \{0,1\}$, computes $\mathsf{ct}_\beta = \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0, \mu_\beta)$ and returns it to $\mathcal{A}$.

3. **Output phase:** A outputs a guess bit β' as the output of the experiment.

For the adversary to be *admissible*, we require that for every $f_1, \ldots, f_p \in \mathcal{F}$, it holds that $f_{j_k}(\mathbf{x}_0, \mathbf{x}_{1,j_1}, \ldots, \mathbf{x}_{k-1,j_{k-1}}) = 1$ for every $j_1, \ldots, j_k \in [p]$.

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

$$\mathsf{Adv}^{\mathsf{Ada\text{-}IND}}_{\mathsf{k\text{-}ABE},\mathsf{A}}(1^\lambda) := \left|\Pr[\exp_{\mathsf{k\text{-}ABE},\mathsf{A}}(1^\lambda) = 1 | \beta = 0] - \Pr[\exp_{\mathsf{k\text{-}ABE},\mathsf{A}}(1^\lambda) = 1 | \beta = 1]\right|.$$

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

Definition 4.2 (VerSel-IND security for k-ABE)**.** The definitions for VerSel-IND security for k-ABE is the same as Ada-IND security above except that the adversary A is required to submit the challenge query and key queries to the challenger before it samples the public key.

Comparing with the MIABE Definition in Chapter 3: We note that the definition of kABE in this chapter is equivalent to the one in Chapter 3, except that the encryption algorithm that encrypts the message with an attribute is a public algorithm in this chapter, while it is a secret algorithm in Chapter 3. In both definitions, the message is associated with only a single attribute, which as shown in Chapter 3 is sufficient. In more detail, $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}, \mu)$ above is same as $\mathsf{Enc}_1(\mathsf{msk}, \mathbf{x}, \mu)$ in Chapter 3, except that Enc_1 is a secret algorithm while Enc is a public algorithm. $\mathsf{KeyGen}_i(\mathsf{msk}, \mathbf{x}_i)$ is same as $\mathsf{Enc}_{i+1}(\mathsf{msk}, \mathbf{x}_i)$ in Chapter 3, $\mathsf{KeyGen}_k(\mathsf{msk}, f)$ is same as $\mathsf{KeyGen}(\mathsf{msk}, f)$ in Chapter 3. Further, note that since the encryption algorithm in this chapter is a public algorithm, it suffices to consider that the adversary issues only one challenge query of the form $(\mathbf{x}_0, (\mu_0, \mu_1))$, while it can issue polynomially many key queries for each slot $i \in [k]$ similar to Chapter 3, where the adversary can issue polynomially many key queries and encryption queries for each slot. Finally, note that since the challenge bit β is encoded only in the ciphertext returned by the (public) encryption algorithm, the distinction between the stronger and weaker security notions in Chapter 3 disappears in this chapter. Thus, the security definition given above is same as the stronger security defined in Chapter 3.

4.4.2 Lattice Preliminaries

We use the standard LWE assumption as defined in Chapter 2. We also use the low-norm version of LWE defined as follows:

LWE with Low-Norm Samples. The following lemma states that the LWE problem is hard even when the public matrix is chosen from a low norm Gaussian distribution.

Lemma 4.1 (BLMR13)**.** *Let $k = k(\lambda)$, $m = m(\lambda)$ and $q = q(\lambda) > 2$ be integers. Then if $\mathsf{LWE}(n, m, q, \gamma)$ hardness assumption holds then for any PPT adversary $\mathcal{A}$ we have*

$$|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{x}) \to 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) \to 1]| \leq \mathrm{negl}(\lambda)$$

where $\mathbf{A} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{k\times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^k$, $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}$, $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, $k \geq 6n\log q$ *and* $\sigma = \Omega(\sqrt{n\log q})$.

Trapdoors. We recall the trapdoor sampling algorithms defined in Chapter 3. In this chapter, we use a slightly different notation to be consistent with the notation used in [Wee22].

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 $\mathbf{A}^{-1}(\mathbf{V},\gamma)$ be an output distribution of $\mathcal{D}_{\mathbb{Z},\gamma}^{m\times m'}$ conditioned on $\mathbf{A}\cdot\mathbf{A}^{-1}(\mathbf{V},\gamma) = \mathbf{V}$. 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 denote a γ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\gamma^{-1}$. We also define the special gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ as the matrix obtained by padding $\mathbf{I}_n \otimes (1,2,4,8,\ldots,2^{\lceil\log q\rceil})$ with zero-columns. The following properties had been established in a long sequence of works [GPV08; CHKP10; ABB10a; ABB10b; MP12; BLP+13].

Lemma 4.2 (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* $\tau_0 = \omega(\sqrt{n\log q\log m})$.

Lattice Evaluation. In this chapter, we use the following abstraction of the evaluation procedure in previous LWE based FHE and ABE schemes. We follow the presentation by Tsabary [Tsa19].

Lemma 4.3 (Fully Homomorphic Computation [BGG+14)**.** *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.*
- $\mathsf{EvalFX}(\mathbf{B},F,\mathbf{x}) \to \widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$. *Here,* $\mathbf{x} \in \{0,1\}^\ell$ *is a binary string whose first bit is* 0 *and the second bit is* 1 *and* $F : \{0,1\}^\ell \to \{0,1\}$ *is a circuit with depth* d *that*

ignores the first and the second bit of the input. Then, we have

$$[\mathbf{B} - \mathbf{x} \otimes \mathbf{G}]\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{G} \mod q,$$

where we denote $[x_1\mathbf{G}\| \cdots \|x_k\mathbf{G}]$ *by* $\mathbf{x} \otimes \mathbf{G}$. *Furthermore, we have*

$$\|\mathbf{H}_F\|_\infty \leq m \cdot 2^{O(d)}, \quad \|\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}\|_\infty \leq m \cdot 2^{O(d)}.$$

Finally, we have that the topmost m *rows of* $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ *constitutes an identity matrix.*

- *The running time of* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *is bounded by* $\mathrm{poly}(n, m, \log q, 2^d)$.

The above algorithms are taken from [BGG+14], and are slightly different from those described in Lemma 3.2, which are taken from [GV15]. Please refer to Chapter 3 for the difference between them.

Remark 5. As pointed out in [KNYY20] (See also [BV15b]), we need some entry of $\mathbf{x}$ to be 1 to support arbitrary F. We therefore assume that the second bit of $\mathbf{x}$ is 1. Furthermore, we assume the first bit of $\mathbf{x}$ is 0. This assumption is introduced to make sure that the topmost m rows of $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ constitutes an identity matrix, which is not guaranteed for the evaluation algorithms in [BGG+14]. As we explain below, this can be ensured easily by modifying the evaluation algorithms in [BGG+14]. Suppose that we have EvalF' and EvalFX' without this property. Denoting $\mathbf{x} = (0, \mathbf{x}')$ and $\mathbf{B} = [\mathbf{B}_0\|\mathbf{B}']$, we have

$$[\mathbf{B}' - \mathbf{x}' \otimes \mathbf{G}]\widehat{\mathbf{H}}'_{\mathbf{B}',F',\mathbf{x}'} = \mathbf{B}'\mathbf{H}'_{F'} - F(\mathbf{x})\mathbf{G} \mod q,$$

where F' is the same function as F except that it ignores only the first bit, $\widehat{\mathbf{H}}'_{\mathbf{B}',F',\mathbf{x}'} = \mathsf{EvalFX}(\mathbf{B}', F', \mathbf{x}')$, and $\mathsf{EvalF}(\mathbf{B}', F') \to \mathbf{H}'_{F'}$. We then define the new evaluation algorithms EvalF and EvalFX as $\mathsf{EvalFX}(\mathbf{B}, F, \mathbf{x}) = \widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}} = \begin{bmatrix} \mathbf{I} \\ \widehat{\mathbf{H}}'_{F',\mathbf{x}'} \end{bmatrix}$ and $\mathsf{EvalF}(\mathbf{B}, F) = \mathbf{H}_F = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}'_{F'} \end{bmatrix}$. It is easy to see that the new evaluation algorithms satisfy all the desired properties. In our work, we implicitly assume that $\mathbf{x}$ input to the circuit F always has $0\|1$ as its prefix so that the above lemma holds and will not explicitly write the leading bits for the sake of notational simplicity. In our context, this means that the

first two bits of an attribute $\mathbf{x}$ associated with a ciphertext should be $0\|1$.

Low Norm Variant. We also consider the low norm variant of the lattice evaluation algorithm defined in [Wee22], where $\mathbf{B}$ has low-norm and $\mathbf{G}$ is replaced with $\mathbf{I}$.

Lemma 4.4. *Fix parameters m, ℓ. Given a matrix $\mathbf{B} \in \mathbb{Z}^{m \times m\ell}$ and a circuit $F : \{0,1\}^\ell \to \{0,1\}$ of depth d, we can efficiently compute a matrix $\mathbf{H}_F \in \mathbb{Z}^{m\ell \times m}$ such that $\|\mathbf{H}_F\|_\infty = (\|\mathbf{B}\|_\infty m)^{O(2^d)}$ and for all $\mathbf{x} \in \{0,1\}^\ell$, there exists a matrix $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}} \in \mathbb{Z}^{\ell m \times m}$ with $\|\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}\|_\infty = (\|\mathbf{B}\|_\infty m)^{O(2^d)}$ such that*

$$(\mathbf{B} - \mathbf{x} \otimes \mathbf{I}_m) \cdot \widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}} = \mathbf{B}\mathbf{H}_F - F(\mathbf{x})\mathbf{I}_m$$

Moreover, $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ is efficiently computable given $\mathbf{B}, F, \mathbf{x}$. We use $\mathsf{EvalF}(\mathbf{B}, F)$, $\mathsf{EvalFX}(\mathbf{B}, F, \mathbf{x})$ to denote the algorithms computing $\mathbf{H}_F, \widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ respectively. Finally, the topmost m rows of $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ constitutes an identity matrix.

Remark 6. The condition that the top most m rows of $\widehat{\mathbf{H}}_{\mathbf{B},F,\mathbf{x}}$ constitutes an identity matrix can be satisfied by adding suitable modifications to the evaluation algorithms without this property. See Remark 5 for the detail.

Smudging Lemma. We will also require the standard smudging lemma.

Lemma 4.5 (Smudging Lemma [WWW22)**.** *Let λ be a security parameter. Take any $a \in \mathbb{Z}$ where $|a| \le B$. Suppose $\chi \ge B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ is* $\mathrm{negl}(\lambda)$.

4.4.3 Tensors

In this work, similarly to [Wee22], 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:

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} a_{1,1}\mathbf{B} & \cdots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \cdots & a_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{Z}_q^{ms \times nt}.$$

Throughout the chapter, 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 [Wee22] where matrix multiplication takes precedence over tensor products, i.e. $\mathbf{A} \otimes \mathbf{BC} = \mathbf{A} \otimes (\mathbf{BC})$.

4.5 ASSUMPTIONS AND NEW IMPLICATIONS

In this section, we discuss the evasive and tensor LWE assumptions. Our variants of these assumptions differ slightly from the original formulation by [Wee22] as discussed below.

4.5.1 Evasive LWE

Below, we state a variant of the Evasive-LWE assumption which will be useful for our constructions.

Assumption 4.6 (Evasive LWE)**.** 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. Let Samp be a PPT algorithm 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. For a PPT adversary Adv, we define the following advantage functions:

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

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

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 holds if for every PPT Samp and Adv_1, there exists another PPT Adv_0 and a polynomial $Q(\cdot)$ such that

$$\mathcal{A}^{\mathrm{PRE}}_{\mathsf{Adv}_0}(\lambda) \geq \mathcal{A}^{\mathrm{POST}}_{\mathsf{Adv}_1}(\lambda)/Q(\lambda) - \mathrm{negl}(\lambda).$$

Remark 7. In the above definition, all the entries of $\mathbf{E}'$ are chosen from the same distribution $D_{\mathbb{Z},\chi'}$. However, in our security proof, we often consider the case where some entries of $\mathbf{E}'$ 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 for $\mathbf{E}'$ is implied by the evasive LWE assumption with all entries in $\mathbf{E}'$ 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.

Remark 8. In the above, we chose χ' to be smaller than χ following [VWW23]. This makes the precondition stronger, which in turn makes evasive LWE weaker.

Our assumption is closely related to the evasive LWE assumption that appeared in Wee [Wee22] with minor differences. In Wee, the secret $\mathbf{S}$ is chosen uniformly whereas in our assumption, the secret can be structured and output by the sampler, subject to the pre-condition being true. On the other hand, in [VWW23], $\mathbf{S}$ is the public matrix and can be structured, while $\mathbf{B}$ is secret and is random. An additional difference is related to the auxiliary input. In Wee, aux contains all the coin tosses used by the sampler

– this suffices to rule out obfuscation based counter-examples where aux may contain information of the trapdoor for $\mathbf{P}$ in a hidden way. On the other hand, in [VWW23], the coins of the sampler are private, and aux contains information including certain Gaussian preimages. They argue that their assumption nevertheless avoids the obfuscation based counter-examples, since their auxiliary input does not contain trapdoor for the matrix $\mathbf{P}$. In both their and our cases, aux is derived from the trapdoor for $\mathbf{P}$ or related information that should be kept hidden, but it does not contain the trapdoor itself. We may therefore expect that there is no space for embedding an obfuscation into our auxiliary input, similarly to [VWW23]. We also note that as observed in [VWW23], Tsabary's variant of evasive LWE is less conservative than ours and theirs, since her definition allows aux to depend on $\mathbf{B}$.

In the security proof of our constructions, we sometimes want to include information dependent on $\mathbf{S}$ into the auxiliary information. However, this makes the corresponding evasive LWE assumption stronger and not desirable. The following lemma allows us to do this without strengthening the assumption under certain conditions. 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 if aux_1 is pseudorandom, then we can apply the evasive LWE with respect to a modified sampler whose aux_1 is replaced with a random string to derive the conclusion on postcondition distribution.

Lemma 4.7. *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 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 $\mathsf{Reconstruct}$ *that allows to derive $\mathbf{P}$ from aux_2, i.e. $\mathbf{P} = \mathsf{Reconstruct}(\mathsf{aux}_2)$.*

We introduce the following advantage functions:

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

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

where

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

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

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

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

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

Then, under the Evasive-LWE *(cited above in Assumption 4.6) with respect to* Samp' *that outputs* $(\mathbf{S}, (\mathbf{c}, \mathsf{aux}_2), \mathbf{P})$ *for random* $\mathbf{c}$*, if* $\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda)$ *is negligible for any PPT adversary* Adv*, so is* $\mathcal{A}^{\mathrm{PRE}'}_{\mathsf{Adv}}(\lambda)$ *for any PPT adversary* Adv*.*

Proof. By the assumption, we have $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_1, \mathsf{aux}_2) \approx_c (\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$. This in particular implies $(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathsf{aux}_2) \approx_c (\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathsf{aux}_2)$ since discarding the term making the task of distinguishing the distributions harder. This further implies

$$(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{SP} + \mathbf{E}', \mathbf{c}, \mathsf{aux}_2) \approx_c (\mathbf{B}, \mathbf{C}_0, \mathbf{C}', \mathbf{c}, \mathsf{aux}_2)$$

since adding random term $\mathbf{c}$ chosen independently from the other terms does not make the task of distinguishing the the distributions easier. Applying the evasive LWE with respect to Samp' defined in the statement, we have

$$(\mathbf{B}, \mathbf{SB} + \mathbf{E}, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2) \approx_c (\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2).$$

To complete the proof, it suffices to show

$$(\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathsf{aux}_1, \mathsf{aux}_2) \approx_c (\mathbf{B}, \mathbf{C}_0, \mathbf{K}, \mathbf{c}, \mathsf{aux}_2).$$

To show this, we first observe that the precondition implies $(\mathsf{aux}_1, \mathsf{aux}_2) \approx_c (\mathbf{c}, \mathsf{aux}_2)$, since discarding the terms making the task of distinguishing the distributions harder. We then observe that $(\mathbf{B}, \mathbf{C}_0, \mathbf{K})$ can be sampled publicly given aux_2. This suffices to complete the proof, since having extra terms that can be computed efficiently from the given terms does not make the task of distinguishing the distributions easier. To sample $(\mathbf{B}, \mathbf{C}_0, \mathbf{K})$, we first sample $\mathbf{B}$ with the trapdoor as $(\mathbf{B}, \mathbf{B}_{\tau_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ where $\tau_0 = \omega(\sqrt{n \log q \log m}) \leq O(m \log q)$, compute $\mathbf{P}$ by $\mathbf{P} = \mathsf{Reconstruct}(\mathsf{aux}_2)$, and finally sample $\mathbf{K} \leftarrow \mathbf{B}^{-1}(\mathbf{P}, O(\sqrt{m \log(q)}))$. ■

4.5.2 Tensor LWE

In this section, we define the tensor LWE assumption introduced by Wee [Wee22. Then, we provide new arguments supporting the assumption.

Assumption 4.8 (Tensor LWE)**.** Let $n, m, q, \ell, Q \in \mathbb{N}$ be parameters and $\gamma, \chi > 0$ be Gaussian parameters. For all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, we have

$$\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{A} - \mathbf{x}_i \otimes \mathbf{G}) + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i \in [Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i \in [Q]}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times \ell m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{mn}, \mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}, \mathbf{r}_i^\mathsf{T} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}, \mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

To gain confidence in the tensor LWE assumption, we study conditions under which it can be reduced to standard LWE. To begin, we recall the following lemma which is implicit in [Wee22]. The lemma says that a variant of the tensor LWE assumption holds under the standard LWE assumption if $\mathbf{A}$ matrices are chosen from Gaussian distribution and $\mathbf{G}$ is replaced with $\mathbf{I}$ in certain parameter settings.

Lemma 4.9 (Implicitly proved in [Wee22])**.** *Let $n, m, q, \ell, Q, \beta \in \mathbb{N}$ be parameters and χ_0, χ, and γ be a Gaussian parameter satisfying $m = \Omega(n \log q)$, $\gamma = \lambda^{\omega(1)}$, $\chi = \chi_0 \gamma \lambda^{\omega(1)}$.*

For all $\mathbf{x}_1, \cdots, \mathbf{x}_Q \in \{0,1\}^\ell$, $\mathsf{LWE}(n, Q+m, q, \chi_0)$ *hardness assumption implies*

$$\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{A} - \mathbf{x}_i \otimes \mathbf{I}_m) + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$$

where $\mathbf{A} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{n\times\ell m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{mn}$, $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}$, $\mathbf{r}_i^\mathsf{T} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}$, $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

4.5.3 New Implications for Tensor LWE

We now introduce a new lemma that also proves the same implication between LWE and Tensor LWE in another particular case. Notably, the lemma shows the hardness for the case where $\mathbf{A}$ is chosen uniformly at random rather than from a Gaussian distribution, albeit with the downside of assuming $\mathbf{x}_i = \mathbf{0}$ for all i.

Lemma 4.10 (Tensor LWE with $\{\mathbf{x}_i = \mathbf{0}\}_i$)**.** *Let* $n, m, q, \ell, Q, \beta \in \mathbb{N}$ *be parameters and* χ_0*,* χ*, and* γ *be a Gaussian parameter satisfying* $m = \Omega(n \log q)$*,* $\gamma = \Omega(\sqrt{n \log q})$*, and* $\chi = \gamma\chi_0\lambda^{\omega(1)}$*. Then,* $\mathsf{LWE}(n, m, q, \chi_0)$ *hardness assumption implies*

$$\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times\ell m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{mn}$, $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}$, $\mathbf{r}_i^\mathsf{T} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}$, $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

Proof. Let Adv be an attacker for Tensor-LWE with $\mathbf{x}_i = \mathbf{0}$ for all $i \in [Q]$. Adv is given either $\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$ or $\mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$. We provide a proof to show that under the LWE assumption, Adv has a negligible advantage of distinguishing the left hand side from the right hand side.

G_0 : Adv is given $\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T}\right\}_{i\in[Q]}$.

G_1 : We rewrite $\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i$ using the tensor decomposition of $\mathbf{s} \in \mathbb{Z}_q^{mn}$. In other words,

$$\mathbf{s} = \sum_{j=1}^{m} \mathbf{s}_j \otimes \boldsymbol{\epsilon}_j,$$

where $\boldsymbol{\epsilon}_j$ are the canonical vectors of $\mathbb{Z}_q^m$ and $\mathbf{s}_j \in \mathbb{Z}_q^n$. Let us fix an index $1 \le i \le Q$

and rewrite the i-th sample. We get

$$\begin{aligned}\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i &= \textstyle\sum_{j=1}^{m}(\mathbf{s}_j \otimes \boldsymbol{\epsilon}_j) \cdot (\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{A} + \mathbf{e}_i \\ &= \textstyle\sum_{j=1}^{m}(\mathbf{s}_j \otimes \underbrace{\boldsymbol{\epsilon}_j \mathbf{r}_i^\mathsf{T}}_{:=\mathbf{r}_i[j] \text{ scalar}})\mathbf{A} + \mathbf{e}_i \\ &= \textstyle\sum_{j=1}^{m} \mathbf{r}_i[j] \cdot \mathbf{s}_j \cdot \mathbf{A} + \mathbf{e}_i,\end{aligned}$$

where $\mathbf{r}_i[j]$ is the j-th entry of the vector $\mathbf{r}_i$. Hence, in this game, Adv is given

$$\mathbf{A}, \left\{ \sum_{j=1}^{m} \mathbf{r}_i[j] \cdot \mathbf{s}_j \cdot \mathbf{A} + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T} \right\}_{i \in [Q]}.$$

This is a conceptual change : $\mathsf{G}_1 \equiv \mathsf{G}_2$.

G_2 : We now add some extra noise to the distribution to introduce an LWE instance. Define $\mathbf{e}'_j \leftarrow \mathcal{D}^{\ell m}_{\mathbb{Z},\chi_0}$ for all $j \in [1, m]$. In this game, the attacker is given

$$\mathbf{A}, \left\{ \sum_{j=1}^{m} \mathbf{r}_i[j] \cdot (\mathbf{s}_j \cdot \mathbf{A} + \mathbf{e}'_j) + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T} \right\}_{i \in [Q]}.$$

Note that this game is different from the previous game only in the noise term. In the previous game, the noise is $\mathbf{e}_i$ for the i-th sample, while it is $\mathbf{e}_i + \sum_j \mathbf{r}_i[j] \cdot \mathbf{e}'_j$ in this game. Since we have $\|\sum_j \mathbf{r}_i[j] \cdot \mathbf{e}'_j\|_\infty \leq \mathrm{poly}(\lambda)\gamma\chi_0$ and $\chi = \lambda^{\omega(1)} \cdot \gamma\chi_0$, we can apply Lemma 4.5 to conclude that this only introduces a statistical change: $\mathsf{G}_2 \approx_s \mathsf{G}_1$.

G_3 : In this game, we replace each $(\mathbf{s}_j \cdot \mathbf{A} + \mathbf{e}'_j)$ by a uniform vector $\mathbf{c}'_j \leftarrow \mathbb{Z}_q^{\ell m}$. The attacker Adv thus gets

$$\mathbf{A}, \left\{ \sum_{j=1}^{m} \mathbf{r}_i[j] \cdot \mathbf{c}'_j + \mathbf{e}_i, \mathbf{r}_i^\mathsf{T} \right\}_{i \in [Q]}.$$

This game is computationally indistinguishable from G_2 under the standard LWE assumption: $\mathsf{G}_3 \approx_c \mathsf{G}_2$.

G_4 : Let us define $\mathbf{C}' := \begin{pmatrix} \mathbf{c}'_1 \\ \vdots \\ \mathbf{c}'_m \end{pmatrix}$ and obtain

$$\begin{pmatrix} \sum_{j=1}^{m} \mathbf{r}_1[j] \cdot \mathbf{c}'_j + \mathbf{e}_1 \\ \vdots \\ \sum_{j=1}^{m} \mathbf{r}_Q[j] \cdot \mathbf{c}'_j + \mathbf{e}_Q \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{r}'_1 \\ \vdots \\ \mathbf{r}'_Q \end{pmatrix}}_{\text{public}} \cdot \underbrace{\mathbf{C}'}_{\text{secret}} + \underbrace{\begin{pmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_Q \end{pmatrix}}_{\text{error}}.$$

In this game, we replace $\mathbf{r}'_i\mathbf{C}' + \mathbf{e}_i$ by a uniform random vector $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$. Hence the adversary is given

$$\mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^{\mathsf{T}}\right\}_{i\in[Q]}.$$

This game is computationally indistinguishable from G_3: we use LWE with short public matrix and large secret [BLMR13], which is implied by the standard LWE (See Lemma 4.1). Hence, $\mathsf{G}_4 \approx_c \mathsf{G}_3$.

In the last game, the distribution corresponds to the random case, which allows to conclude the proof. ■

We can introduce a corollary that follows from Lemma 4.10 where $\mathbf{A}$ is replaced by $\mathbf{A} - \mathbf{x} \otimes \mathbf{G}$.

Corollary 4.10.1 (Tensor LWE with the same $\mathbf{x}_i$)**.** *Let $n, m, q, \ell, Q \in \mathbb{N}$, χ_0, χ, and γ be parameters defined as Lemma 4.10. Let $\mathbf{x} \in \{0,1\}^\ell$. Then, $\mathsf{LWE}(n, m, q, \chi_0)$ hardness assumption implies*

$$\mathbf{A}, \left\{\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^{\mathsf{T}})(\mathbf{A} - \mathbf{x} \otimes \mathbf{G} + \mathbf{e}_i, \mathbf{r}_i^{\mathsf{T}}\right\}_{i\in[Q]} \approx_c \mathbf{A}, \left\{\mathbf{c}_i, \mathbf{r}_i^{\mathsf{T}}\right\}_{i\in[Q]}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times\ell m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{mn}$, $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}$, $\mathbf{r}_i^{\mathsf{T}} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}$, $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{\ell m}$.

What prevents Lemma 4.10 to be proved in the general case? The proof of Lemma 4.10 cannot be easily adapted for arbitrary $\mathbf{x}_i$. Following the same proof strategy, we have to prove the pseudorandomness of the following terms:

$$\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i)(\mathbf{A} - \mathbf{x}_i \otimes \mathbf{G}) + \mathbf{e}_i \quad = \sum_j \mathbf{r}_i[j] \cdot (\mathbf{s}_j \cdot (\mathbf{A} - \mathbf{x}_i \otimes \mathbf{G}) + \mathbf{e}'_j) - \sum_j \mathbf{r}_i[j]\mathbf{e}'_j + \mathbf{e}_i.$$

However, it is not possible to replace $\mathbf{s}_j \cdot (\mathbf{A} - \mathbf{x}_i \otimes \mathbf{G}) + \mathbf{e}'_j$ with random vectors as is done in Section 4.5.2, if we are given the term for multiple i with different $\mathbf{x}_i$ and for the same $\mathbf{s}_j$. Thus, the approach cannot be directly transferred.

4.5.4 New Implications from LWE

In this section, we provide new lemmata under the LWE assumption which will be useful for our constructions. We believe these may be of broader applicability.

Lemma 4.11. *Let $n = n(\lambda)$, $m = m(\lambda)$, $N = N(\lambda)$, $q = q(\lambda)$, $\gamma = \gamma(\lambda)$, $\chi_0 = \chi_0(\lambda) \in \lambda^{\omega(1)}$, $\chi = \chi(\lambda)$, and $k = O(1)$ be parameters satisfying $m = \Omega(n \log q)$, $\chi(\lambda) \geq (m\gamma\chi_0)^k$. If $\mathsf{LWE}(n, Q, q, \chi_0)$ holds, then the following distributions are computationally indistinguishable:*

$$\left\{\mathbf{c}_{j_1,\ldots,j_k} := \mathbf{s}(\mathbf{I}_N \otimes \mathbf{r}^{\mathsf{T}}_{1,j_1} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k}) + \mathbf{e}_{j_1,\ldots,j_k}\right\}_{j_1,\ldots,j_k \in [Q]} \approx_c \left\{\mathbf{w}_{j_1,\ldots,j_k}\right\}_{j_1,\ldots,j_k \in [Q]}$$

where $\mathbf{s} \leftarrow \mathbb{Z}_q^{Nm^k}$, $\mathbf{r}_{i,j_i} \leftarrow \mathcal{D}^m_{\mathbb{Z},\gamma}$, $\mathbf{e}_{j_1,\ldots,j_k} \leftarrow \mathcal{D}^N_{\mathbb{Z},\chi}$, $\mathbf{w}_{j_1,\ldots,j_k} \leftarrow \mathbb{Z}_q^N$ for $i \in [k]$ and $j_1, \ldots, j_k \in [Q]$.

Proof. We prove this by induction. The case of $k = 1$ follows from LWE with short public matrices [BLMR13] (Lemma 4.1). Here, we prove the statement for $k = \tau + 1$ assuming it is is true for $k = \tau$. To show the indistinguishability, we start from the distribution on the left hand side and gradually change it to that on the right hand side. We first change the distribution of $\{\mathbf{c}_{j_1,\ldots,j_{\tau+1}}\}_{j_1,\ldots,j_{\tau+1}}$ so that they are sampled as

$$\mathbf{c}_{j_1,\ldots,j_{\tau+1}} = \underbrace{\left(\mathbf{s}(\mathbf{I}_N \otimes \mathbf{I}_m \otimes \mathbf{r}^{\mathsf{T}}_{2,j_2} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{\tau+1,j_{\tau+1}}) + \mathbf{e}'_{j_2,\ldots,j_{\tau+1}}\right)}_{:=\mathbf{s}'_{j_2,\ldots,j_{\tau+1}}} \left(\mathbf{I}_N \otimes \mathbf{r}^{\mathsf{T}}_{1,j_1}\right) + \mathbf{e}_{j_1,\ldots,j_{\tau+1}}.$$

where $\mathbf{e}'_{j_2,\ldots,j_{\tau+1}} \leftarrow \mathcal{D}^{Nm}_{\mathbb{Z},(m\gamma\chi_0)^\tau}$ for $j_2, \ldots, j_{\tau+1} \in [Q]$. We claim that this is statistically indistinguishable from the original distribution. To see this, we observe that

$$\left(\mathbf{s}(\mathbf{I}_N \otimes \mathbf{I}_m \otimes \mathbf{r}^{\mathsf{T}}_{2,j_2} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{\tau+1,j_{\tau+1}}) + \mathbf{e}'_{j_2,\ldots,j_{\tau+1}}\right) \left(\mathbf{I}_N \otimes \mathbf{r}^{\mathsf{T}}_{1,j_1}\right) + \mathbf{e}_{j_1,\ldots,j_{\tau+1}}$$

$$= \mathbf{s}(\mathbf{I}_N \otimes \mathbf{r}_{1,j_1}^\mathsf{T} \otimes \mathbf{r}_{2,j_2}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_{\tau+1,j_{\tau+1}}^\mathsf{T}) + \underbrace{\mathbf{e}'_{j_2,\ldots,j_{\tau+1}} \left(\mathbf{I}_N \otimes \mathbf{r}_{1,j_1}^\mathsf{T}\right) + \mathbf{e}_{j_1,\ldots,j_{\tau+1}}}_{=\text{error}}$$

and these distributions only differ in the error terms. We have

$$\mathbf{e}_{j_1,\ldots,j_{\tau+1}} \approx_s \mathbf{e}'_{j_2,\ldots,j_{\tau+1}} \left(\mathbf{I}_N \otimes \mathbf{r}_{1,j_1}^\mathsf{T}\right) + \mathbf{e}_{j_1,\ldots,j_{\tau+1}}$$

by the smudging lemma, since we have $\chi \geq (m\gamma\chi_0)^{\tau+1}$ and $\|\mathbf{e}'_{j_2,\ldots,j_{\tau+1}} \left(\mathbf{I}_N \otimes \mathbf{r}_{1,j_1}^\mathsf{T}\right)\|_\infty \leq m\gamma \cdot \mathrm{poly}(\lambda) \cdot \|\mathbf{e}'_{j_2,\ldots,j_{\tau+1}}\|_\infty \leq (m\gamma)^{\tau+1} \cdot \chi_0^\tau \cdot \mathrm{poly}(\lambda)\cdot$.

In the next step, we replace each $\mathbf{s}'_{j_2,\ldots,j_{\tau+1}}$ with random vectors. Namely, $\{\mathbf{c}_{j_1,\ldots,j_{\tau+1}}\}_{j_1,\ldots,j_{\tau+1}}$ are sampled as

$$\mathbf{c}_{j_1,\ldots,j_{\tau+1}} = \mathbf{s}'_{j_2,\ldots,j_{\tau+1}} \left(\mathbf{I}_N \otimes \mathbf{r}_{1,j_1}^\mathsf{T}\right) + \mathbf{e}_{j_1,\ldots,j_{\tau+1}},$$

where $\mathbf{s}'_{j_2,\ldots,j_{\tau+1}} \leftarrow \mathbb{Z}_q^{Nm}$. We can see that this change is computationally indistinguishable, by applying the induction hypothesis for each combination of indices $(j_2, \ldots, j_{\tau+1})$. We then use the induction hypothesis for the case of $k = 1$ to replace $\{\mathbf{c}_{j_1,\ldots,j_{\tau+1}}\}_{j_1}$ with random vectors for each combination of $j_2, \ldots, j_{\tau+1}$ one by one. This brings us to the distribution where all $\mathbf{c}_{j_1,\ldots,j_{\tau+1}}$ are random vectors. This completes the proof of the lemma. ■

Lemma 4.12. *Let $n = n(\lambda)$, $m = m(\lambda)$, $N = N(\lambda)$, $q = q(\lambda)$, $\chi = \chi(\lambda)$, and $k = O(1)$ be parameters. If $\mathsf{LWE}(n, (m+1)^k N, q, \chi)$ holds, then, the following distributions are computationally indistinguishable:*

$$\left(\{\mathbf{B}_i\}_{i\in[0,k]}, \mathbf{s}(\mathbf{B}_0 \otimes \mathbf{I}_m^{\otimes k}) + \mathbf{e}_0, \ldots, \mathbf{s}(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(k-i)}) + \mathbf{e}_i, \ldots, \mathbf{s}\mathbf{B}_k + \mathbf{e}_k\right)$$
$$\approx_c \left(\{\mathbf{B}_i\}_{i\in[0,k]}, \mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_k\right)$$

where $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{nm^i \times Nm^i}$, $\mathbf{e}_i \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{m^k N}$, and $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_k \leftarrow \mathbb{Z}_q^{m^k N}$ for $i \in [0, k]$, $\mathbf{s} \leftarrow \mathbb{Z}_q^{nm^k}$.

Proof. We prove the lemma by induction. First, the statement is trivially true when $k = 0$. We then prove that the statement is true for $k = \tau + 1$ assuming it is true for $k = \tau$.

To show this, we first observe that any $\mathbf{x} \in \mathbb{Z}_q^{nm^{\tau+1}}$ can be written as $\mathbf{x} = \sum_{j\in[m]} \mathbf{x}_j \otimes \boldsymbol{\epsilon}_j$ using $\mathbf{x}_j \in \mathbb{Z}_q^{nm^\tau}$ where $\boldsymbol{\epsilon}_j$ is the j-th canonical unit vector of dimension m. We then have

$$\begin{aligned}\mathbf{s}(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau+1-i)}) + \mathbf{e}_i &= \sum_{j\in[m]} (\mathbf{s}_j \otimes \boldsymbol{\epsilon}_j)((\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau-i)}) \otimes \mathbf{I}_m) + \sum_{j\in[m]} \mathbf{e}_{i,j} \otimes \boldsymbol{\epsilon}_j \\ &= \sum_{j\in[m]} \left(\mathbf{s}_j(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau-i)}) + \mathbf{e}_{i,j}\right) \otimes \boldsymbol{\epsilon}_j\end{aligned}$$

for $i \in [0,\tau]$ where we decompose $\mathbf{s}$ and $\mathbf{e}_i$ as $\mathbf{s} = \sum_{j\in[m]} \mathbf{s}_j \otimes \boldsymbol{\epsilon}_j$ and $\mathbf{e}_i = \sum_{j\in[m]} \mathbf{e}_{i,j} \otimes \boldsymbol{\epsilon}_j$. We also have

$$\mathbf{s}\mathbf{B}_{\tau+1} + \mathbf{e}_{\tau+1} = \sum_{j\in[m]} \mathbf{s}_j(\mathbf{I}_{nm^\tau} \otimes \boldsymbol{\epsilon}_j)\mathbf{B}_{\tau+1} + \mathbf{e}_{\tau+1}.$$

Therefore, omitting $\{\mathbf{B}_i\}_{i\in[0,\tau+1]}$, the input to the adversary is

$$\begin{aligned}&\left(\left\{\mathbf{s}(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau+1-i)}) + \mathbf{e}_i\right\}_{i\in[0,\tau]}, \mathbf{s}\mathbf{B}_{\tau+1} + \mathbf{e}_{\tau+1}\right) \\ =\ &\left(\left\{\sum_{j\in[m]} \left(\mathbf{s}_j(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau-i)}) + \mathbf{e}_{i,j}\right) \otimes \boldsymbol{\epsilon}_j\right\}_{i\in[0,\tau]}, \sum_{j\in[m]} \mathbf{s}_j(\mathbf{I}_{nm^\tau} \otimes \boldsymbol{\epsilon}_j)\mathbf{B}_{\tau+1} + \mathbf{e}_{\tau+1}\right) \\ \approx_c\ &\left(\left\{\mathbf{c}_{i,1} \otimes \boldsymbol{\epsilon}_1 + \sum_{j\in[2,m]} \left(\mathbf{s}_j(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau-i)}) + \mathbf{e}_{i,j}\right) \otimes \boldsymbol{\epsilon}_j\right\}_{i\in[0,\tau]},\right. \\ &\left.\mathbf{c}_{\tau+1} + \sum_{j\in[2,m]} \mathbf{s}_j(\mathbf{I}_{nm^\tau} \otimes \boldsymbol{\epsilon}_j)\mathbf{B}_{\tau+1}\right) \\ \equiv\ &\left(\left\{\mathbf{c}_{i,1} \otimes \boldsymbol{\epsilon}_1 + \sum_{j\in[2,m]} \left(\mathbf{s}_j(\mathbf{B}_i \otimes \mathbf{I}_m^{\otimes(\tau-i)}) + \mathbf{e}_{i,j}\right) \otimes \boldsymbol{\epsilon}_j\right\}_{i\in[0,\tau]}, \mathbf{c}_{\tau+1}\right) \\ \approx_c\ &\left(\left\{\mathbf{c}_{i,1} \otimes \boldsymbol{\epsilon}_1 + \sum_{j\in[2,m]} \mathbf{c}_{i,j} \otimes \boldsymbol{\epsilon}_j\right\}_{i\in[0,\tau]}, \mathbf{c}_{\tau+1}\right) \\ \equiv\ &\left(\{\mathbf{c}_i\}_{i\in[0,\tau]}, \mathbf{c}_{\tau+1}\right)\end{aligned}$$

where $\mathbf{c}_i \leftarrow \mathbb{Z}_q^{m^{\tau+1}N}$ and $\mathbf{c}_{i,j} \leftarrow \mathbb{Z}_q^{m^\tau N}$. In the third line, we used the induction hypothesis

for secret $\mathbf{s}' := \mathbf{s}_1$ and matrices

$$\mathbf{B}'_i := \begin{cases} \mathbf{B}_i \in \mathbb{Z}_q^{n \times Nm^i} & \text{if } i \in [0, \tau - 1] \\ (\mathbf{B}_\tau \| (\mathbf{I}_{nm^\tau} \otimes \boldsymbol{\epsilon}_j)\mathbf{B}_{\tau+1}) \in \mathbb{Z}_q^{n \times N(m+1)m^\tau} & \text{if } i = \tau \end{cases}$$

and the parameter $N' = (m+1)N$. Note that the number of columns in each $\mathbf{B}'_i$ is at most $N'm^i$ and thus the indistinguishability follows from the induction hypothesis and the assumption $\mathsf{LWE}(n, (m+1)^{\tau+1}N, q, \chi)$. The indistinguishability of the fifth line also holds from the induction hypothesis similarly to the third line. Here, we apply the induction hypothesis for each $j \in [2, m]$ one by one, by setting secret $\mathbf{s}' := \mathbf{s}_j$ and matrices $\mathbf{B}'_i = \mathbf{B}_i$ for all $i \in [0, \tau]$. This completes the proof of the lemma.

4.6 TWO-INPUT ABE FROM EVASIVE AND TENSOR LWE

4.6.1 Construction

In this section, we define our construction of 2ABE for P using evasive LWE (Assumption 4.6) and tensor LWE (Assumption 4.8). As discussed in Section 4.1, when restricted to NC_1, our construction can be modified to rely only on evasive LWE. We defer the details of this modification to Section 4.7 and focus on circuit class P for this section.

Let ℓ be the length of the attribute in each slot. The construction supports general circuits with bounded depth d and the decryption is possible when $f(\mathbf{x}_0 \| \mathbf{x}_1) = 0$, where $\mathbf{x}_0$ is the attribute associated with a ciphertext, $\mathbf{x}_1$ is the attribute associated with the first slot key, and f is the function associated with the second slot key. Below $\mathbf{I}$ refers to $\mathbf{I}_m$.

$\mathsf{Setup}(1^\lambda)$**:** The setup algorithm takes as input the security parameter λ and does the following:

- Sample $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2 \leftarrow \mathbb{Z}_q^{n \times m\ell}$; $(\mathbf{B}, \mathbf{B}_{\tau_B}^{-1}) \leftarrow \mathsf{TrapGen}(1^\lambda, 2nm + nm^2, (2nm + nm^2)w)$; $(\mathbf{C}, \mathbf{C}_{\tau_C}^{-1}) \leftarrow \mathsf{TrapGen}(1^\lambda, nm, nmw)$, where $w \in O(\log q)$; $\mathbf{D} \leftarrow \mathbb{Z}_q^{n \times m}$.

- Output $\mathsf{mpk} = (\mathbf{A}_0, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, where $\mathbf{A} = (\mathbf{A}_1 \| \mathbf{A}_2)$, $\mathsf{msk} = (\mathbf{B}_{\tau_B}^{-1}, \mathbf{C}_{\tau_C}^{-1})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0, \mu)$**:** The encryption algorithm takes as input the master public key mpk, an attribute $\mathbf{x}_0$ and message bit $\mu \in \{0, 1\}$ and does the following:

- If $\mu = 1$, sample $\mathbf{c}_1 \leftarrow \mathbb{Z}_q^{m^2\ell}$, $\mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(2nm+nm^2)w}$.
- Else,
  - Sample $\mathbf{s}, \mathbf{s}_0 \leftarrow \mathbb{Z}_q^{nm}$ and $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^{nm^2}$.
  - Sample error vectors $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_1}^{m^2\ell}$, $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_2}^{(2nm+nm^2)w}$.
  - Compute $\mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0) \begin{pmatrix} (\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I} \end{pmatrix} + \mathbf{e}_1$.
  - Compute $\mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1)\mathbf{B} + \mathbf{e}_2$.
- Output $\mathsf{ct}_{\mathbf{x}_0} = (\mathbf{c}_1, \mathbf{c}_2)$.

$\mathsf{KeyGen}_1(\mathsf{msk}, \mathbf{x}_1)$**:** The keygen algorithm for slot 1 takes as input the master secret key msk and the slot attribute $\mathbf{x}_1 \in \{0, 1\}^\ell$ and does the following:

- Sample $\mathbf{r} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^m$.
- Sample $\mathbf{L}_{\mathbf{x}_1} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix} (\mathbf{A}_2 - \mathbf{x}_1 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T} & & \\ & \mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T} & \\ & & \mathbf{C} \otimes \mathbf{r}^\mathsf{T} \end{pmatrix}, \tau_B\right)$.
- Output $\mathsf{sk}_{1,\mathbf{x}_1} = \left(\mathbf{r}, \mathbf{L}_{\mathbf{x}_1}\right)$.

$\mathsf{KeyGen}_2(\mathsf{msk}, f)$ The keygen algorithm for slot 2 takes as input the master secret key msk and slot function f, which is a function represented as a binary circuit $f : \{0, 1\}^{2\ell} \to \{0, 1\}$ and does the following:

- Sample $\mathbf{t} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^m$, $\mathbf{U} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m\times m}$.
- Compute $\mathbf{H}_f = \mathsf{EvalF}(\mathbf{A}, f)$ and $\mathbf{A}_f = \mathbf{A}\mathbf{H}_f$.

- Sample $\mathbf{M}_f \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix} \mathbf{A}_f\mathbf{U} \otimes \mathbf{I} \\ \mathbf{0}_{nm \times m^2} \\ \mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{I} \end{pmatrix}, \tau_B\right)$ and $\mathbf{N}_f \leftarrow \mathbf{C}^{-1}\left((\mathbf{D} \otimes \mathbf{t}^\mathsf{T}), \tau_C\right)$.
- Output $\mathsf{sk}_{2,f} = \left(\mathbf{t}, \mathbf{U}, \mathbf{M}_f, \mathbf{N}_f\right)$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}_{\mathbf{x}_0}, \mathsf{sk}_{1,\mathbf{x}_1}, \mathsf{sk}_{2,f})$ The decryption algorithm takes as input the ciphertext $\mathsf{ct}_{\mathbf{x}_0}$, key $\mathsf{sk}_{1,\mathbf{x}_1}$ for slot 1, and key $\mathsf{sk}_{2,f}$ for slot 2 and does the following:

- Parse $\mathsf{ct}_{\mathbf{x}_0}$ as $(\mathbf{c}_1, \mathbf{c}_2)$, $\mathsf{sk}_{1,\mathbf{x}_1}$ as $(\mathbf{r}, \mathbf{L}_{\mathbf{x}_1})$ and $\mathsf{sk}_{2,f}$ as $(\mathbf{t}, \mathbf{U}, \mathbf{M}_f, \mathbf{N}_f)$.
- Compute $\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)} = \mathsf{EvalFX}(\mathbf{A}, f, (\mathbf{x}_0\|\mathbf{x}_1))$.
- Compute the following:
  $\mathbf{d}_1 = \mathbf{c}_1, \quad (\mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4) = \mathbf{c}_2\mathbf{L}_{\mathbf{x}_1}, \quad \mathbf{d}_5 = \mathbf{c}_2\mathbf{M}_f,$
  $\mathbf{d}_6 = \mathbf{N}_f, \quad \mathbf{d}'_1 = \mathbf{d}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d}_3, \quad \mathbf{d}'_5 = \mathbf{d}_5(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d}_4\mathbf{d}_6,$
  $\mathbf{d}_7 = (\mathbf{d}'_1\|\mathbf{d}_2)\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}, \quad \mathbf{d}_8 = \mathbf{d}_7 - \mathbf{d}'_5.$
  Note that $\mathbf{d}_6$ is a matrix of size $nmw \times m$ and $\mathbf{d}_i$ for all $i \neq 6$ are vectors.
- If $\|\mathbf{d}_8\|_\infty \leq \beta_0$ (where β_0 is as defined in the Sec. 4.6.1) then output $\mu' = 0$, else output 1.

Correctness. Here, we show correctness of the scheme.

When $\mu = 1$: We first show the correctness for the case of $\mu = 1$. For an honest run of the protocol, $\mathbf{d}_1$ is distributed uniformly at random over its domain. Then, since $\mathbf{r} \neq \mathbf{0}$ with overwhelming probability and thus $\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}$ is a full-rank matrix, $\mathbf{d}'_1$ is distributed uniformly at random over its domain. Then, since the topmost m rows of $\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}$ constitutes an identity matrix by Lemma 3.2, $(\mathbf{d}'_1\|\mathbf{d}_2)\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}$ is distributed uniformly at random over its domain. Finally, since each column of $\mathbf{U}$ is chosen from $\mathcal{D}^m_{\mathbb{Z},\gamma}$, with overwhelming probability, there exists $i \in [m]$ such that the i-th column of $\mathbf{U}$ is not a zero vector. This in turn implies that that the i-th entry of $\mathbf{d}_7$ is distributed uniformly at random over $\mathbb{Z}_q$. Since we set $\beta_0/q = \lambda^{-\omega(1)}$, the probability that the decryption algorithm falsely outputs 0 is negligible as desired.

When $\mu = 0$: Next, we show the correctness for the case of $\mu = 0$. For an honest run of the protocol, we have

- $\mathbf{d}_1 = \mathbf{c}_1 = \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1$.

  Let $(\mathbf{e}'_2,\ \mathbf{e}'_3,\ \mathbf{e}'_4) = \mathbf{e}_2 \cdot \mathbf{L}_{\mathbf{x}_1}$

- $\mathbf{d}_2 = \mathbf{s}((\mathbf{A}_2 - \mathbf{x}_1 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_2$,

- $\mathbf{d}_3 = \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_3$,

- $\mathbf{d}_4 = \mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_4$,

- $\mathbf{d}_5 = \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{e}'_5$, where $\mathbf{e}'_5 = \mathbf{e}_2 \cdot \mathbf{M}_f$

- $\mathbf{d}'_1$ is computed as

$$\begin{aligned}
\mathbf{d}'_1 &= \mathbf{d}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d}_3 \\
&= (\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1)(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) - \mathbf{e}'_3 \\
&= \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) - \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}''_1 \\
&= \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}''_1
\end{aligned}$$

  Here $\mathbf{e}''_1 = \mathbf{e}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{e}'_3$

- $\mathbf{d}'_5$ is computed as

$$\begin{aligned}
\mathbf{d}'_5 &= \mathbf{d}_5(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{d}_4\mathbf{d}_6 \\
&= (\mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{I}) + \mathbf{e}'_5)(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - (\mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_4)\mathbf{N}_f \\
&= \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}''_5, \\
&= \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}''_5
\end{aligned}$$

  where we use $(\mathbf{C} \otimes \mathbf{r}^\mathsf{T})\mathbf{N}_f = \mathbf{C}\mathbf{N}_f \otimes \mathbf{r}^\mathsf{T} = \mathbf{D} \otimes \mathbf{t}^\mathsf{T} \otimes \mathbf{r}^\mathsf{T}$ and define $\mathbf{e}''_5 := \mathbf{e}'_5(\mathbf{I} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{e}'_4\mathbf{N}_f$ on the third line.

- $\mathbf{d}_7$ is computed as

$$\begin{aligned}
\mathbf{d}_7 &= (\mathbf{d}'_1 \| \mathbf{d}_2) \cdot (\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}) \\
&= ((\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}''_1) \| (\mathbf{s}((\mathbf{A}_2 - \mathbf{x}_1 \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_2)) \cdot (\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}) \\
&= (\mathbf{s}((\mathbf{A}_1 \| \mathbf{A}_2 - (\mathbf{x}_0 \| \mathbf{x}_1) \otimes \mathbf{G}) \otimes \mathbf{r}^\mathsf{T}) + (\mathbf{e}''_1 \| \mathbf{e}'_2)) \cdot (\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}) \\
&= \mathbf{s}((\mathbf{A}_f - f(\mathbf{x}_0\|\mathbf{x}_1)\mathbf{G}) \otimes \mathbf{r}^\mathsf{T})\mathbf{U} + \mathbf{e}'_7 \\
&= \mathbf{s}((\mathbf{A}_f - f(\mathbf{x}_0\|\mathbf{x}_1)\mathbf{G})\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_7 \\
&= \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_7 \quad \text{if } f(\mathbf{x}_0\|\mathbf{x}_1) = 0
\end{aligned}$$

where we define $\mathbf{e}'_7 := (\mathbf{e}''_1 \| \mathbf{e}'_2)\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}$ on the fourth line.

- $\mathbf{d}_8 = \mathbf{d}_7 - \mathbf{d}'_5 = \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) + \mathbf{e}'_7 - \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}^\mathsf{T}) - \mathbf{e}''_5 = \mathbf{e}'_7 - \mathbf{e}''_5$ which is small ($\leq \beta_0$).

Therefore, the decryption algorithm outputs 0 as desired.

Error Bound: The error term is bounded as follows. Let β_0 denote the error bound.

$$\begin{aligned}
\|\mathbf{e}'_7\|_\infty + \|\mathbf{e}''_5\|_\infty &= \|(\mathbf{e}''_1\|\mathbf{e}'_2)\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}\|_\infty + \|\mathbf{e}'_5(\mathbf{I}\otimes\mathbf{r}^\mathsf{T}) - \mathbf{e}'_4\mathbf{N}_f\|_\infty \\
&= \|(\mathbf{e}_1(\mathbf{I}_{m\ell}\otimes\mathbf{r}^\mathsf{T}) - \mathbf{e}'_3\|\mathbf{e}'_2)\widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0\|\mathbf{x}_1)}\mathbf{U}\|_\infty + \|\mathbf{e}'_5(\mathbf{I}\otimes\mathbf{r}^\mathsf{T}) - \mathbf{e}'_4\mathbf{N}_f\|_\infty \\
&\leq ((\chi_1\gamma + \chi_2\tau_B)\beta\gamma + \chi_2\tau_B\gamma + \chi_2\tau_B\tau_C)\,\mathrm{poly}(m) \\
&\quad \text{since } (\mathbf{e}'_2,\ \mathbf{e}'_3,\ \mathbf{e}'_4) = \mathbf{e}_2\mathbf{L}_{\mathbf{x}_1} \text{ and } \mathbf{e}'_5 = \mathbf{e}_2\mathbf{M}_f \\
&\leq \beta_0.
\end{aligned}$$

Parameters. We set the parameters as follows.

$$n = \mathrm{poly}(\lambda, d), \qquad m = O(n\log q), \qquad \tau_B = O(\sqrt{(2nm + nm^2)\log q}),$$
$$\tau_C = O(\sqrt{nm\log q}), \qquad \beta = (2m)^d, \qquad \gamma = \chi_1 = \lambda^{\omega(1)},$$
$$\chi_3 = \chi_4 = \chi_6 = \gamma\chi_1\lambda^{\omega(1)}, \qquad \chi_7 = \chi_3\chi_4\beta\gamma\lambda^{\omega(1)}, \qquad \chi_5 = \lambda^{\omega(1)}\chi_7,$$
$$\chi_2 = \chi_5\lambda^{\omega(1)}, \qquad \beta_0 = \beta\gamma^2\tau_B\tau_C\chi_1\chi_2\lambda^{\omega(1)}, \qquad q = \beta_0\lambda^{\omega(1)}.$$

In the above, χ_3, χ_4, χ_5, χ_6, and χ_7 are the parameters that only appear in the security proof.

4.6.2 Security

Here, we prove the following theorem, which asserts the security of our scheme.

Theorem 4.13. *Assuming evasive* LWE *(Assumption 4.6), tensor* LWE *(Assumption 4.8), and* LWE*, our construction for 2-input* ABE *for* P *satisfies very selective security (Definition 4.2). Moreover, for the restricted class* NC_1*, our construction for 2-input* ABE *relies only on evasive* LWE*.*

Proof. To prove the security, we need to prove the indistinguishability of the following

two distributions. Let Q_c and Q_s be the number of slot 1 and slot 2 key queries, respectively. In the following, for simplicity, we let $Q_c = Q_s = Q$, which can be assumed without loss of generality.

Distribution D_0:

$$\left(\mathsf{mpk}, \quad \mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0)\begin{pmatrix}(\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I}\end{pmatrix} + \mathbf{e}_1, \quad \mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1)\mathbf{B} + \mathbf{e}_2 \right.$$
$$\left. \left\{\mathsf{sk}_{1,\mathbf{x}_{1,i}} = \left(\mathbf{r}_i, \mathbf{L}_{\mathbf{x}_{1,i}}\right)\right\}_{i\in[Q]}, \quad \left\{\mathsf{sk}_{2,f_j} = \left(\mathbf{t}_j, \mathbf{U}_j, \mathbf{M}_{f_j}, \mathbf{N}_{f_j}\right)\right\}_{j\in[Q]} \right)$$

Distribution D_1:

$$\left(\mathsf{mpk}, \quad \mathbf{c}_1 \leftarrow \mathbb{Z}_q^{m^2\ell}, \quad \mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(2nm+nm^2)w} \right.$$
$$\left. \left\{\mathsf{sk}_{1,\mathbf{x}_{1,i}} = \left(\mathbf{r}_i, \mathbf{L}_{\mathbf{x}_{1,i}}\right)\right\}_{i\in[Q]}, \quad \left\{\mathsf{sk}_{2,f_j} = \left(\mathbf{t}_j, \mathbf{U}_j, \mathbf{M}_{f_j}, \mathbf{N}_{f_j}\right)\right\}_{j\in[Q]} \right)$$

where $\mathbf{x}_0$ is the attribute for the encryption, $\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,Q}$ are the key queries for slot 1, $f_1, \ldots, f_Q$ are key queries for slot 2, and $\mathsf{sk}_{1,\mathbf{x}_{1,i}}$ (resp., sk_{2,f_j}) is secret key for $\mathbf{x}_{1,i}$ (resp., f_j) for slot 1 (resp., slot 2). In particular, we have

$$\mathbf{L}_{\mathbf{x}_{1,i}} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix}(\mathbf{A}_2 - \mathbf{x}_{1,i} \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T} & & \\ & \mathbf{A}_0 \otimes \mathbf{r}_i^\mathsf{T} & \\ & & \mathbf{C} \otimes \mathbf{r}_i^\mathsf{T}\end{pmatrix}, \tau_B\right)$$

and

$$\mathbf{M}_{f_j} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix}\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I} \\ \mathbf{0}_{nm\times m^2} \\ \mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{I}\end{pmatrix}, \tau_B\right), \quad \mathbf{N}_{f_j} \leftarrow \mathbf{C}^{-1}\left((\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T}), \tau_C\right).$$

We note that we have $f_j(\mathbf{x}_0 \| \mathbf{x}_{1,i}) = 1$ for all $i, j \in [Q]$ by the definition of the security game. We can see that D_0 and D_1 are the views of the adversary when $\mu = 0$ and $\mu = 1$ are encrypted, respectively. We then apply our variant of evasive LWE (Lemma 4.7)

assumption for matrix $\mathbf{B}$ with the sampler Samp that outputs $(\mathbf{S}, \mathbf{P}, \mathsf{aux} = (\mathsf{aux}_1, \mathsf{aux}_2))$ defined as follows:[5]

$$
\begin{aligned}
\mathbf{S} &= (\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1) \\
\mathsf{aux}_1 &= \mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0) \begin{pmatrix} (\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I} \end{pmatrix} + \mathbf{e}_1, \\
\mathsf{aux}_2 &= (\mathbf{x}_0, \mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,Q}, f_1, \ldots, f_Q, \mathsf{coins}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_Q, \mathbf{t}_1, \ldots, \mathbf{t}_Q, \mathbf{U}_1, \ldots, \mathbf{U}_Q, \\
&\quad \mathbf{N}_{f_1}, \ldots, \mathbf{N}_{f_Q}, \mathbf{A}_0, \mathbf{A}, \mathbf{C}, \mathbf{D}) \\
\mathbf{P}_0 &= \left((\mathbf{A}_2 - \mathbf{x}_{1,1} \otimes \mathbf{G}) \otimes \mathbf{r}_1^\top \| \cdots \| (\mathbf{A}_2 - \mathbf{x}_{1,Q} \otimes \mathbf{G}) \otimes \mathbf{r}_Q^\top \right) \\
\mathbf{P}_1 &= (\mathbf{A}_0 \otimes \mathbf{r}_1^\top \| \cdots \| \mathbf{A}_0 \otimes \mathbf{r}_Q^\top) \\
\mathbf{P}_2 &= \left(\mathbf{C} \otimes \mathbf{r}_1^\top \| \cdots \| \mathbf{C} \otimes \mathbf{r}_Q^\top \right) \\
\mathbf{P}_3 &= \left(\begin{pmatrix} \mathbf{A}_{f_1} \mathbf{U}_1 \otimes \mathbf{I} \\ \mathbf{0}_{nm \times m^2} \\ \mathbf{D} \otimes \mathbf{t}_1^\top \otimes \mathbf{I} \end{pmatrix} \middle\| \cdots \middle\| \begin{pmatrix} \mathbf{A}_{f_Q} \mathbf{U}_Q \otimes \mathbf{I} \\ \mathbf{0}_{nm \times m^2} \\ \mathbf{D} \otimes \mathbf{t}_Q^\top \otimes \mathbf{I} \end{pmatrix} \right) \\
\mathbf{P} &= \left(\begin{pmatrix} \mathbf{P}_0 & & \\ & \mathbf{P}_1 & \\ & & \mathbf{P}_2 \end{pmatrix} \middle\| \mathbf{P}_3 \right)
\end{aligned}
$$

where $\mathsf{coins}_{\mathcal{A}}$ is adversary's coin. By Lemma 4.7, to prove that D_0 and D_1 are computationally indistinguishable, it suffices to show the computational indistinguishability of the following distributions:

Distribution D_0':

[5] By Lemma 4.7, it suffices to invoke the evasive LWE for a modified sampler that outputs random aux_1, instead of aux_1 that is dependent on $(\mathbf{s}, \mathbf{s}_0)$. The same comments apply to other invocations of the assumption.

$$\left(\begin{array}{ccc} \mathbf{c}_1 = (\mathbf{s}, \mathbf{s}_0)\begin{pmatrix}(\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I} \\ \mathbf{A}_0 \otimes \mathbf{I}\end{pmatrix} + \mathbf{e}_1, & \mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1)\mathbf{B} + \mathbf{e}_2, & \mathbf{B} \\ \left\{\mathbf{c}_{3,i}, \mathbf{c}_{4,i}, \mathbf{c}_{5,i}\right\}_{i\in[Q]}, & \left\{\mathbf{c}_{6,j}\right\}_{j\in[Q]}, & \mathsf{aux}' \end{array}\right)$$

Distribution D_1':

$$\left(\begin{array}{ccc} \mathbf{w}_1, & \mathbf{w}_2, & \mathbf{B} \\ \left\{\mathbf{w}_{3,i}, \mathbf{w}_{4,i}, \mathbf{w}_{5,i}\right\}_{i\in[Q]}, & \left\{\mathbf{w}_{6,j}\right\}_{j\in[Q]}, & \mathsf{aux}' \end{array}\right)$$

In the above distributions,

$$(\mathbf{c}_{3,i},\ \mathbf{c}_{4,i},\ \mathbf{c}_{5,i},\ \mathbf{c}_{6,j}) = (\mathbf{s}\ , \mathbf{s}_0,\ \mathbf{s}_1)\left(\begin{array}{ccc} (\mathbf{A}_2 - \mathbf{x}_{1,i} \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T} & & \\ & \mathbf{A}_0 \otimes \mathbf{r}_i^\mathsf{T} & \\ & & \mathbf{C} \otimes \mathbf{r}_i^\mathsf{T} \end{array}\right)\left\|\left(\begin{array}{c} \mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I} \\ \mathbf{0}_{nm\times m^2} \\ \mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{I} \end{array}\right)\right. + (\mathbf{e}_{3,i},\ \mathbf{e}_{4,i},\ \mathbf{e}_{5,i},\ \mathbf{e}_{6,j})$$

where aux' above is defined as aux_2 in distribution D_0, $\mathbf{e}_{3,i} \leftarrow \mathcal{D}^{m\ell}_{\mathbb{Z},\chi_3}$, $\mathbf{e}_{4,i} \leftarrow \mathcal{D}^{m\ell}_{\mathbb{Z},\chi_4}$, $\mathbf{e}_{5,i} \leftarrow \mathcal{D}^{nmw}_{\mathbb{Z},\chi_5}$, and $\mathbf{e}_{6,j} \leftarrow \mathcal{D}^{m^2}_{\mathbb{Z},\chi_6}$ and all the $\mathbf{w}$ vectors are of the same dimension as the corresponding $\mathbf{c}$ vector and chosen randomly from their respective domains. Note that we set $\chi_2 > \chi_3, \chi_4, \chi_5, \chi_6$ so that we can rely on quantitatively weaker evasive LWE assumption (See Remark 8). We also note that here, we have $\chi_5 \neq \chi_3 = \chi_4 = \chi_6$, where Gaussian distributions with different standard deviations are mixed. We refer to Remark 7 for details. We have

$\mathbf{c}_{3,i} = \mathbf{s}((\mathbf{A}_2 - \mathbf{x}_{1,i} \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{3,i}$

$\mathbf{c}_{4,i} = \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{4,i}$

$\mathbf{c}_{5,i} = \mathbf{s}_1(\mathbf{C} \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{5,i}$ which can be rewritten as $\mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_i^\mathsf{T})\mathbf{C} + \mathbf{e}_{5,i}$

$\mathbf{c}_{6,j} = \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{I}) + \mathbf{e}_{6,j}$.

We then apply the evasive LWE assumption once again, now for matrix $\mathbf{C}$ with sampler Samp' that outputs $(\mathbf{S}', \mathbf{P}', \mathsf{aux}' = (\mathsf{aux}'_1, \mathsf{aux}'_2))$ defined as follows:

$$
\begin{aligned}
\mathbf{S}' &= \begin{pmatrix} \mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_1^\mathsf{T}) \\ \vdots \\ \mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_Q^\mathsf{T}) \end{pmatrix} \\
\mathsf{aux}'_1 &= (\mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_{3,i}, \mathbf{c}_{4,i}\}_{i\in[Q]}, \{\mathbf{c}_{6,j}\}_{j\in[Q]}) \\
\mathsf{aux}'_2 &= (\mathbf{x}_0, \mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,Q}, f_1, \ldots, f_Q, \mathsf{coins}_{\mathcal{A}}, \mathbf{r}_1, \ldots, \mathbf{r}_Q, \mathbf{t}_1, \ldots, \mathbf{t}_Q, \mathbf{U}_1, \ldots, \mathbf{U}_Q, \\
&\quad \mathbf{A}_0, \mathbf{A}, \mathbf{B}, \mathbf{D}) \\
\mathbf{P}' &= (\mathbf{D} \otimes \mathbf{t}_1^\mathsf{T} \| \cdots \| \mathbf{D} \otimes \mathbf{t}_Q^\mathsf{T})
\end{aligned}
$$

where $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_{3,i}$, $\mathbf{c}_{4,i}$, and $\mathbf{c}_{6,j}$ are chosen as in distribution D'_0. By Lemma 4.7, it suffices to prove the computational indistinguishability of the following distributions:

Distribution D''_0:

$$
\begin{pmatrix} \mathbf{c}_1, & \mathbf{c}_2, & \mathbf{C}, \\ \{\mathbf{c}_{3,i}, \mathbf{c}_{4,i}, \mathbf{c}_{5,i}\}_{i\in[Q]}, & \{\mathbf{c}_{6,j}\}_{j\in[Q]}, & \{\mathbf{c}_{7,i,j}\}_{i,j\in[Q]}, \quad \mathsf{aux}'' \end{pmatrix}
$$

Distribution D''_1:

$$
\begin{pmatrix} \mathbf{w}_1, & \mathbf{w}_2, & \mathbf{C}, \\ \{\mathbf{w}_{3,i}, \mathbf{w}_{4,i}, \mathbf{w}_{5,i}\}_{i\in[Q]}, & \{\mathbf{w}_{6,j}\}_{j\in[Q]}, & \{\mathbf{w}_{7,i,j}\}_{i,j\in[Q]}, \quad \mathsf{aux}'' \end{pmatrix}
$$

where aux'' is defined as aux'_2 in distribution D'_0,

$$\mathbf{c}_{7,i,j} = \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j}, \qquad \text{where } \mathbf{e}_{7,i,j} \leftarrow D^m_{\mathbb{Z},\chi_7}$$

and $\mathbf{w}_{7,i,j}$ is a random vector with the same dimension as $\mathbf{c}_{7,i,j}$. Note that we set $\chi_5 > \chi_7$ so that we can rely on quantitatively weaker evasive LWE assumption (See Remark 8). The rest of the vectors are defined as in distribution D'_0 and D'_1. From the above discussion, it suffices to prove Lemma 4.14 in the following to complete the proof of Theorem 4.13. ■

Lemma 4.14. *Distributions D''_0 and D''_1 are computationally indistinguishable under the hardness assumption of* LWE *and tensor* LWE.

Proof. We prove the lemma via the following hybrids.

G_0 : This is same as D''_0.

G_1 : In this hybrid, the challenger computes $\mathbf{c}_{4,i}$ as

$$\mathbf{c}_{4,i} = \mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\mathsf{T}) - \underbrace{\left(\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{4,i}\right)}_{:=\mathbf{c}'_{4,i}}.$$

G_2 : In this hybrid, the challenger samples $\mathbf{c}_1$ and $\mathbf{c}_2$ randomly as $\mathbf{c}_1 \leftarrow \mathbb{Z}_q^{m^2\ell}$, $\mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(2nm+nm^2)w}$.

G_3: In this hybrid, $\mathbf{c}_{7,i,j}$ is computed as $\mathbf{c}_{7,i,j} = \mathbf{c}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T}) - \underbrace{\left(\mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j}\right)}_{:=\mathbf{c}'_{7,i,j}}$.

G_4 : In this hybrid, $\mathbf{c}_{5,i}$ and $\mathbf{c}_{6,j}$ are chosen randomly as $\mathbf{c}_{5,i} \leftarrow \mathbb{Z}_q^{nmw}$ and $\mathbf{c}_{6,j} \leftarrow \mathbb{Z}_q^{m^2}$.

G_5: In this hybrid, $\mathbf{c}'_{7,i,j}$ is computed differently as

$$\mathbf{c}'_{7,i,j} = [\mathbf{c}'_{4,i} \| \mathbf{c}_{3,i}] \widehat{\mathbf{H}}_{\mathbf{A},f,(\mathbf{x}_0 \| \mathbf{x}_{1,i})} \mathbf{U}_j + \underbrace{\mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\intercal) + \mathbf{e}_{7,i,j}}_{\mathbf{c}''_{7,i,j}}.$$

G_6 : In this hybrid, $\mathbf{c}_{3,i} \leftarrow \mathbb{Z}_q^{m\ell}$, $\mathbf{c}'_{4,i} \leftarrow \mathbb{Z}_q^{m\ell}$ and $\mathbf{c}''_{7,i,j} \leftarrow \mathbb{Z}_q^{m}$.

G_7 : In this hybrid, $\mathbf{c}_{4,i} \leftarrow \mathbb{Z}_q^{m\ell}$, $\mathbf{c}_{7,i,j} \leftarrow \mathbb{Z}_q^{m}$.
It is easy to see that the distribution in G_7 is the same as that of D''_1.

Indistinguishability of hybrids:

We prove the indistinguishability between the hybrid distributions via the following claims.

Claim 4.15. $\mathsf{G}_0 \approx_s \mathsf{G}_1$.

Proof. The two hybrids differ only in the error term in $\mathbf{c}_{4,i}$ and are indistinguishable due to the smudging lemma.

In G_0:

$$\mathbf{c}_{4,i} = \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}_i^\intercal) + \mathbf{e}_{4,i}$$

In G_1:

$$\begin{aligned}
\mathbf{c}_{4,i} &= \mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\intercal) - \left(\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\intercal) + \mathbf{e}_{4,i}\right) \\
&= (\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1)(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\intercal) - \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\intercal) - \mathbf{e}_{4,i} \\
&= \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\intercal) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}_i^\intercal) - \mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\intercal) + \mathbf{e}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\intercal) - \mathbf{e}_{4,i} \\
&= \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}_i^\intercal) + \mathbf{e}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\intercal) - \mathbf{e}_{4,i}
\end{aligned}$$

Clearly, the two hybrids differ only in the error terms in $\mathbf{c}_{4,i}$. Thus, the indistinguishability

follows due to the following:

$$\mathbf{e}_{4,i} \approx -\mathbf{e}_{4,i} + \mathbf{e}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\mathsf{T})$$

which is true since the distribution of $-\mathbf{e}_{4,i}$ is the same as that of $\mathbf{e}_{4,i}$ by the symmetry of the discrete Gaussian distribution and $\chi_4 \geq \gamma\chi_1\lambda^{\omega(1)}$. ■

Claim 4.16. $\mathsf{G}_1 \approx_c \mathsf{G}_2$ due to LWE.

Proof. Let us write $\mathbf{B}$ as $(\mathbf{B}_U^\mathsf{T} \ \mathbf{B}_M^\mathsf{T} \ \mathbf{B}_L^\mathsf{T})^\mathsf{T}$. Then we can see that the indistinguishability follows from LWE by applying Lemma 4.12 for $k = 1$, which implies $(\mathbf{A}_0, \mathbf{B}_M, \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1, \mathbf{s}_0\mathbf{B}_M + \mathbf{e}_2) \approx_c (\mathbf{A}_0, \mathbf{B}_M, \mathbf{w}_1', \mathbf{w}_2')$, where $\mathbf{w}_1' \leftarrow \mathbb{Z}_q^{m^2\ell}, \mathbf{w}_2' \leftarrow \mathbb{Z}_q^{(2nm+nm^2)w}$.

In particular, let $\mathbf{B} = (\mathbf{B}_U^\mathsf{T} \ \mathbf{B}_M^\mathsf{T} \ \mathbf{B}_L^\mathsf{T})^\mathsf{T}$. Then
In G_1,

$$\begin{aligned}
(\mathbf{c}_1, \mathbf{c}_2) &= (\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1, \mathbf{s}\mathbf{B}_U + \mathbf{s}_0\mathbf{B}_M + \mathbf{s}_1\mathbf{B}_L + \mathbf{e}_2) \\
&\approx_c (\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{w}_1', \mathbf{s}\mathbf{B}_U + \mathbf{s}_1\mathbf{B}_L + \mathbf{w}_2') \quad \text{(from LWE)} \\
&\approx_s (\mathbf{w}_1, \mathbf{w}_2) \quad \text{where } \mathbf{w}_1 \leftarrow \mathbb{Z}_q^{m^2\ell}, \mathbf{w}_2 \leftarrow \mathbb{Z}_q^{(2nm+nm^2)w}
\end{aligned}$$

■

Claim 4.17. $\mathsf{G}_2 \approx_s \mathsf{G}_3$

Proof. The two hybrids differ only in the error terms in $\mathbf{c}_{7,i,j}$ and are indistinguishable due to the smudging lemma.
In G_2:

$$\mathbf{c}_{7,i,j} = \mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T}) + \mathbf{e}_{7,i,j}$$

In G_3:

$$\mathbf{c}_{7,i,j} = \mathbf{c}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{e}_{7,i,j}$$

$$
\begin{aligned}
&= (\mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{I}) + \mathbf{e}_{6,j})(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{e}_{7,i,j} \\
&= \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{e}_{7,i,j} \\
&= \mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T}) + \mathbf{e}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T}) - \mathbf{e}_{7,i,j}
\end{aligned}
$$

Clearly, the two hybrids differ only in the error terms in $\mathbf{c}_{7,i,j}$. Thus, the indistinguishability follows due to the following:

$$
\mathbf{e}_{7,i,j} \quad \approx_s \quad -\mathbf{e}_{7,i,j} + \mathbf{e}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\mathsf{T})
$$

which is true since $\chi_7 \geq \gamma\chi_6\lambda^{\omega(1)}$ and by the symmetry of the discrete Gaussian distribution. ■

Claim 4.18. $\mathsf{G}_3 \approx \mathsf{G}_4$

Proof. The indistinguishability follows from Lemma 4.22. ■

Claim 4.19. $\mathsf{G}_4 \approx_s \mathsf{G}_5$.

Proof. The two hybrids differ only in the error terms in $\mathbf{c}_{7,i,j}$. The indistinguishability follows from the smudging lemma.

In G_4,

$$
\mathbf{c}'_{7,i,j} = \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j}.
$$

In G_5,

$$
\begin{aligned}
\mathbf{c}'_{7,i,j} &= (\mathbf{c}'_{4,i} \| \mathbf{c}_{3,i})\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}_0\|\mathbf{x}_{1,i})}\mathbf{U}_j + \mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j} \\
&= \left(\mathbf{s}((\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{4,i} | \mathbf{s}((\mathbf{A}_2 - \mathbf{x}_{1,i} \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{3,i}\right) \widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}_0\|\mathbf{x}_{1,i})}\mathbf{U}_j \\
&\quad + \mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j} \\
&= \mathbf{s}((\mathbf{A}_1 \| \mathbf{A}_2 - (\mathbf{x}_0 \| \mathbf{x}_{1,i}) \otimes \mathbf{G}) \otimes \mathbf{r}_i^\mathsf{T})\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}_0\|\mathbf{x}_{1,i})}\mathbf{U}_j + (\mathbf{e}_{4,i} \| \mathbf{e}_{3,i})\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}_0\|\mathbf{x}_{1,i})}\mathbf{U}_j \\
&\quad + \mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j} \\
&= \mathbf{s}((\mathbf{A}_f - f_j(\mathbf{x}_0 \| \mathbf{x}_{1,i})\mathbf{G})\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + (\mathbf{e}_{4,i} \| \mathbf{e}_{3,i})\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}_0\|\mathbf{x}_{1,i})}\mathbf{U}_j + \mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{7,i,j}
\end{aligned}
$$

$$= \mathbf{s}(\mathbf{A}_f \mathbf{U}_j \otimes \mathbf{r}_i^\mathsf{T}) + (\mathbf{e}_{4,i} \| \mathbf{e}_{3,i}) \widehat{\mathbf{H}}_{\mathbf{A}, f_j, (\mathbf{x}_0 \| \mathbf{x}_{1,i})} \mathbf{U}_j + \mathbf{e}_{7,i,j} \quad (\text{since } f_j(\mathbf{x}_0 \| \mathbf{x}_{1,i}) = 1)$$

Clearly, the two hybrids differ only in the error terms in $\mathbf{c}'_{7,i,j}$. Thus, the indistinguishability follows due to the following:

$$\mathbf{e}_{7,i,j} \approx_s \mathbf{e}_{7,i,j} + (\mathbf{e}_{4,i} \| \mathbf{e}_{3,i}) \widehat{\mathbf{H}}_{\mathbf{A}, f_j, (\mathbf{x}_0 \| \mathbf{x}_{1,i})} \mathbf{U}_j$$

which is true when $\chi_7 \geq \chi_3 \chi_4 \beta \gamma \lambda^{\omega(1)}$, where $\|\widehat{\mathbf{H}}_{\mathbf{A}, f_j, (\mathbf{x}_0 \| \mathbf{x}_{1,i})}\|_\infty \leq \beta$ ■

Claim 4.20. $\mathsf{G}_5 \approx_c \mathsf{G}_6$ under the tensor-LWE assumption.

Proof. The indistinguishability between the two hybrids follows from tensor-LWE which implies

$$\mathbf{A}_1, \mathbf{A}_2, \{\mathbf{U}_j, \mathbf{r}_i^\mathsf{T}, \mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{A}_1 - \mathbf{x}_0 \otimes \mathbf{G}) + \mathbf{e}_{4,i}, \mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})(\mathbf{A}_2 - \mathbf{x}_{1,i} \otimes \mathbf{G}) + \mathbf{e}_{3,i},$$
$$\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_i^\mathsf{T})\mathbf{G}\mathbf{U}_j + \mathbf{e}_{7,i,j}\}_{i,j} \approx_c \mathbf{A}_1, \mathbf{A}_2, \{\mathbf{U}_j, \mathbf{r}_i^\mathsf{T}, \mathsf{random}, \mathsf{random}\}_{i,j}.$$

■

Claim 4.21. $\mathsf{G}_6 \equiv \mathsf{G}_7$

Proof. This follows since in G_6, $\mathbf{c}_{4,i}$ and $\mathbf{c}_{7,i,j}$ are masked by random vectors $\mathbf{c}'_{4,i}$ and $\mathbf{c}'_{7,i,j}$, respectively. ■

To complete the proof of Lemma 4.14, it remains to prove the following.

Lemma 4.22. *Given* $\{\mathbf{t}_j\}_{j \in [Q]}$, $\{\mathbf{r}_i\}_{i \in [Q]}$, $\mathbf{C}, \mathbf{D}$,

$$(\{\mathbf{z}_{C,i} := \mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_i^\mathsf{T})\mathbf{C} + \mathbf{e}_{5,i}\}_i, \{\mathbf{z}_{D,j} := \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\mathsf{T} \otimes \mathbf{I}) + \mathbf{e}_{6,j}\}_j) \approx_c (\{\mathbf{w}'_{5,i}\}_i, \{\mathbf{w}'_j\}_{6,j}),$$

where $\mathbf{w}'_{5,i} \leftarrow \mathbb{Z}_q^{nmw}$ *and* $\mathbf{w}'_{6,j} \leftarrow \mathbb{Z}_q^{m^2}$ *assuming* LWE.

Proof. We prove the lemma by considering a sequence of games where we start from the LHS and gradually change it to that of RHS in a way that is not noticed by the adversary.

$\tilde{\mathsf{G}}_0$: This is the same distribution as in LHS.

$\tilde{\mathsf{G}}_1$: In this hybrid, we change the distribution to be

$$\left\{ \mathbf{z}_{C,i} = \underbrace{(\mathbf{s}_1(\mathbf{C}\otimes\mathbf{I}) + \mathbf{e}_C)}_{:=\mathbf{s}_C}(\mathbf{I}_{nmw}\otimes\mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{5,i} \right\}_i,$$

$$\left\{ \mathbf{z}_{D,j} = \underbrace{\left(\mathbf{s}_1(\mathbf{D}\otimes\mathbf{I}^{\otimes 2}) + \mathbf{e}_D\right)}_{:=\mathbf{s}_D}(\mathbf{I}\otimes\mathbf{t}_j^\mathsf{T}\otimes\mathbf{I}) + \mathbf{e}_{6,j} \right\}_j,$$

where $\mathbf{e}_C \leftarrow \mathcal{D}^{nm^2w}_{\mathbb{Z},\chi_1}$, $\mathbf{e}_D \leftarrow \mathcal{D}^{m^3}_{\mathbb{Z},\chi_1}$.

This hybrid differs from the previous one only in the error terms in $\mathbf{z}_{C,i}$ and $\mathbf{z}_{D,j}$. The indistinguishability follows from the smudging lemma.

To see this, observe that we have

$$(\mathbf{s}_1(\mathbf{C}\otimes\mathbf{I}) + \mathbf{e}_C)\,(\mathbf{I}_{nmw}\otimes\mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{5,i} = \mathbf{s}_1(\mathbf{C}\otimes\mathbf{r}_i^\mathsf{T}) + \underbrace{\mathbf{e}_C(\mathbf{I}_{nmw}\otimes\mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{5,i}}_{=\text{error}}$$

and

$$\left(\mathbf{s}_1(\mathbf{D}\otimes\mathbf{I}^{\otimes 2}) + \mathbf{e}_D\right)(\mathbf{I}\otimes\mathbf{t}_j^\mathsf{T}\otimes\mathbf{I}) + \mathbf{e}_{6,j} = \mathbf{s}_1(\mathbf{D}\otimes\mathbf{t}_j^\mathsf{T}\otimes\mathbf{I}) + \underbrace{\mathbf{e}_D(\mathbf{I}\otimes\mathbf{t}_j^\mathsf{T}\otimes\mathbf{I}) + \mathbf{e}_{6,j}}_{=\text{error}}.$$

Thus, the indistinguishability follows due to the following:

$$\mathbf{e}_{5,i} \approx_s \mathbf{e}_C(\mathbf{I}_{nmw}\otimes\mathbf{r}_i^\mathsf{T}) + \mathbf{e}_{5,i}, \quad \mathbf{e}_{6,j} \approx_s \mathbf{e}_D(\mathbf{I}\otimes\mathbf{t}_j^\mathsf{T}\otimes\mathbf{I}) + \mathbf{e}_{6,j},$$

which is true when $\chi_5, \chi_6 > \gamma\lambda^{\omega(1)}\chi_1$.

$\tilde{\mathsf{G}}_2$: In this hybrid, we replace $\mathbf{s}_C$ and $\mathbf{s}_D$ with random vectors sampled as $\mathbf{s}_C \leftarrow \mathbb{Z}_q^{nm^2w}$,

$\mathbf{s}_D \leftarrow \mathbb{Z}_q^{m^3}$. This hybrid is indistinguishable from the previous one by Lemma 4.12 with $k = 2$ assuming LWE.

$\tilde{\mathsf{G}}_3$: In this game, $\{\mathbf{z}_{C,i}\}_i$ and $\{\mathbf{z}_{D,j}\}_j$ are replaced with random vectors sampled as $\mathbf{z}_{C,i} \leftarrow \mathbb{Z}_q^{nmw}$ and $\mathbf{z}_{D,j} \leftarrow \mathbb{Z}_q^{m^2}$ for all $i, j \in [Q]$. We can see that this hybrid is indistinguishable from the previous one by LWE with low norm samples (Lemma 4.1) once with respect to secret $\mathbf{s}_C$, and then with respect to $\mathbf{s}_D$.

It is clear that the distribution in $\tilde{\mathsf{G}}_3$ is the same as that of RHS in the statement of the lemma. ■

This completes the proof of Lemma 4.14. ■

4.7 MULTI-INPUT ABE FOR ANY CONSTANT ARITY

In this section, we extend the construction in Sec. 4.6 to construct k-ABE for any constant k using evasive LWE. Our main construction supports functions in NC_1 and proven secure assuming evasive LWE. We also discuss a variant that supports any polynomial size circuit of bounded depth, which can be proven secure assuming a strengthening of tensor LWE in addition.

4.7.1 Construction for NC_1 Circuits

Here, we show our construction. Let ℓ be the length of each of the k attributes. Decryption is possible when $f(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}) = 0$, where $\mathbf{x}_0 \in \{0,1\}^\ell$ is the attribute associated with the public encryption, $\mathbf{x}_i \in \{0,1\}^\ell$ is the attribute associated with the slot i, and f is a binary circuit associated with the slot k key. Below $\mathbf{I}$ refers to $\mathbf{I}_m$. We require an upper bound on the depth of the circuit and denote it by d. We require $d = O(\log \lambda)$.

In the construction, we will use the low-norm variant of the lattice evaluation algorithms (EvalF, EvalFX) from Lemma 4.4.

Table 4.2: Summary of hybrids in the proof of security for 2ABE construction. All the $\mathbf{w}$ and $\mathbf{w}'$ vectors are sampled randomly.

To prove $D_0'' \approx D_1''$: Given $\mathbf{C}$, $\mathsf{aux}'' = (\mathbf{x}, \{\mathbf{y}_i, \mathbf{r}_i\}_i, \mathsf{coins}_{\mathcal{A}}, \{f_j, \mathbf{t}_j, \mathbf{U}_j\}_j \mathbf{A}_0, \mathbf{A}, \mathbf{B}, \mathbf{D})$, to prove pseudorandomness of $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_{3,i}, \mathbf{c}_{4,i}, \mathbf{c}_{5,i}, \mathbf{c}_{6,j}, \mathbf{c}_{7,i,j}$								
	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_{3,i}$	$\mathbf{c}_{4,i}$	$\mathbf{c}_{5,i}$	$\mathbf{c}_{6,j}$	$\mathbf{c}_{7,i,j}$	Remark
G_0	$\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{I}) + \mathbf{e}_1$	$(\mathbf{s}, \mathbf{s}_0, \mathbf{s}_1)\mathbf{B} + \mathbf{e}_2$	$\mathbf{s}((\mathbf{A}_2 - \mathbf{y}_i \otimes \mathbf{G}) \otimes \mathbf{r}_i^\top) + \mathbf{e}_{3,i}$	$\mathbf{s}_0(\mathbf{A}_0 \otimes \mathbf{r}_i^\top) + \mathbf{e}_{4,i}$	$\mathbf{s}_1(\mathbf{I}_{nm} \otimes \mathbf{r}_i^\top)\mathbf{C} + \mathbf{e}_{5,i}$	$\mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I}) + \mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\top \otimes \mathbf{I}) + \mathbf{e}_{6,j}$	$\mathbf{s}_1(\mathbf{D} \otimes \mathbf{t}_j^\top \otimes \mathbf{r}_i^\top) + \mathbf{e}_{7,i,j}$	
G_1	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\top) - (\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{r}_i^\top) + \mathbf{e}_{4,i})$	$\downarrow$	$\downarrow$	$\downarrow$	smudging lemma
$\mathsf{G}_{1.5}$	$\mathbf{s}((\mathbf{A}_1 - \mathbf{x} \otimes \mathbf{G}) \otimes \mathbf{I}) + \mathbf{w}_1'$,	$(\mathbf{s}, \mathbf{0}, \mathbf{s}_1)\mathbf{B} + \mathbf{w}_2'$,	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	LWE
G_2	$\mathbf{w}_1$	$\mathbf{w}_2$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	random mask
G_3	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{c}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\top) - \underbrace{\left(\mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{r}_i^\top) + \mathbf{e}_{7,i,j}\right)}_{:=\mathbf{c}'_{7,i,j}}$	smudging
$\mathsf{G}_{3.5}$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{w}_{5,i}$	$\mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I}) + \mathbf{w}'_{6,j}$	$\downarrow$	LWE
G_4	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{w}_{6,j}$		random mask
G_5	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{c}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\top) - [\mathbf{c}'_{4,i} \| \mathbf{c}_{3,i}]\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}\|\mathbf{y}_i)}\mathbf{U}_j - \underbrace{(\mathbf{s}(\mathbf{G}\mathbf{U}_j \otimes \mathbf{r}_i^\top) + \mathbf{e}_{7,i,j})}_{:=\mathbf{c}''_{7,i,j}}$	smudging
G_6	$\downarrow$	$\downarrow$	$\mathbf{w}_{3,i}$	$\mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_i^\top) - \mathbf{w}'_{4,i}$	$\downarrow$	$\downarrow$	$\mathbf{c}_{6,j}(\mathbf{I} \otimes \mathbf{r}_i^\top) - [\mathbf{c}'_{4,i} \| \mathbf{c}_{3,i}]\widehat{\mathbf{H}}_{\mathbf{A},f_j,(\mathbf{x}\|\mathbf{y}_i)}\mathbf{U}_j - \mathbf{w}''_{7,i,j}$	tensor LWE
G_7	$\downarrow$	$\downarrow$	$\downarrow$	$\mathbf{w}_{4,i}$	$\downarrow$	$\downarrow$	$\mathbf{w}_{7,i,j}$	random mask

$\mathsf{Setup}(1^\lambda)$**:** The setup algorithm takes as input the security parameter and does the following:

- Sample $\mathbf{A}_0, \ldots, \mathbf{A}_{k-1} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m\times m\ell}$; $\mathbf{D}_0, \ldots, \mathbf{D}_{k-1} \leftarrow \mathbb{Z}_q^{n\times m\ell}$, $\mathbf{D}_k \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{U} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m\times m}$;
- Sample $(\mathbf{B}, \mathbf{B}_{\tau_B}^{-1}) \leftarrow \mathsf{TrapGen}(1^\lambda, m^{k+1}+(k+1)nm^k, (m^{k+1}+(k+1)nm^k)w)$, where $w \in O(\log q)$;
  $\{(\mathbf{C}_i, \mathbf{C}_{i,\tau_C}^{-1}) \leftarrow \mathsf{TrapGen}(1^\lambda, (k+1)nm^{i-1}, (k+1)nm^{i-1}w)\}_{i\in[2,k]}$
- Set $\mathbf{C}_1 = \begin{pmatrix} \mathbf{D}_0 & & & \\ & \mathbf{D}_1 & & \\ & & \ddots & \\ & & & \mathbf{D}_k \end{pmatrix}$
- Let $\mathbf{A} = (\mathbf{A}_0, \ldots, \mathbf{A}_{k-1})$.
  Output $\mathsf{mpk} = (\mathbf{A}, \mathbf{B}, \mathbf{C}_1, \ldots, \mathbf{C}_k, \mathbf{D}_0, \ldots, \mathbf{D}_k, \mathbf{U})$, $\mathsf{msk} = (\mathbf{B}, \mathbf{B}_{\tau_B}^{-1}, \mathbf{C}_{2,\tau_C}^{-1}, \ldots, \mathbf{C}_{k,\tau_C}^{-1})$.

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0, \mu)$**:** The Enc algorithm is a public encryption algorithm. It takes as input the master public key mpk, attribute $\mathbf{x}_0$ and message bit $\mu \in \{0,1\}$ and does the following:

- Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^{m^{k+1}}$, $\mathbf{s}_0, \ldots, \mathbf{s}_k \leftarrow \mathbb{Z}_q^{nm^k}$.
- If $\mu = 1$, sample $\mathbf{c}_1 \leftarrow \mathbb{Z}_q^{\ell m^{k+1}}$, $\mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(m^{k+1}+(k+1)nm^k)w}$.
  Else, compute
  - $\mathbf{c}_1 = \mathbf{s}((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}) \otimes \mathbf{I}^{\otimes k}) + \mathbf{s}_0(\mathbf{D}_0 \otimes \mathbf{I}^{\otimes k}) + \mathbf{e}_1$, where $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_1}^{\ell m^{k+1}}$.
  - $\mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \cdots, \mathbf{s}_k)\mathbf{B} + \mathbf{e}_2$, where $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_2}^{(m^{k+1}+(k+1)nm^k)w}$.
- Output $\mathsf{ct}_{\mathbf{x}_0} = (\mathbf{c}_1, \mathbf{c}_2)$.

$\mathsf{KeyGen}_i(\mathsf{msk}, \mathbf{x}_i)$ **for** $1 \le i \le k-1$**:** The keygen algorithm for slot $1 \le i \le k-1$, takes as input the master secret key msk and attribute $\mathbf{x}_i$ and does the following:

- Samples $\mathbf{r}_i \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^m$

- Samples $\mathbf{X}_i \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix} (\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{I}) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\intercal \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{inm^k \times \ell m^k} \\ \mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\intercal \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{(k-i)nm^k \times \ell m^k} \end{pmatrix}, \tau_B\right)$ and
  $\mathbf{Y}_i \leftarrow \mathbf{C}_{i+1}^{-1}\left((\mathbf{C}_i \otimes \mathbf{r}_i^\intercal), \tau_C\right)$
- Returns $\mathsf{sk}_{i,\mathbf{x}_i} = (\mathbf{r}_i, \mathbf{X}_i, \mathbf{Y}_i)$

$\mathsf{KeyGen}_k(\mathsf{msk}, f)$**:** The keygen algorithm for slot k takes as input the master secret key, msk, and k-arity function f and does the following:

- Samples $\mathbf{r}_k \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^m$
- Computes $\mathbf{H}_f = \mathsf{EvalF}(\mathbf{A}, f)$ and $\mathbf{A}_f = \mathbf{A}\mathbf{H}_f$
- Computes $\mathbf{M}_f \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix} \mathbf{A}_f\mathbf{U} \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\intercal \\ \mathbf{0}_{knm^k \times m^k} \\ \mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\intercal \end{pmatrix}, \tau_B\right)$ and
  $\mathbf{N}_f \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix} \mathbf{0}_{m^{k+1} \times (k+1)nm^{k-1}w} \\ \mathbf{C}_k \otimes \mathbf{r}_k^\intercal \end{pmatrix}, \tau_B\right)$
- Returns $\mathsf{sk}_{k,f} = \left(\mathbf{r}_k, \mathbf{M}_f, \mathbf{N}_f\right)$

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{ct}_{\mathbf{x}_0}, \mathsf{sk}_{1,\mathbf{x}_1}, \ldots, \mathsf{sk}_{k-1,\mathbf{x}_{k-1}}, \mathsf{sk}_{k,f})$ The decryption algorithm takes a ciphertext $\mathsf{ct}_{\mathbf{x}_0}$, k keys $\mathsf{sk}_{1,\mathbf{x}_1}, \ldots, \mathsf{sk}_{k-1,\mathbf{x}_{k-1}}$ and $\mathsf{sk}_{k,f}$ and does the following:

- Parse $\mathsf{ct}_{\mathbf{x}_0}$ as $(\mathbf{c}_1, \mathbf{c}_2)$, $\mathsf{sk}_{i,\mathbf{x}_i}$ as $(\mathbf{r}_i, \mathbf{X}_i, \mathbf{Y}_i)$ for $1 \le i \le k-1$ and $\mathsf{sk}_{k,f}$ as $(\mathbf{r}_k, \mathbf{M}_f, \mathbf{N}_f)$.

  Let $\mathbf{x} = (\mathbf{x}_0, \ldots, \mathbf{x}_{k-1})$.
- Compute $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{A}, f, \mathbf{x})$.
- Compute the following
  * $\mathbf{d}'_0 = \mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)$
  * $\mathbf{d}'_i = \mathbf{c}_2\mathbf{X}_i(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\intercal \cdots \otimes \mathbf{r}_{i-1}^\intercal \otimes \mathbf{r}_{i+1}^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)$, for $1 \le i \le k-1$,
  * $\mathbf{d}'_f = \mathbf{c}_2\mathbf{M}_f(\mathbf{I}_m \otimes \mathbf{r}_1^\intercal \cdots \otimes \mathbf{r}_{k-1}^\intercal)$

* $(\mathbf{d}''_0, \cdots, \mathbf{d}''_{k-1}, \mathbf{d}''_f) = \mathbf{c}_2 \mathbf{N}_f \mathbf{Y}_{k-1} \cdots \mathbf{Y}_1$

* $\mathbf{d}_i = \mathbf{d}'_i - \mathbf{d}''_i$, for $i = 0$ to $k-1$.

* $\mathbf{d}_f = \mathbf{d}'_f - \mathbf{d}''_f$

* $\mathbf{d} = (\mathbf{d}_0| \cdots |\mathbf{d}_{k-1})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U} - \mathbf{d}_f$

- If $\|\mathbf{d}\|_\infty \leq \beta_0$, where β_0 is as defined in section 4.7.1 then return $\mu = 0$, else return $\mu = 1$.

Correctnes. Here, we show the correctness of the scheme.

When $\mu = 1$: We first show the correctness for the case of $\mu = 1$. For an honest run of the protocol, $\mathbf{c}_1$ is distributed uniformly at random over its domain. Then, since $\mathbf{r}_i \neq \mathbf{0}$ for all $i \in [k]$ with overwhelming probability and thus $\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top$ is a full-rank matrix, $\mathbf{d}'_0$ and thus $\mathbf{d}_0$ are distributed uniformly at random over their domains. Then, since the topmost m rows of $\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$ constitutes an identity matrix by Lemma 4.4, $(\mathbf{d}_0\| \cdots \|\mathbf{d}_{k-1})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}$ is distributed uniformly at random over its domain. Finally, since each column of $\mathbf{U}$ is chosen from $\mathcal{D}^m_{\mathbb{Z},\gamma}$, with overwhelming probability, there exists $i \in [m]$ such that the i-th column of $\mathbf{U}$ is not a zero vector. This in turn implies that that the i-th entry of $\mathbf{d}$ is distributed uniformly at random over $\mathbb{Z}_q$. Since we set $\beta_0/q = \lambda^{-\omega(1)}$, the probability that the decryption algorithm falsely outputs 0 is negligible as desired.

When $\mu = 0$: We then show the correctness for the case of $\mu = 0$.

* Let us first compute $\mathbf{d}'_0$.

$$\begin{aligned}\mathbf{d}'_0 &= \mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top) \\ &= \left(\mathbf{s} \cdot \left((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes k}\right) + \mathbf{s}_0 \cdot (\mathbf{D}_0 \otimes \mathbf{I}^{\otimes k})\right) \cdot \left(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top\right) + \mathbf{e}_{\mathbf{d}'_0} \\ &= \mathbf{s} \cdot \left((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}_m) \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top\right) + \mathbf{s}_0 \cdot \left(\mathbf{D}_0 \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top\right) + \mathbf{e}_{\mathbf{d}'_0}\end{aligned}$$

where $\mathbf{e}_{\mathbf{d}'_0} := \mathbf{e}_1 \cdot (\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\top \otimes \cdots \otimes \mathbf{r}_k^\top)$.

* Let $1 \leq i \leq k-1$,

$$\begin{aligned}
\mathbf{d}'_i &= \mathbf{c}_2 \cdot \mathbf{X}_i \cdot \left(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{i-1}^\mathsf{T} \otimes \mathbf{r}_{i+1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}\right) \\
&= \left((\mathbf{s},\ \mathbf{s}_0,\ \cdots,\ \mathbf{s}_k)\,\mathbf{B} + \mathbf{e}_2\right) \\
&\quad \cdot \mathbf{B}^{-1}\left(\begin{pmatrix} (\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{inm^k \times \ell m^k} \\ \mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{(k-i)nm^k \times \ell m^k} \end{pmatrix}, \tau_B\right) \\
&\quad \cdot \left(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{i-1}^\mathsf{T} \otimes 1 \otimes \mathbf{r}_{i+1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}\right) \\
&= \left(\mathbf{s}((\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)}) + \mathbf{s}_i \cdot (\mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_i^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)})\right) \\
&\quad \cdot \left(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{i-1}^\mathsf{T} \otimes 1 \otimes \mathbf{r}_{i+1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}\right) + \mathbf{e}_{\mathbf{d}'_i}. \\
&= \mathbf{s} \cdot ((\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{I}_m) \otimes \mathbf{r}_1^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}) + \mathbf{s}_i \cdot (\mathbf{D}_i \otimes \mathbf{r}_1^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}) + \mathbf{e}_{\mathbf{d}'_i}
\end{aligned}$$

where $\mathbf{e}_{\mathbf{d}'_i} := \mathbf{e}_2 \cdot \mathbf{X}_i(\mathbf{I}_{m\ell} \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{i-1}^\mathsf{T} \otimes \mathbf{r}_{i+1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T})$.

* Now we compute $\mathbf{d}'_f$.

$$\begin{aligned}
\mathbf{d}'_f &= \mathbf{c}_2\mathbf{M}_f(\mathbf{I}_m \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{k-1}^\mathsf{T}) \\
&= ((\mathbf{s},\ \mathbf{s}_0,\ \cdots,\ \mathbf{s}_k)\mathbf{B} + \mathbf{e}_2)\mathbf{B}^{-1}\left(\begin{pmatrix} \mathbf{A}_f\mathbf{U} \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\mathsf{T} \\ \mathbf{0}_{knm^k \times m^k} \\ \mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\mathsf{T} \end{pmatrix}, \tau_B\right) \\
&\quad \cdot (\mathbf{I}_m \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{k-1}^\mathsf{T} \otimes 1) \\
&= \left(\mathbf{s} \cdot (\mathbf{A}_f\mathbf{U} \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\mathsf{T}) + \mathbf{s}_k \cdot (\mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_k^\mathsf{T})\right) \\
&\quad \cdot (\mathbf{I}_m \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{k-1}^\mathsf{T} \otimes 1) + \mathbf{e}_{\mathbf{d}'_f} \\
&= \mathbf{s} \cdot (\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}_1^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}) + \mathbf{s}_k \cdot (\mathbf{D}_k \otimes \mathbf{r}_1^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_k^\mathsf{T}) + \mathbf{e}_{\mathbf{d}'_f}
\end{aligned}$$

where $\mathbf{e}_{\mathbf{d}'_f} := \mathbf{e}_2\mathbf{M}_f(\mathbf{I} \otimes \mathbf{r}_1^\mathsf{T} \cdots \otimes \mathbf{r}_{k-1}^\mathsf{T})$.

* Next, we compute:

$$
\begin{aligned}
&(\mathbf{d}''_0,\ \cdots,\ \mathbf{d}''_{k-1},\ \mathbf{d}''_f)\\
&= \mathbf{c}_2\mathbf{N}_f\mathbf{Y}_{k-1}\cdots\mathbf{Y}_1\\
&= ((\mathbf{s},\ \mathbf{s}_0,\ \cdots,\ \mathbf{s}_k)\mathbf{B}+\mathbf{e}_2)\mathbf{B}^{-1}\left(\begin{pmatrix}\mathbf{0}_{m^{k+1}\times(k+1)nm^{k-1}w}\\ \mathbf{C}_k\otimes\mathbf{r}_k^\mathsf{T}\end{pmatrix},\tau_B\right)\cdot\mathbf{Y}_{k-1}\cdots\mathbf{Y}_1\\
&= (\mathbf{s}_0,\cdots,\mathbf{s}_k)\cdot(\mathbf{C}_k\otimes\mathbf{r}_k^\mathsf{T})\cdot\mathbf{Y}_{k-1}\cdots\mathbf{Y}_1+(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k})\\
&= (\mathbf{s}_0,\cdots,\mathbf{s}_k)\cdot(\mathbf{C}_k\otimes\mathbf{r}_k^\mathsf{T})\cdot(\mathbf{C}_k^{-1}\left((\mathbf{C}_{k-1}\otimes\mathbf{r}_{k-1}^\mathsf{T}),\tau_C\right)\otimes 1)\cdot\mathbf{Y}_{k-2}\cdots\mathbf{Y}_1\\
&\quad+(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k})\\
&= (\mathbf{s}_0,\cdots,\mathbf{s}_k)\cdot(\mathbf{C}_k\cdot\mathbf{C}_k^{-1}\left((\mathbf{C}_{k-1}\otimes\mathbf{r}_{k-1}^\mathsf{T}),\tau_C\right)\otimes\mathbf{r}_k^\mathsf{T})\cdot\mathbf{Y}_{k-2}\cdots\mathbf{Y}_1\\
&\quad+(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k})\\
&= (\mathbf{s}_0,\cdots,\mathbf{s}_k)\cdot(\mathbf{C}_{k-1}\otimes\mathbf{r}_{k-1}^\mathsf{T}\otimes\mathbf{r}_k^\mathsf{T})\cdot\mathbf{Y}_{k-2}\cdots\mathbf{Y}_1+(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k})\\
&= \vdots\\
&= (\mathbf{s}_0,\cdots,\mathbf{s}_k)\cdot(\mathbf{C}_1\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k})\\
&= (\mathbf{s}_0\cdot(\mathbf{D}_0\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{s}_k\cdot(\mathbf{D}_k\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{e}_{\mathbf{d}''_k})
\end{aligned}
$$

with $(\mathbf{e}_{\mathbf{d}''_0},\cdots,\mathbf{e}_{\mathbf{d}''_k}) := \mathbf{e}_2\mathbf{N}_f\mathbf{Y}_{k-1}\cdots\mathbf{Y}_1$.

* Let $0 \le i \le k-1$,

$$
\begin{aligned}
\mathbf{d}_i &= \mathbf{d}'_i-\mathbf{d}''_i\\
&= \mathbf{s}((\mathbf{A}_i-\mathbf{x}_i\otimes\mathbf{I}_m)\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{s}_i(\mathbf{D}_i\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})\\
&\quad+\mathbf{e}_{\mathbf{d}'_i}-\mathbf{s}_i(\mathbf{D}_i\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})-\mathbf{e}_{\mathbf{d}''_i}\\
&= \mathbf{s}((\mathbf{A}_i-\mathbf{x}_i\otimes\mathbf{I}_m)\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{e}_{\mathbf{d}_i}
\end{aligned}
$$

with $\mathbf{e}_{\mathbf{d}_i} := \mathbf{e}_{\mathbf{d}'_i}-\mathbf{e}_{\mathbf{d}''_i}$.

* Next, $\mathbf{d}_f = \mathbf{d}'_f-\mathbf{d}''_f$. So,

$$
\begin{aligned}
\mathbf{d}_f &= \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{s}_k(\mathbf{D}_k\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{e}_{\mathbf{d}'_f}\\
&\quad-\mathbf{s}_k(\mathbf{D}_k\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})-\mathbf{e}_{\mathbf{d}''_f}\\
&= \mathbf{s}(\mathbf{A}_f\mathbf{U}\otimes\mathbf{r}_1^\mathsf{T}\otimes\cdots\otimes\mathbf{r}_k^\mathsf{T})+\mathbf{e}_{\mathbf{d}_f}
\end{aligned}
$$

with $\mathbf{e}_{\mathbf{d}_f} := \mathbf{e}_{\mathbf{d}'_f}-\mathbf{e}_{\mathbf{d}''_f}$ where $\mathbf{e}_{\mathbf{d}''_f} := \mathbf{e}_{\mathbf{d}''_k}$.

* And finally, $\mathbf{d} = (\mathbf{d}_0 \| \cdots \| \mathbf{d}_{k-1}) \cdot \widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U} - \mathbf{d}_f$. First,

$$
\begin{aligned}
&(\mathbf{d}_0 \| \cdots \| \mathbf{d}_{k-1}) \\
&= (\mathbf{s}((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) \| \cdots \| \mathbf{s}((\mathbf{A}_{k-1} - \mathbf{x}_{k-1} \otimes \mathbf{I}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)) \\
&\quad + (\mathbf{e}_{\mathbf{d}_0} \| \cdots \| \mathbf{e}_{\mathbf{d}_{k-1}}) \\
&= (\mathbf{s}(((\mathbf{A}_0 \| \mathbf{A}_1 \| \cdots \| \mathbf{A}_{k-1}) - (\mathbf{x}_0 \| \mathbf{x}_1 \| \cdots \| \mathbf{x}_{k-1}) \otimes \mathbf{I}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)) \\
&\quad + (\mathbf{e}_{\mathbf{d}_0} \| \cdots \| \mathbf{e}_{\mathbf{d}_{k-1}}) \\
&= (\mathbf{s}((\mathbf{A} - \underbrace{(\mathbf{x}_0 \| \mathbf{x}_1 \| \cdots \| \mathbf{x}_{k-1})}_{=\mathbf{x}} \otimes \mathbf{I}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)) + (\mathbf{e}_{\mathbf{d}_0} \| \cdots \| \mathbf{e}_{\mathbf{d}_{k-1}}).
\end{aligned}
$$

From Lemma 4.4, we deduce

$$(\mathbf{A} - \mathbf{x} \otimes \mathbf{I})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}} = \mathbf{A}_f - f(\mathbf{x})\mathbf{I} \mod q.$$

Hence,

$$
\begin{aligned}
&(\mathbf{d}_0 \| \cdots \| \mathbf{d}_{k-1})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U} - \mathbf{d}_f \\
&= \mathbf{s}((\mathbf{A} - \mathbf{x} \otimes \mathbf{I}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal)(\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U} \otimes 1 \otimes 1 \cdots \otimes 1) \\
&\quad + \mathbf{e}_{\mathbf{d}} - \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) - \mathbf{e}_{\mathbf{d}_f} \\
&= \mathbf{s}((\mathbf{A} - \mathbf{x} \otimes \mathbf{I})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) - \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) + \mathbf{e}_{\mathbf{d}} - \mathbf{e}_{\mathbf{d}_f} \\
&= \mathbf{s}((\mathbf{A}_f\mathbf{U} - f(\mathbf{x})\mathbf{U}) \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) - \mathbf{s}(\mathbf{A}_f\mathbf{U} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) + \mathbf{e}_{\mathbf{d}} - \mathbf{e}_{\mathbf{d}_f} \\
&= -\mathbf{s}(f(\mathbf{x})\mathbf{U} \otimes \mathbf{r}_1^\intercal \otimes \cdots \otimes \mathbf{r}_k^\intercal) + \mathbf{e}_{\mathbf{d}} - \mathbf{e}_{\mathbf{d}_f} \\
&= \mathbf{e}_{\mathbf{d}} - \mathbf{e}_{\mathbf{d}_f} \text{ if } f(\mathbf{x}) = 0.
\end{aligned}
$$

where $\mathbf{e}_{\mathbf{d}} := (\mathbf{e}_{\mathbf{d}_0} \| \cdots \| \mathbf{e}_{\mathbf{d}_{k-1}})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U}$. Thus, when $\mu = 0$, $\|\mathbf{d}\|_\infty$ is small ($\leq \beta_0$), and hence, the decryption correctly outputs 0.

Error Bound: The error term is bounded as follows. Let β_0 denote the error bound.

$$
\begin{aligned}
&\|\mathbf{e}_{\mathbf{d}}\|_\infty + \|\mathbf{e}_{\mathbf{d}_f}\|_\infty \\
&= \|(\mathbf{e}_{\mathbf{d}_0} \| \cdots \| \mathbf{e}_{\mathbf{d}_{k-1}})\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U}\|_\infty + \|\mathbf{e}_{\mathbf{d}'_f} - \mathbf{e}_{\mathbf{d}''_f}\|_\infty \\
&= \|\Big((\mathbf{e}_{\mathbf{d}'_0} \| \cdots \| \mathbf{e}_{\mathbf{d}'_{k-1}}) - (\mathbf{e}_{\mathbf{d}''_0} \| \cdots \| \mathbf{e}_{\mathbf{d}''_{k-1}})\Big)\widehat{\mathbf{H}}_{\mathbf{A},f,\mathbf{x}}\mathbf{U}\|_\infty + \|\mathbf{e}_{\mathbf{d}'_f} - \mathbf{e}_{\mathbf{d}''_f}\|_\infty \\
&\leq \Big(\Big(\chi_1(m\gamma)^{O(k)} + w\chi_2\tau_B(m\gamma)^{O(k)} + \chi_2\tau_B(w\tau_C)^{O(k)}m^{O(k^2)}\Big) \cdot m\beta\gamma \\
&\quad + w\chi_2\tau_B(m\gamma)^{O(k)} + \chi_2\tau_B(w\tau_C)^{O(k)}m^{O(k^2)}\Big)\,\mathrm{poly}(m) \\
&\leq (m\chi_1\chi_2 w\gamma\tau_B\tau_C)^{O(k^2)} \\
&\leq \beta_0,
\end{aligned}
$$

where we used $\|\mathbf{e}_{\mathbf{d}'_0}\|_\infty \leq \chi_1(m\gamma)^{O(k)}$, $\|\mathbf{e}_{\mathbf{d}'_i}\|_\infty, \|\mathbf{e}_{\mathbf{d}'_f}\|_\infty \leq w\chi_2\tau_B(m\gamma)^{O(k)}$, and $\|\mathbf{e}_{\mathbf{d}''_i}\|_\infty, \|\mathbf{e}_{\mathbf{d}''_f}\|_\infty \leq \chi_2\tau_B(w\tau_C)^{O(k)}m^{O(k^2)}$.

Parameters. We set the parameters as follows.

$$n = \mathrm{poly}(\lambda, 2^d), \quad m = O(n\log q), \quad \tau_B = O(\sqrt{2km^{k+1}\log q}), \quad \tau_C = O(\sqrt{2km^k \log q}),$$
$$\beta = (m\gamma)^{O(2^d)}, \quad \gamma = \lambda^{\omega(1)}, \quad \chi_1 = (m\gamma)^{2k}, \quad \chi_3 = \chi_4 = (m\gamma)^{4k},$$
$$\chi_6 = (m\gamma)^{6k}, \quad \chi_7 = m\beta\ell\chi_6\lambda^{\omega(1)}, \quad \chi_{5,i} = \gamma^{k-i}\cdot\chi_7 \text{ for } i \in [0,k], \quad \chi_2 = \gamma\chi_5$$
$$\beta_0 = (m\chi_1\chi_2 w\gamma\tau_B\tau_C)^{O(k^2)}, \quad q = \beta_0\lambda^{\omega(1)}$$

where we define $\chi_5 := \chi_{5,0}$. We note that in the above, χ_3, χ_4, $\chi_{5,i}$, χ_6, and χ_7 are the parameters that only appear in the security proof.

4.7.2 Security

Here, we prove the following theorem, which asserts the security of our scheme.

Theorem 4.23. *Assuming evasive* LWE *(Assumption 4.6) and* LWE*, our construction for k-input* ABE *for* NC_1 *satisfies very selective security (Definition 4.2).*

Proof. To prove the security, we need to prove the indistinguishability of the two distributions given below. Let Q_i be the number of key queries to $\mathsf{KeyGen}_i(\mathsf{msk}, \cdot)$ oracle for $i \in [k]$. In the following, for simplicity, we let $Q_1 = \cdots = Q_k = Q$. Note that this can be assumed without loss of generality.

Note that compared to Section 4.6.2 where i and j are the indexes for the keys, in this proof, $i \in [k]$ is the index of the key generator, and we denote $j_1, \cdots, j_k \in [Q]$ the indexes of the keys. In the sequel, for the ease of the reading, we often suppress the subscript and simply write j when differentiating the indexes is not necessary.

Distribution D_0:

$$\left(\mathsf{mpk},\quad \mathbf{c}_1 = (\mathbf{s},\ \mathbf{s}_0)\begin{pmatrix}(\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes k} \\ \mathbf{D}_0 \otimes \mathbf{I}^{\otimes k}\end{pmatrix} + \mathbf{e}_1,\quad \mathbf{c}_2 = (\mathbf{s},\ \mathbf{s}_0,\ \cdots,\ \mathbf{s}_k)\mathbf{B} + \mathbf{e}_2,\right.$$
$$\left.\{\mathsf{sk}_{i,\mathbf{x}_{i,j}} = (\mathbf{r}_{i,j}, \mathbf{X}_{i,j}, \mathbf{Y}_{i,j})\}_{i\in[k-1],j\in[Q]},\quad \{\mathsf{sk}_{k,f_j} = (\mathbf{r}_{k,j}, \mathbf{M}_{f_j}, \mathbf{N}_{f_j})\}_{j\in[Q]}\right)$$

Distribution D_1:

$$\left(\mathsf{mpk},\quad \mathbf{c}_1 \leftarrow \mathbb{Z}_q^{\ell m^{k+1}},\quad \mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(m^{k+1}+(k+1)nm^k)w},\right.$$
$$\left.\{\mathsf{sk}_{i,\mathbf{x}_{i,j}} = (\mathbf{r}_{i,j}, \mathbf{X}_{i,j}, \mathbf{Y}_{i,j})\}_{i\in[k-1],j\in[Q]},\quad \{\mathsf{sk}_{k,f_j} = (\mathbf{r}_{k,j}, \mathbf{M}_{f_j}, \mathbf{N}_{f_j})\}_{j\in[Q]}\right),$$

where $\mathbf{x}_0$ is the attribute for public encryption, $\mathbf{x}_{i,j}$ for $i \in [k-1]$ is the j-th key query for slot i, and f_j is the j-th key query to $\mathsf{KeyGen}_k(\mathsf{msk}, \cdot)$, $\mathsf{sk}_{i,\mathbf{x}_{i,j}}$ is the j-th key for slot i and sk_{k,f_j} is the key for function f_j. In particular, we have

$$\mathbf{X}_{i,j} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix}(\mathbf{A}_i - \mathbf{x}_{i,j} \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{inm^k\times\ell m^k} \\ \mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{(k-i)nm^k\times\ell m^k}\end{pmatrix}, \tau_B\right)$$
$$\mathbf{Y}_{i,j} \leftarrow \mathbf{C}_{i+1}^{-1}\left((\mathbf{C}_i \otimes \mathbf{r}_{i,j}^\mathsf{T}), \tau_C\right)$$
$$\mathbf{M}_{f_j} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix}\mathbf{A}_{f_j}\mathbf{U}_j \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T} \\ \mathbf{0}_{knm^k\times m^k} \\ \mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T}\end{pmatrix}, \tau_B\right)$$
$$\mathbf{N}_{f_j} \leftarrow \mathbf{B}^{-1}\left(\begin{pmatrix}\mathbf{0}_{m^{k+1}\times(k+1)nm^{k-1}w} \\ \mathbf{C}_k \otimes \mathbf{r}_{k,j}^\mathsf{T}\end{pmatrix}, \tau_B\right)$$

$\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_1}^{\ell m^{k+1}}$, $\mathbf{e}_2 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_2}^{(m^{k+1}+(k+1)nm^k)w}$.

We can see that D_0 and D_1 are the views of the adversary when $\mu = 0$ and $\mu = 1$ are encrypted, respectively. We then apply Evasive LWE (EvLWE) with respect to matrix $\mathbf{B}$ with sampler Samp^1 that outputs $\mathsf{aux}^1 = (\mathsf{aux}^1_1, \mathsf{aux}^1_2), \mathbf{P}^1, \mathbf{S}^1$ as follows:[6]

$$
\begin{aligned}
\mathbf{S}^1 &= (\mathbf{s}, \mathbf{s}_0, \ldots, \mathbf{s}_k) \\
\mathsf{aux}^1_1 &= \mathbf{s}((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}_m) \otimes \mathbf{I}^{\otimes k}) + \mathbf{s}_0(\mathbf{D}_0 \otimes \mathbf{I}^{\otimes k}) + \mathbf{e}_1 \\
\mathsf{aux}^1_2 &= (\mathbf{x}_0, \{\mathbf{x}_{i,j}, \mathbf{r}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{\mathbf{Y}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{f_j, \mathbf{r}_{k,j}\}_{j\in[Q]}, \mathbf{A}, \mathbf{C}_1, \ldots, \mathbf{C}_k, \mathbf{U}) \\
\mathbf{P}_{i,j} &= \begin{pmatrix} (\mathbf{A}_i - \mathbf{x}_{i,j} \otimes \mathbf{I}) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{inm^k \times \ell m^k} \\ \mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)} \\ \mathbf{0}_{(k-i)nm^k \times \ell m^k} \end{pmatrix}, \text{ for } i \in [k-1], j \in [Q] \\
\mathbf{P}_{k,j} &= \begin{pmatrix} \mathbf{A}_{f_j}\mathbf{U} \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T} \\ \mathbf{0}_{knm^k \times m^k} \\ \mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T} \end{pmatrix}, \text{ for } j \in [Q] \\
\mathbf{P}_{k+1,j} &= \begin{pmatrix} \mathbf{0}_{m^{k+1} \times (k+1)nm^{k-1}w} \\ \mathbf{C}_k \otimes \mathbf{r}_{k,j}^\mathsf{T} \end{pmatrix}, \text{ for } j \in [Q] \\
\mathbf{P}^1 &= (\mathbf{P}_{1,1} \| \cdots \| \mathbf{P}_{1,Q} \| \cdots \| \mathbf{P}_{k-1,1} \| \cdots \| \mathbf{P}_{k-1,Q} \| \mathbf{P}_{k,1} \| \cdots \| \mathbf{P}_{k,Q} \| \mathbf{P}_{k+1,1} \| \cdots \| \mathbf{P}_{k+1,Q})
\end{aligned}
$$

Then from Lemma 4.7, to prove that D_0 and D_1 are computationally indistinguishable, it suffices to prove the computational indistinguishability between the following distributions:

Distribution D_0^1:

$$
\begin{pmatrix} \mathsf{aux}^1_2, & \mathbf{B}, & \mathbf{c}_1, & \mathbf{c}_2, \\ & & \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{c}_{k,j}, \mathbf{d}_j\}_{j\in[Q]} \end{pmatrix}
$$

[6] By Lemma 4.7, it suffices to invoke the evasive LWE for a modified sampler that outputs random aux_1. The same comments apply to other invocations of the assumption.

Distribution D_1^1:

$$\left(\mathsf{aux}_2^1,\quad \mathbf{B},\quad \mathbf{v}_1,\quad \mathbf{v}_2,\quad \{\mathbf{v}_{i,j}\}_{i\in[k-1],j\in[Q]},\quad \{\mathbf{v}_{k,j},\mathbf{w}_j\}_{j\in[Q]}\right),$$

where $\mathbf{v}$ (resp., $\mathbf{w}$) vectors above are sampled uniformly at random from the same domain as the corresponding $\mathbf{c}$ (resp., $\mathbf{d}$) vectors and

$$\begin{aligned}
\mathbf{c}_{i,j} &= \mathbf{S}^1\mathbf{P}_{i,j} + \mathbf{e}_{i,j} \\
&= \mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j}\otimes\mathbf{I})\otimes\mathbf{I}^{\otimes(i-1)}\otimes\mathbf{r}_{i,j}^\mathsf{T}\otimes\mathbf{I}^{\otimes(k-i)}) + \mathbf{s}_i(\mathbf{D}_i\otimes\mathbf{I}^{\otimes(i-1)}\otimes\mathbf{r}_{i,j}^\mathsf{T}\otimes\mathbf{I}^{\otimes(k-i)}) \\
&\quad +\mathbf{e}_{i,j} \\
\mathbf{c}_{k,j} &= \mathbf{S}^1\mathbf{P}_{k,j} + \mathbf{e}_{k,j} \\
&= \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U}_j\otimes\mathbf{I}^{\otimes(k-1)}\otimes\mathbf{r}_{k,j}^\mathsf{T}) + \mathbf{s}_k(\mathbf{D}_k\otimes\mathbf{I}^{\otimes(k-1)}\otimes\mathbf{r}_{k,j}^\mathsf{T}) + \mathbf{e}_{k,j} \\
\mathbf{d}_j &= \mathbf{S}^1\mathbf{P}_{k+1,j} + \mathbf{e}'_j \\
&= (\mathbf{s}_0,\ldots,\mathbf{s}_k)(\mathbf{C}_k\otimes\mathbf{r}_{k,j}^\mathsf{T}) + \mathbf{e}'_j \\
&= (\mathbf{s}_0,\ldots,\mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}}\otimes\mathbf{r}_{k,j}^\mathsf{T})\mathbf{C}_k + \mathbf{e}'_j
\end{aligned}$$

where $\mathbf{e}_{i,j} \leftarrow \mathcal{D}^{\ell m^k}_{\mathbb{Z},\chi_3}$, $\mathbf{e}_{k,j} \leftarrow \mathcal{D}^{m^k}_{\mathbb{Z},\chi_4}$, $\mathbf{e}'_j \leftarrow \mathcal{D}^{(k+1)nm^{k-1}w}_{\mathbb{Z},\chi_5}$.

Note that we set $\chi_2 > \chi_3, \chi_4, \chi_5$ so that we can rely on quantitatively weaker evasive LWE assumption (See Remark 8). We also note that here, we have $\chi_3 = \chi_4 \neq \chi_5$, where Gaussian distributions with different standard deviations are mixed in the precondition distribution. We refer to Remark 7 for the detail.

To show the indistinguishability between the two distributions D_0^1 and D_1^1, we again apply Evasive LWE, this time with respect to matrix $\mathbf{C}_k$ and a sampler Samp^2 as described

below:

$$\begin{aligned}
\mathbf{S}^2 &= \begin{pmatrix} (\mathbf{s}_0,\ldots,\mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}} \otimes \mathbf{r}_{k,1}^\mathsf{T}) \\ \vdots \\ (\mathbf{s}_0,\ldots,\mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}} \otimes \mathbf{r}_{k,Q}^\mathsf{T}) \end{pmatrix} \\
\mathsf{aux}_1^2 &= \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{\mathbf{c}_{k,j}\}_{j\in[Q]} \\
\mathsf{aux}_2^2 &= (\mathbf{x}_0, \{\mathbf{x}_{i,j}, \mathbf{r}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{\mathbf{Y}_{i,j}\}_{i\in[k-2],j\in[Q]}, \{f_j, \mathbf{r}_{k,j}\}_{j\in[Q]}, \\
&\quad \mathbf{A}, \mathbf{B}, \mathbf{C}_1, \ldots, \mathbf{C}_{k-1}, \mathbf{U}) \\
\mathbf{P}^2 &= (\mathbf{C}_{k-1} \otimes \mathbf{r}_{k-1,1}^\mathsf{T} \| \cdots \| \mathbf{C}_{k-1} \otimes \mathbf{r}_{k-1,Q}^\mathsf{T}),
\end{aligned}$$

where $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_{i,j}, \mathbf{c}_{k,j}$ for $i \in [k-1]$, $j \in [Q]$ are as defined in distribution D_0^1. Then again using Lemma 4.7, to prove that the two distributions are computationally indistinguishable, it suffices to prove the computational indistinguishability between the following two distributions:

Distribution D_0^2:

$$\begin{pmatrix} \mathsf{aux}_2^2, & \mathbf{C}_k, & \mathbf{c}_1, & \mathbf{c}_2, \\ \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{d}_{j_1}, \mathbf{d}_{j_1,j_2}\}_{j_1,j_2\in[Q]}, & \{\mathbf{c}_{k,j}\}_{j\in[Q]} \end{pmatrix}$$

Distribution D_1^2:

$$\begin{pmatrix} \mathsf{aux}_2^2, & \mathbf{C}_k, & \mathbf{v}_1, & \mathbf{v}_2, \\ \{\mathbf{v}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{w}_{j_1}, \mathbf{w}_{j_1,j_2}\}_{j_1,j_2\in[Q]}, & \{\mathbf{v}_{k,j}\}_{j\in[Q]} \end{pmatrix},$$

where

$$(\mathbf{d}_{j_1,j_2})_{j_1,j_2\in[Q]} = \mathbf{S}^2\mathbf{P}^2 + (\mathbf{e}'_{j_1,j_2})_{j_1,j_2\in[Q]}, \qquad \mathbf{e}'_{j_1,j_2} \leftarrow \mathcal{D}_{\mathbb{Z},\chi_{5,2}}^{(k+1)nm^{k-2}w},$$

all the $\mathbf{c}$ vectors and $\{\mathbf{d}_{j_1}\}_{j_1}$ are defined same as previously, and $\mathbf{v}$ (resp., $\mathbf{w}$) vectors are sampled uniformly at random from the same domain as their corresponding $\mathbf{c}$ (resp., $\mathbf{d}$) vectors. In the above, $(\mathbf{a}_{j_1,j_2})_{j_1,j_2\in[Q]}$ denotes a matrix obtained by vertically

concatenating vectors $\{\mathbf{a}_{j_1,j_2}\}_{j_1,j_2}$ of the same dimensions for all possible combinations of $j_1, j_2 \in [Q]$. In particular, we have

$$\mathbf{d}_{j_1,j_2} = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}} \otimes \mathbf{r}_{k,j_1}^{\mathsf{T}})(\mathbf{I}_{(k+1)nm^{k-2}} \otimes \mathbf{r}_{k-1,j_2}^{\mathsf{T}})\mathbf{C}_{k-1} + \mathbf{e}'_{j_1,j_2}.$$

To show that the two distributions - D_0^2 and D_1^2 - are computationally indistinguishable, we again apply Evasive LWE, now with respect to matrix $\mathbf{C}_{k-1}$. In general, we apply evasive LWE k times, where the sampler Samp^l for $l \in [k]$ for the l-th application of the evasive LWE assumption is defined as follows: Samp^1 is as defined before.
For $l \in [2, k]$, evasive LWE is applied with respect to the matrix $\mathbf{C}_{k-(l-2)}$ and Samp^l outputs the following:

$$\begin{aligned}
\mathbf{S}^l &= \begin{pmatrix} \vdots \\ (\mathbf{s}_0, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}} \otimes \mathbf{r}_{k,j_1}^{\mathsf{T}})(\mathbf{I}_{(k+1)nm^{k-2}} \otimes \mathbf{r}_{k-1,j_2}^{\mathsf{T}}) \cdots (\mathbf{I}_{(k+1)nm^{k-l+1}} \otimes \mathbf{r}_{k-l+2,j_{l-1}}^{\mathsf{T}}) \\ \vdots \end{pmatrix}_{j_1,\ldots,j_{l-1} \in [Q]} \\
\mathsf{aux}_1^l &= \mathbf{c}_1, \mathbf{c}_2, \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{\mathbf{c}_{k,j}\}_{j\in[Q]}, \{\mathbf{d}_{j_1}, \mathbf{d}_{j_1,j_2}, \ldots, \mathbf{d}_{j_1,\ldots,j_{l-2}}\}_{j_1,\ldots,j_{l-2}\in[Q]} \\
\mathsf{aux}_2^l &= (\mathbf{x}_0, \{\mathbf{x}_{i,j}, \mathbf{r}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{\mathbf{Y}_{i,j}\}_{i\in[k-l],j\in[Q]}, \{f_j, \mathbf{r}_{k,j}\}_{j\in[Q]}, \\
&\quad \mathbf{A}, \mathbf{B}, \{\mathbf{C}_i\}_{i\in[k]\setminus\{k-l+2\}}, \mathbf{U}) \\
\mathbf{P}^l &= (\mathbf{C}_{k-l+1} \otimes \mathbf{r}_{k-l+1,1}^{\mathsf{T}} \| \cdots \| \mathbf{C}_{k-l+1} \otimes \mathbf{r}_{k-l+1,Q}^{\mathsf{T}}),
\end{aligned}$$

where

$$\begin{aligned}
&\mathbf{d}_{j_1,j_2,\ldots,j_t} \\
=\ &(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-1}} \otimes \mathbf{r}_{k,j_1}^{\mathsf{T}}) \cdots (\mathbf{I}_{(k+1)nm^{k-t}} \otimes \mathbf{r}_{k-t+1,j_t}^{\mathsf{T}})\mathbf{C}_{k-t+1} + \mathbf{e}'_{j_1,\ldots,j_t} \\
=\ &(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{k-t}} \otimes \mathbf{r}_{k-t+1,j_t}^{\mathsf{T}} \otimes \mathbf{r}_{k-t+2,j_{t-1}}^{\mathsf{T}} \otimes \cdots \otimes \mathbf{r}_{k,j_1}^{\mathsf{T}})\mathbf{C}_{k-t+1} \\
&+\mathbf{e}'_{j_1,\ldots,j_t}, \text{for } t \in [k]
\end{aligned}$$

where $\mathbf{e}'_{j_1,\ldots,j_t} \leftarrow \mathcal{D}_{\mathbb{Z},\chi_{5,t}}^{(k+1)nm^{k-t}}$ when $t \leq k-1$. When $t = k$, $\mathbf{e}'_{j_1,\ldots,j_k}$ is chosen

as $\mathbf{e}'_{j_1,\ldots,j_k} = (\mathbf{e}'_{0,j_1,\ldots,j_k}, \ldots, \mathbf{e}'_{k,j_1,\ldots,j_k})$, where $\mathbf{e}'_{i,j_1,\ldots,j_k} \leftarrow \mathcal{D}^{m\ell}_{\mathbb{Z},\chi_6}$ for $i \in [0, k-1]$ and $\mathbf{e}'_{k,j_1,\ldots,j_k} \leftarrow \mathcal{D}^{m}_{\mathbb{Z},\chi_7}$. Similarly to the first application of evasive LWE, we set $\chi_5 > \chi_{5,2} > \cdots > \chi_{5,k-1} > \chi_6, \chi_7$ so that we can rely on quantitatively weaker evasive LWE assumption (See Remark 8). We also note that here, we have $\chi_6 \neq \chi_7$ for the final usage of evasive LWE, which means that Gaussian distributions with different standard deviations are mixed in the precondition distribution. We refer to Remark 7 for the detail. Thus, after applying EvLWE l times and using Lemma 4.7, it suffices to prove the indistinguishability between the following two distributions.

Distribution D^l_0:

$$\left(\begin{array}{lll} \mathsf{aux}^l_2, & \mathbf{C}_{k-l+2}, & \mathbf{c}_1, \qquad\qquad \mathbf{c}_2, \\ \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{d}_{j_1}, \mathbf{d}_{j_1,j_2}, \mathbf{d}_{j_1,j_2,j_3}, \ldots, \mathbf{d}_{j_1,j_2,\ldots,j_l}\}_{j_1,\ldots,j_l\in[Q]}, & \{\mathbf{c}_{k,j}\}_{j\in[Q]} \end{array}\right)$$

Distribution D^l_1:

$$\left(\begin{array}{lll} \mathsf{aux}^l_2, & \mathbf{C}_{k-l+2}, & \mathbf{v}_1, \qquad\qquad \mathbf{v}_2, \\ \{\mathbf{v}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{w}_{j_1}, \mathbf{w}_{j_1,j_2}, \mathbf{w}_{j_1,j_2,j_3}, \ldots, \mathbf{w}_{j_1,j_2,\ldots,j_l}\}_{j_1,\ldots,j_l\in[Q]}, & \{\mathbf{v}_{k,j}\}_{j\in[Q]} \end{array}\right),$$

where all the $\mathbf{c}$ and the $\mathbf{d}$ vectors are same as defined previously and $\mathbf{v}$ (resp., $\mathbf{w}$) vectors are sampled uniformly at random from the same domain as their corresponding $\mathbf{c}$ (resp., $\mathbf{d}$) vectors.

In particular, we get that after applying EvLWE k times, it suffices to prove the indistinguishability between the following two distributions:

Distribution $D'_0 = D^k_0$:

$$\left(\begin{array}{lll} \mathsf{aux}'_2, & \mathbf{C}_2, & \mathbf{c}_1, \qquad\qquad \mathbf{c}_2, \\ \{\mathbf{c}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{d}_{j_1}, \mathbf{d}_{j_1,j_2}, \mathbf{d}_{j_1,j_2,j_3}, \ldots, \mathbf{d}_{j_1,j_2,\ldots,j_k}\}_{j_1,\ldots,j_k\in[Q]}, & \{\mathbf{c}_{k,j}\}_{j\in[Q]} \end{array}\right)$$

Distribution $D'_1 = D^k_1$:

$$\left(\begin{array}{cccc} \mathsf{aux}_2', & \mathbf{C}_2, & \mathbf{v}_1, & \mathbf{v}_2, \\ \{\mathbf{v}_{i,j}\}_{i\in[k-1],j\in[Q]}, & \{\mathbf{w}_{j_1}, \mathbf{w}_{j_1,j_2}, \mathbf{w}_{j_1,j_2,j_3}, \ldots, \mathbf{w}_{j_1,j_2,\ldots,j_l}\}_{j_1,\ldots,j_l\in[Q]}, & \{\mathbf{v}_{k,j}\}_{j\in[Q]} \end{array}\right),$$

where $\mathsf{aux}_2' = (\mathbf{x}_0, \{\mathbf{x}_{i,j}, \mathbf{r}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{f_j, \mathbf{r}_{k,j}\}_{j\in[Q]}, \mathbf{A}, \mathbf{B}, \{\mathbf{C}_i\}_{i\in[k]\setminus\{2\}}, \mathbf{U})$. All the $\mathbf{c}$, $\mathbf{d}$, $\mathbf{v}$, and $\mathbf{w}$ vectors are same as defined before.

From the discussion above, to complete the proof of Theorem 4.23, it suffices to prove Lemma 4.24 in the following. ■

Lemma 4.24. *Distributions D_0' and D_1' are computationally indistinguishable under the hardness assumption of* LWE.

Proof. We prove the computational indistinguishability between the two hybrids - D_0' and D_1' via the following hybrids:

G_0 : This is same as the distribution D_0'. For ease of reading and setting up notations, let us list what the adversary can see here. The adversary can see

$$\begin{aligned}
\mathsf{aux}: &= \left(\mathbf{x}_0, \{\mathbf{x}_{i,j}, \mathbf{r}_{i,j}\}_{i\in[k-1],j\in[Q]}, \{f_j, \mathbf{r}_{k,j}\}_{j\in[Q]}, \mathbf{A}, \mathbf{B}, \{\mathbf{C}_i\}_{i\in[k]}, \mathbf{U}\right) \\
\mathbf{c}_1 &= \mathbf{s}((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}) \otimes \mathbf{I}^{\otimes k}) + \mathbf{s}_0(\mathbf{D}_0 \otimes \mathbf{I}^{\otimes k}) + \mathbf{e}_1 \\
\mathbf{c}_2 &= (\mathbf{s},\ \mathbf{s}_0,\ \cdots,\ \mathbf{s}_k)\mathbf{B} + \mathbf{e}_2 \\
\mathbf{c}_{i,j} &= \mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j} \otimes \mathbf{I}) \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)}) \\
&\quad + \mathbf{s}_i(\mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}_{i,j}^\mathsf{T} \otimes \mathbf{I}^{\otimes(k-i)}) + \mathbf{e}_{i,j} \\
&\quad \text{for } i \in [k-1],\ j \in [Q] \\
\mathbf{c}_{k,j} &= \mathbf{s}(\mathbf{A}_{f_j}\mathbf{U} \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T}) + \mathbf{s}_k(\mathbf{D}_k \otimes \mathbf{I}^{\otimes(k-1)} \otimes \mathbf{r}_{k,j}^\mathsf{T}) + \mathbf{e}_{k,j} \\
&\quad \text{for } j \in [Q] \\
\mathbf{d}_{j_t,j_{t+1},\ldots,j_k} &= (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{t-1}} \otimes \mathbf{r}_{t,j_t}^\mathsf{T} \otimes \mathbf{r}_{t+1,j_{t+1}}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_{k,j_k}^\mathsf{T})\mathbf{C}_t + \mathbf{e}'_{j_t,\ldots,j_k}, \\
&\quad \text{for } t \in [k],\ j_t, \ldots j_k \in [Q]
\end{aligned}$$

where we have relabeled the subscripts $j_1, j_2, \ldots$, for making the notation simpler. Note that this can be done without loss of generality. We then observe that

$$\begin{aligned}
&\mathbf{d}_{j_1,j_2,\ldots,j_k} \\
&= (\mathbf{s}_0,\ \mathbf{s}_1,\ \ldots,\ \mathbf{s}_k)(\mathbf{I}_{(k+1)n} \otimes \mathbf{r}_{1,j_1}^\top \otimes \cdots \otimes \mathbf{r}_{k,j_k}^\top)\mathbf{C}_1 + \mathbf{e}'_{j_1,\ldots,j_k} \\
&= (\mathbf{s}_0,\ \mathbf{s}_1,\ \ldots,\ \mathbf{s}_k)\begin{pmatrix} \mathbf{I}_n \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top & & \\ & \ddots & \\ & & \mathbf{I}_n \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top \end{pmatrix}\begin{pmatrix} \mathbf{D}_0 & & \\ & \ddots & \\ & & \mathbf{D}_k \end{pmatrix} + \mathbf{e}'_{j_1,\ldots,j_k} \\
&= \Bigg(\underbrace{\mathbf{s}_0(\mathbf{I}_n \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top)\mathbf{D}_0 + \mathbf{e}'_{0,j_1,\ldots,j_k}}_{:=\mathbf{p}_{0,j_1,\ldots,j_k}}, \ldots \underbrace{\mathbf{s}_k(\mathbf{I}_n \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top)\mathbf{D}_k + \mathbf{e}'_{k,j_1,\ldots,j_k}}_{:=\mathbf{p}_{k,j_1,\ldots,j_k}}\Bigg)
\end{aligned}$$

where $\mathbf{r}_{j_1,\ldots,j_k}^\top = \mathbf{r}_{1,j_1}^\top \otimes \cdots \otimes \mathbf{r}_{k,j_k}^\top$ and $\mathbf{e}'_{j_1,\ldots,j_k} = (\mathbf{e}'_{0,j_1,\ldots,j_k}, \ldots, \mathbf{e}'_{k,j_1,\ldots,j_k})$.

G_1 : In this hybrid, $\mathbf{d}_{j_1,j_2,\ldots,j_k} = \{\mathbf{p}_{i,j_1,\ldots,j_k}\}_{i\in[0,k],j_1,\ldots,j_k\in[Q]}$ is computed differently. Namely, for $j_1, \ldots, j_k \in [Q]$, they are computed as

$$\begin{aligned}
\mathbf{p}_{0,j_1,\ldots,j_k} &= \mathbf{c}_1(\mathbf{I}_{m\ell} \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top) - \underbrace{\left(\mathbf{s}((\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}_m) \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top) + \mathbf{e}'_{0,j_1,\ldots,j_k}\right)}_{:=\mathbf{c}'_{0,j_1,\ldots,j_k}} \\
\mathbf{p}_{i,j_1,\ldots,j_k} &= \mathbf{c}_{i,j_i}(\mathbf{I}_{m\ell} \otimes \mathbf{r}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k}^\top) - \underbrace{\left(\mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}_m) \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top) + \mathbf{e}'_{i,j_1,\ldots,j_k}\right)}_{:=\mathbf{c}'_{i,j_1,\ldots,j_k}} \\
&\quad \text{for } i \in [k-1], \\
\mathbf{p}_{k,j_1,\ldots,j_k} &= \mathbf{c}_{k,j_k}(\mathbf{I}_m \otimes \mathbf{r}_{j_1,\ldots,j_{k-1}}^\top) - \underbrace{\left(\mathbf{s}(\mathbf{A}_{f_{j_k}} \otimes \mathbf{r}_{j_1,\ldots,j_k}^\top) + \mathbf{e}'_{k,j_1,\ldots,j_k}\right)}_{:=\mathbf{c}'_{k,j_1,\ldots,j_k}}
\end{aligned}$$

G_2 : In this hybrid, the challenger samples $\mathbf{d}_{j_t,j_{t+1},\ldots,j_k}$ for $t \geq 2$ differently. Namely, for $t \in [2,k]$ and $j_t, \ldots, j_k \in [Q]$, they are computed as

$$\mathbf{d}_{j_t,j_{t+1},\ldots,j_k}$$

$$= \underbrace{\left((\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{C}_t \otimes \mathbf{I}^{\otimes(k-t+1)}) + \mathbf{e}''_t\right)}_{:=\mathbf{s}'_t}(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k}) + \mathbf{e}'_{j_t,\ldots,j_k}$$

where $\mathbf{e}''_t$ for $t \in [2,k]$ are sampled as $\mathbf{e}''_t \leftarrow D^{(k+1)nm^k w}_{\mathbb{Z},\chi_1}$.

G_3 : In this hybrid, $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{s}'_t$ for $t \in [2,k]$ are replaced with random vectors sampled as $\mathbf{c}_1 \leftarrow \mathbb{Z}_q^{\ell m^{k+1}}$, $\mathbf{c}_2 \leftarrow \mathbb{Z}_q^{(m^{k+1}+(k+1)nm^k)w}$, and $\mathbf{s}'_t \leftarrow \mathbb{Z}_q^{(k+1)nm^k w}$ for $t \in [2,k]$.

G_4 : In this hybrid, the challenger samples $\mathbf{d}_{j_t,j_{t+1},\ldots,j_k}$ for $t \geq 2$ randomly as $\mathbf{d}_{j_t,j_{t+1},\ldots,j_k} \leftarrow \mathbb{Z}_q^{(k+1)nm^{t-1}w}$.

G_5 : In this hybrid, the challenger samples $\mathbf{c}_{i,j}$ for $i \in [k]$, $j \in [Q]$ randomly. Namely, they are sampled as $\mathbf{c}_{i,j} \leftarrow \mathbb{Z}_q^{\ell m^k}$ for $i \in [k-1]$, $j \in [Q]$ and $\mathbf{c}_{k,j} \leftarrow \mathbb{Z}_q^{m^k}$ for $j \in [Q]$. Note that in this hybrid, all the vectors except for $\{\mathbf{d}_{j_1,j_2,\ldots,j_k}\}_{j_1,\ldots,j_k\in[Q]} = \{\mathbf{p}_{i,j_1,\ldots,j_k}\}_{i\in[0,k],j_1,\ldots,j_k\in[Q]}$ are random.

G_6 : In this hybrid, $\mathbf{c}'_{i,j_1,\ldots,j_k}$ for $i \in [0,k], j_1,\ldots,j_k \in [Q]$ are sampled differently. Namely, they are sampled as

$$\begin{aligned}
\mathbf{c}'_{0,j_1,\ldots,j_k} &= \underbrace{\left(\mathbf{s}(\mathbf{I} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_k}) + \mathbf{e}''_{j_1,\ldots,j_k}\right)}_{:=\mathbf{s}'_{j_1,\ldots,j_k}}(\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}) + \mathbf{e}'_{0,j_1,\ldots,j_k} \\
\mathbf{c}'_{i,j_1,\ldots,j_k} &= \underbrace{\left(\mathbf{s}(\mathbf{I} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_k}) + \mathbf{e}''_{j_1,\ldots,j_k}\right)}_{=\mathbf{s}'_{j_1,\ldots,j_k}}(\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}) + \mathbf{e}'_{i,j_1,\ldots,j_k} \\
\mathbf{c}'_{k,j_1,\ldots,j_k} &= \underbrace{\left(\mathbf{s}(\mathbf{I} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_k}) + \mathbf{e}''_{j_1,\ldots,j_k}\right)}_{=\mathbf{s}'_{j_1,\ldots,j_k}}\mathbf{A}_{f_{j_k}} + \mathbf{e}'_{k,j_1,\ldots,j_k}
\end{aligned}$$

for $i \in [k-1]$, $j_1,\ldots,j_k \in [Q]$, where $\mathbf{e}''_{j_1,\ldots,j_k} \leftarrow D^m_{\mathbb{Z},\chi_1}$.

G_7 : In this hybrid, $\mathbf{s}'_{j_1,\ldots,j_k}$ for $j_1,\ldots,j_k \in [Q]$ are replaced with random vectors

sampled as $\mathbf{s}'_{j_1,\ldots,j_k} \leftarrow \mathbb{Z}_q^m$.

G_8 : In this hybrid, $\mathbf{c}'_{k,j_1,\ldots,j_k}$ for $j_1,\ldots,j_k \in [Q]$ are computed differently as

$$\mathbf{c}'_{k,j_1,\ldots,j_k} = \mathbf{c}'_{[0,k-1],j_1,\ldots j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\mathbf{U} + \left(\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{U} + \mathbf{e}'_{k,j_1,\ldots,j_k}\right).$$

where $\mathbf{c}'_{[0,k-1],j_1,\ldots j_k} := (\mathbf{c}'_{0,j_1,\ldots j_k}|\cdots|\mathbf{c}'_{k-1,j_1,\ldots j_k})$ and $\mathbf{x}_{j_1,\ldots j_{k-1}} = (\mathbf{x}_0|\mathbf{x}_{1,j_1}|\cdots|\mathbf{x}_{k-1,j_{k-1}})$

G_9 : In this hybrid, $\mathbf{c}'_{i,j_1,\ldots,j_k}$ for $i \in [0,k], j_1,\ldots,j_k \in [Q]$ are sampled randomly. Namely, for $j_1,\ldots,j_k \in [Q]$, we have $\mathbf{c}'_{i,j_1,\ldots,j_k} \leftarrow \mathbb{Z}_q^{m\ell}$ for $i \in [0,k-1]$ and $\mathbf{c}'_{k,j_1,\ldots,j_k} \leftarrow \mathbb{Z}_q^m$.

It is easy to see that the distribution in G_9 is the same as that of D'_1.

Indistinguishability of hybrids:

We prove the indistinguishability between the hybrid distributions via the following claims.

Claim 4.25. $\mathsf{G}_0 \approx_s \mathsf{G}_1$

Proof. The two hybrids differ only in the error terms in $\{\mathbf{p}_{i,j_1,\ldots,j_k}\}_{i\in[0,k]}$ and are indistinguishable due to the smudging lemma 4.5. We show this for the case of $i \in [k-1]$ here. The case of $i = 0$ and $i = k$ can be shown similarly.

In G_0:

$$\mathbf{p}_{i,j_1,\ldots,j_k} = \mathbf{s}_i(\mathbf{I}_n \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k})\mathbf{D}_i + \mathbf{e}'_{i,j_1,\ldots,j_k}$$

In G_1:

$$\begin{aligned}
\mathbf{p}_{i,j_1,\ldots,j_k} &= \mathbf{c}_{i,j_i}(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k}) - \left(\mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}_m) \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{e}'_{i,j_1,\ldots,j_k}\right) \\
&= \mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}_m) \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{s}_i(\mathbf{D}_i \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{e}_{i,j}(\mathbf{I}_{m\ell} \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k}) \\
&\quad - \left(\mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}_m) \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{e}'_{i,j_1,\ldots,j_k}\right)
\end{aligned}$$

$$= \mathbf{s}_i(\mathbf{D}_i \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_k}) + \underbrace{\mathbf{e}_{i,j}(\mathbf{I}_{m\ell} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k}) - \mathbf{e}'_{i,j_1,\ldots,j_k}}_{:=\text{error}}$$

Clearly, the two hybrids differ only in the error terms. Thus, the indistinguishability follows due to the following:

$$\mathbf{e}'_{i,j_1,\ldots,j_k} \approx_s -\mathbf{e}'_{i,j_1,\ldots,j_k} + \mathbf{e}_{i,j}(\mathbf{I}_{m\ell} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k}).$$

The above is true since the distribution of $-\mathbf{e}_{i,j_1,\ldots,j_k}$ is the same as that of $\mathbf{e}_{i,j_1,\ldots,j_k}$ by the symmetry of the discrete Gaussian distribution and by the sumdging lemma, which is applicable since $\chi_6 \geq (m\gamma)^k \lambda^{\omega(1)} \chi_3$ and we have $\|\mathbf{e}_{i,j}(\mathbf{I}_{m\ell} \otimes \mathbf{r}^{\mathsf{T}}_{j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_k})\|_\infty \leq (m\gamma\,\mathrm{poly}(\lambda))^k \chi_3$. The case of $i = k$ is handled similarly, by using $\chi_7 \geq (m\gamma)^k \lambda^{\omega(1)} \chi_4$.

■

Claim 4.26. $\mathsf{G}_1 \approx_s \mathsf{G}_2$

Proof. The two hybrids differ only in the error term in $\{\mathbf{d}_{j_t,j_{t+1},\ldots,j_k}\}_{t\geq 2, j_t,\ldots,j_k \in [Q]}$ and are indistinguishable due to the smudging lemma. In G_1:

$$\mathbf{d}_{j_t,j_{t+1},\ldots,j_k} = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{t-1}} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k})\mathbf{C}_t + \mathbf{e}'_{j_t,\ldots,j_k}$$

In G_2:

$$\begin{aligned}
&\mathbf{d}_{j_t,j_{t+1},\ldots,j_k} \\
=\ &\left((\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{C}_t \otimes \mathbf{I}^{\otimes(k-t+1)}) + \mathbf{e}''_t\right)(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k}) + \mathbf{e}'_{j_t,\ldots,j_k} \\
=\ &(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{I}_{(k+1)nm^{t-1}} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k})\mathbf{C}_t \\
&\underbrace{+\mathbf{e}'_{j_t,\ldots,j_k} + \mathbf{e}''_t(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k})}_{=\text{error}}
\end{aligned}$$

Clearly, the two hybrids differ only in the error terms. Thus, the indistinguishability follows due to the following:

$$\mathbf{e}'_{j_t,\ldots,j_k} \approx_s \mathbf{e}'_{j_t,\ldots,j_k} + \mathbf{e}''_t(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^{\mathsf{T}}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^{\mathsf{T}}_{k,j_k}).$$

The above is true by the smudging lemma, since we have $\chi_{5,t} \geq (m\gamma)^k \chi_1 \cdot \lambda^{\omega(1)}$ for $t \geq 2$ and $\|\mathbf{e}''_t(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^\mathsf{T}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^\mathsf{T}_{k,j_k}\|_\infty \leq (m\gamma\,\mathrm{poly}(\lambda))^k \chi_1$. ■

Claim 4.27. $\mathsf{G}_2 \approx_c \mathsf{G}_3$ due to LWE.

Proof. Let us write $\mathbf{B}$ as $(\mathbf{B}^\mathsf{T}_U|\mathbf{B}^\mathsf{T}_M|\mathbf{B}^\mathsf{T}_L)^\mathsf{T}$ so that

$$\mathbf{c}_2 = (\mathbf{s}, \mathbf{s}_0, \cdots, \mathbf{s}_k)\mathbf{B} + \mathbf{e}_2 = \mathbf{s}\mathbf{B}_U + (\mathbf{s}_1, \ldots, \mathbf{s}_k)\mathbf{B}_L + (\mathbf{s}_0\mathbf{B}_M + \mathbf{e}_2).$$

We also write $\mathbf{C}_t$ as $\mathbf{C}_t = (\mathbf{C}^\mathsf{T}_{t,U}|\mathbf{C}^\mathsf{T}_{t,L})^\mathsf{T}$ so that

$$\begin{aligned}
\mathbf{s}'_t &= (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{C}_t \otimes \mathbf{I}^{\otimes k-t+1}) + \mathbf{e}''_t \\
&= (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k)\begin{pmatrix} \mathbf{C}_{t,U} \otimes \mathbf{I}^{\otimes k-t+1} \\ \mathbf{C}_{t,L} \otimes \mathbf{I}^{\otimes k-t+1} \end{pmatrix} + \mathbf{e}''_t \\
&= (\mathbf{s}_1, \ldots, \mathbf{s}_k)(\mathbf{C}_{t,L} \otimes \mathbf{I}^{\otimes k-t+1}) + \left(\mathbf{s}_0(\mathbf{C}_{t,U} \otimes \mathbf{I}^{\otimes k-t+1}) + \mathbf{e}''_t\right)
\end{aligned}$$

By Lemma 4.12, we have that $\mathbf{s}_0(\mathbf{D} \otimes \mathbf{I}^{\otimes k}) + \mathbf{e}_1$, $\mathbf{s}_0\mathbf{B}_M + \mathbf{e}_2$, and $\{\mathbf{s}_0(\mathbf{C}_{t,U} \otimes \mathbf{I}^{\otimes k-t+1}) + \mathbf{e}''_t\}_{t\in[2,k]}$ are indistinguishable from random vectors. The claim follows since these terms mask $\mathbf{c}_1$, $\mathbf{c}_2$, and $\{\mathbf{s}'_t\}_{t\in[2,k]}$, respectively. ■

Claim 4.28. $\mathsf{G}_3 \approx_c \mathsf{G}_4$ due to LWE.

Proof. In G_3, $\mathbf{d}_{j_t,j_{t+1},\ldots,j_k}$ is chosen as $\mathbf{s}'_t(\mathbf{I}_{(k+1)nm^{t-1}w} \otimes \mathbf{r}^\mathsf{T}_{t,j_t} \otimes \cdots \otimes \mathbf{r}^\mathsf{T}_{k,j_k}) + \mathbf{e}'_{j_t,\ldots,j_k}$ where $\mathbf{s}'_t$ is chosen uniformly at random for all t. The indistinguishability follows by applying Lemma 4.11 for each $t \in [2, k]$, which is possible since we set $\chi_{5,t} \geq (m\gamma \cdot \lambda^{\omega(1)})^k$. ■

Claim 4.29. $\mathsf{G}_4 \approx_c \mathsf{G}_5$ due to LWE.

Proof. We observe that $\mathbf{c}_{i,j}$ is masked by $\mathbf{v}_{i,j} := \mathbf{s}_i(\mathbf{D}_i \otimes \mathbf{I}^{\otimes(i-1)} \otimes \mathbf{r}^\mathsf{T}_{i,j} \otimes \mathbf{I}^{\otimes(k-i)}) + \mathbf{e}_{i,j}$ for $i \in [k]$, $j \in [Q]$. We show that $\{\mathbf{v}_{i,j}\}_{j\in[Q]}$ is pseudorandom for the case of $i = k$. Other cases can be shown similarly. To show the indistinguishability, we first change the

distribution of $\{\mathbf{v}_{k,j}\}_j$ so that they are sampled as

$$\mathbf{v}_{k,j} = \underbrace{\left(\mathbf{s}_k(\mathbf{D}_k \otimes \mathbf{I}^{\otimes k}) + \mathbf{e}''_k\right)}_{:=\mathbf{s}'_k} \left(\mathbf{I}^{\otimes k} \otimes \mathbf{r}_{k,j}\right) + \mathbf{e}_{k,j}.$$

where $\mathbf{e}''_k \leftarrow \mathcal{D}^{m+1}_{\mathbb{Z},\chi_1}$. We claim that this is statistically indistinguishable from the original distribution. To see this, we observe that

$$\mathbf{v}_{k,j} = \mathbf{s}_k(\mathbf{D}_k \otimes \mathbf{r}_{k,j}) + \underbrace{\mathbf{e}''_k \left(\mathbf{I}^{\otimes k} \otimes \mathbf{r}_{k,j}\right) + \mathbf{e}_{k,j}}_{=\text{error}}$$

and these distributions only differ in the error terms. We have

$$\mathbf{e}_{k,j} \approx_s \mathbf{e}''_k \left(\mathbf{I}^{\otimes k} \otimes \mathbf{r}_{k,j}\right) + \mathbf{e}_{k,j}$$

by the smudging lemma, since we have $\chi_3 \geq (m\gamma)^k \lambda^{\omega(1)} \chi_1$ and $\|\mathbf{e}''_k \left(\mathbf{I}^{\otimes k} \otimes \mathbf{r}_{k,j}\right)\|_\infty \leq (m\gamma\,\mathrm{poly}(\lambda))^k \chi_1$. The case of $i \neq k$ is shown similarly, using $\chi_4 \geq (m\gamma)^k \lambda^{\omega(1)} \chi_1$. We then observe that we can replace $\mathbf{s}'_k$ with a random vector by applying LWE with secret $\mathbf{s}_k$. We then apply LWE once again, now the variant with short public matrix and with the secret $\mathbf{s}'_k$, we can conclude that $\{\mathbf{v}_{k,j}\}_{k,j}$ are indistinguishable from random vectors. ■

Claim 4.30. $\mathsf{G}_5 \approx_s \mathsf{G}_6$

Proof. The two hybrids differ only in the error terms in $\{\mathbf{c}'_{i,j_1,\ldots,j_k}\}_{i\in[0,k],j_1,\ldots,j_k\in[Q]}$ and are indistinguishable due to the smudging lemma. We first show this for the case of $i \in [k-1]$. In G_5:

$$\mathbf{c}'_{i,j_1,\ldots,j_k} = \mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}) \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{e}'_{i,j_1,\ldots,j_k}$$

In G_6:

$$\begin{aligned}
\mathbf{c}'_{i,j_1,\ldots,j_k} &= \left(\mathbf{s}(\mathbf{I}_m \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \mathbf{e}''_{j_1,\ldots,j_k}\right)(\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}) + \mathbf{e}'_{i,j_1,\ldots,j_k} \\
&= \mathbf{s}((\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}) \otimes \mathbf{r}^\mathsf{T}_{j_1,\ldots,j_k}) + \underbrace{\mathbf{e}''_{j_1,\ldots,j_k}(\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}) + \mathbf{e}'_{i,j_1,\ldots,j_k}}_{=\text{error}}
\end{aligned}$$

Clearly, the two hybrids differ only in the error terms. Thus, the indistinguishability follows due to the following:

$$\mathbf{e}'_{i,j_1,\ldots,j_k} \approx_s \mathbf{e}'_{i,j_1,\ldots,j_k} + \mathbf{e}''_{j_1,\ldots,j_k}(\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I})$$

The above is true by the smudging lemma, since we have $\chi_6 \geq m\gamma\chi_1\lambda^{\omega(1)}$ and $\|\mathbf{e}''_{j_1,\ldots,j_k}(\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I})\|_\infty \leq m\gamma \operatorname{poly}(\lambda)$. The case of $i = 0$ is shown in the same manner. The case of $i = k$ is shown similarly, noting that

$$\mathbf{e}'_{k,j_1,\ldots,j_k} \approx_s \mathbf{e}'_{k,j_1,\ldots,j_k} + \mathbf{e}''_{j_1,\ldots,j_k}\mathbf{A}_{f_{j_k}}$$

holds since we have $\chi_7 \geq m\beta\chi_1 \cdot \lambda^{\omega(1)}$ and $\|\mathbf{e}''_{j_1,\ldots,j_k}\mathbf{A}_{f_{j_k}}\|_\infty \leq m\chi_1\|\mathbf{A}_{f_{j_k}}\|_\infty \cdot \operatorname{poly}(\lambda) \leq m\beta\chi_1 \cdot \operatorname{poly}(\lambda)$. ■

Claim 4.31. $\mathsf{G}_6 \approx_c \mathsf{G}_7$

Proof. The indistinguishability follows from LWE by Lemma 4.11, which is applicable since $\chi_1 \geq (m\gamma)^k\lambda^{\omega(1)}$. ■

Claim 4.32. $\mathsf{G}_7 \approx_s \mathsf{G}_8$

Proof. The two hybrids differ only in the error terms in $\mathbf{c}'_{k,j_1,\ldots,j_k}$. The indistinguishability follows from the smudging lemma. In G_7,

$$\mathbf{c}'_{k,j_1,\ldots,j_k} = \mathbf{s}'_{j_1,\ldots,j_k}\mathbf{A}_{f_{j_k}}\mathbf{U} + \mathbf{e}'_{k,j_1,\ldots,j_k}$$

In G_8,

$$\begin{aligned}
\mathbf{c}'_{k,j_1,\ldots,j_k} &= \mathbf{c}'_{[0,k-1],j_1,\ldots j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\mathbf{U} + \left(\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{U} + \mathbf{e}'_{k,j_1,\ldots,j_k}\right) \\
&= \left(\mathbf{s}'_{j_1,\ldots,j_k}(\mathbf{A} - \mathbf{x}_{j_1,\ldots,j_{k-1}} \otimes \mathbf{I}) + \mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}\right)\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\mathbf{U} \\
&\quad + \left(\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{U} + \mathbf{e}'_{k,j_1,\ldots,j_k}\right) \\
&= \mathbf{s}'_{j_1,\ldots,j_k}(\mathbf{A}_{f_{j_k}} - f_{j_k}(\mathbf{x}_{j_1,\ldots,j_{k-1}}) \cdot \mathbf{I})\mathbf{U} +
\end{aligned}$$

$$\begin{aligned}&\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{U}+\mathbf{e}'_{k,j_1,\ldots,j_k}+\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\mathbf{U}\\&=\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{A}_{f_{j_k}}\mathbf{U}+\underbrace{\mathbf{e}'_{k,j_1,\ldots,j_k}+\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\mathbf{U}}_{=\text{error}},\end{aligned}$$

where we define $\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}=(\mathbf{e}'_{0,j_1,\ldots,j_k},\ldots,\mathbf{e}'_{k-1,j_1,\ldots,j_k})$ in the second line and we use $f_{j_k}(\mathbf{x}_{j_1,\ldots,j_{k-1}})=1$ in the last line. Clearly, the two hybrids differ only in the error terms. Thus, the indistinguishability follows due to the following:

$$\mathbf{e}'_{k,j_1,\ldots,j_k}+\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\approx_s\mathbf{e}'_{k,j_1,\ldots,j_k}$$

which is true when $\chi_7\geq m\beta\ell\chi_6\cdot\lambda^{\omega(1)}$, since we have $\|\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k}\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\|_\infty\leq m\beta\ell\chi_6\,\mathrm{poly}(\lambda)$, where $\|\widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}}\|_\infty\leq\beta$. ■

Claim 4.33. $\mathsf{G}_8\approx_c\mathsf{G}_9$

Proof. The indistinguishability between the two hybrids follows from the fact that the following distribution is indistinguishable from random:

$$\mathbf{A},\mathbf{U},\left\{\mathbf{r}_{i,j_i},\mathbf{s}'_{j_1,\ldots,j_k}(\mathbf{A}-\mathbf{x}_{j_1,\ldots,j_k}\otimes\mathbf{I})+\mathbf{e}'_{[0,k-1],j_1,\ldots,j_k},\mathbf{s}'_{j_1,\ldots,j_k}\mathbf{U}+\mathbf{e}'_{k,j_1,\ldots,j_k}\right\}_{i\in[k],j_1,\ldots,j_k\in[Q]}$$

This can be shown by LWE with short public matrix as follows. Here, we change the LWE sample with respect to matrix $(\mathbf{A}-\mathbf{x}_{j_1,\ldots,j_k}\otimes\mathbf{I}|\mathbf{U})$ into random vectors for each combination of $(j_1,\ldots,j_k)$ one by one. To do so, we first use the smudging lemma to see that the distribution of $\mathbf{A}$ and $\mathbf{A}-\mathbf{x}_{j_1,\ldots,j_k}\otimes\mathbf{I}$ are statistically indistinguishable, since each entry of $\mathbf{x}_{j_1,\ldots,j_k}\otimes\mathbf{I}$ is either 0 or 1, while that of $\mathbf{A}$ is chosen from $\mathcal{D}_{\mathbb{Z},\gamma}$ with $\gamma=\lambda^{\omega(1)}$. We then apply the LWE with short public matrix to see that the LWE samples with respect to the secret $\mathbf{s}_{j_1,\ldots,j_k}$ are indistinguishable from the random vectors. ■

This completes the proof of Lemma 4.24. ■

4.7.3 A Construction for P

Here, we discuss the variant of our scheme that can deal with circuits with arbitrary (bounded) polynomial depth. Because the construction is very similar to our construction for NC_1 circuits, we only highlight the difference here.

- We sample the matrices $\mathbf{A}_0, \ldots, \mathbf{A}_{k-1}$ uniformly at random from $\mathbb{Z}_q^{n \times m\ell}$ rather than a Gaussian distribution over $\mathbb{Z}^{m \times m\ell}$.
- We replace $\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}$ in the encryption algorithm with $\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{G}$. Similarly, we also replace $\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{I}$ in $\mathbf{X}_i$ with $\mathbf{A}_i - \mathbf{x}_i \otimes \mathbf{G}$. Accordingly, $\mathbf{s}$ is chosen randomly from $\mathbb{Z}_q^{nm^{k-1}}$ rather than $\mathbb{Z}_q^{m^k}$.
- The low-norm variant of the lattice evaluation algorithms ($\mathsf{EvalF}, \mathsf{EvalFX}$) from Lemma 4.4 used in KeyGen_k and Dec are replaced with those of Lemma 3.2 (i.e., the regular one).
- We use the same parameters as Sec. 4.7.1 except that β is set to be $(2m)^{O(d)}$ reflecting the fact that we replace the homomorphic lattice evaluation algorithm.

The correctness of the scheme can be shown similarly to Sec. 4.7.1. The scheme can be proven secure assuming the strengthening of the tensor LWE assumption defined below.

Assumption 4.34 (Extended Tensor LWE)**.** Let $n, m, q, \ell, Q \in \mathbb{N}$ be parameters, χ and γ be Gaussian distributions, and k be a constant. For all $\mathbf{x}_{j_1,\ldots,j_k} \in \{0,1\}^\ell$ indexed by $j_1, \ldots, j_k \in [Q]$, we have

$$\begin{aligned} & \mathbf{A}, \left\{ \mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_{1,j_1}^\mathsf{T} \otimes \cdots \otimes \mathbf{r}_{k,j_k}^\mathsf{T})(\mathbf{A} - \mathbf{x}_{j_1,\ldots,j_k} \otimes \mathbf{G}) + \mathbf{e}_{j_1,\ldots,j_k}, \mathbf{r}_{i,j_i} \right\}_{i \in [k], j_1,\ldots,j_k \in [Q]} \\ \approx_c \quad & \mathbf{A}, \{\mathbf{c}_{j_1,\ldots,j_k}, \mathbf{r}_{i,j_i}\}_{i \in [k], j_1,\ldots,j_k \in [Q]} \end{aligned}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times \ell m}, \mathbf{s} \leftarrow \mathbb{Z}_q^{m^k n}, \mathbf{e}_{j_1,\ldots,j_k} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\ell m}, \mathbf{r}_{i,j} \leftarrow \mathcal{D}_{\mathbb{Z},\gamma}^{m}, \mathbf{c}_{i,j_i} \leftarrow \mathbb{Z}_q^{\ell m}$ for $i \in [k], j_1, \ldots, j_k \in [Q]$.

Theorem 4.35. *Assuming evasive* LWE *(Assumption 4.6) and extended tensor* LWE *(Assumption 4.34) our k input* MIABE *for* P *satisfies very selective security (Definition 4.2).*

Since the proof is very similar to that of Theorem 4.23 in Sec. 4.7.2, we only provide an overview while highlighting the difference. The first step of the proof is the same as that

of Theorem 4.23, where we invoke the evasive LWE assumption k times to conclude that in order to prove the security of the scheme, it suffices to show the indistinguishability of the two distributions D'_0 and D'_1. These distributions are defined similarly, except that $\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{I}$ and $\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{I}$ are replaced with $\mathbf{A}_0 - \mathbf{x}_0 \otimes \mathbf{G}$ and $\mathbf{A}_i - \mathbf{x}_{i,j_i} \otimes \mathbf{G}$. Then, the indistinguishability is shown by similar sequence of hybrids with the following difference.

- We skip G_6 and G_7 and directly argue that $\mathsf{G}_5 \approx_c \mathsf{G}_8$, where $\mathbf{c}'_{k,j_1,\ldots,j_k}$ is replaced with

  $$\mathbf{c}'_{k,j_1,\ldots,j_k} = \mathbf{c}'_{[0,k-1],j_1,\ldots j_k} \widehat{\mathbf{H}}_{\mathbf{A},f_{j_k},\mathbf{x}_{j_1,\ldots j_{k-1}}} \mathbf{U} + \left(\mathbf{s}(\mathbf{I}_n \otimes \mathbf{r}_{j_1,\ldots,j_k})\mathbf{G}\mathbf{U} + \mathbf{e}'_{k,j_1,\ldots,j_k}\right).$$

  in G_8.

- $\mathsf{G}_8 \approx_c \mathsf{G}_9$ is shown directly from the extension of the evasive LWE assumption above.

■

CHAPTER 5

ATTRIBUTE-BASED MULTI-INPUT FE (AND MORE) FOR ATTRIBUTE-WEIGHTED SUMS

5.1 INTRODUCTION

We continue with the theme of constructing advanced encryption schemes for distributed data where we now focus on constructing functional encryption schemes in multi-party settings. Functional encryption (FE) [SW05; BSW11] is a generalization of public key encryption which enables learning specific functions of encrypted data via "functional" keys and nothing else. In FE, a secret key sk_f is associated with a function f, a ciphertext $\mathsf{ct}_{\mathbf{x}}$ is associated with a message $\mathbf{x}$ and decryption allows to compute $f(\mathbf{x})$ and nothing else.

In this chapter, we build attribute based multi-input functional encryption (AB-MIFE), multi-client functional encryption (MCFE) and dynamic decentralized functional encryption (DDFE) for attribute weighted sums (AWS). We describe these primitives below.

The Attribute-Weighted Sums (AWS) Functionality: The AWS functionality, introduced by Abdalla, Gong, and Wee [AGW20], supports the computation of aggregate statistics on encrypted databases. Concretely, consider a database with N attribute-value pairs $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ where $\mathbf{x}_i$ is a public attribute associated with user i, and $\mathbf{z}_i$ is private. Given a function f, the AWS functionality on input $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ is defined as

$$\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i.$$

Multi-Input FE (MIFE): In MIFE [GGG+14] the input to a function is distributed among multiple (say n) parties. Thus, the i^{th} party encrypts its input $\mathbf{z}_i$ to obtain ct_i and a key authority holding a master secret generates a functional key sk_f and these enable the decryptor to compute $f(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ and nothing else. In *attribute based* MIFE (AB-MIFE) [ACGU20], for some functionality f, an attribute $\mathbf{y}_i$ is associated with a ciphertext for slot i, in addition to an input $\mathbf{z}_i$. The secret key is associated with an access control policy g in addition to the function input $\mathbf{c}$. Decryption first checks if $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = 1$, and if so, it computes the MIFE functionality $f(\{\mathbf{z}_i\}, \mathbf{c})$.

Multi-Client FE (MCFE): MCFE [GGG+14; CDG+18a; CDG+18b] is a generalization of MIFE, where the inputs $\mathbf{z}_i$ are additionally associated with public "labels" L_i and any input can be combined with other inputs only if they share the same label.

Dynamic Decentralized FE (DDFE): DDFE [CDSG+20], as the name suggests, is a decentralized variant of FE, where not only can ciphertexts be generated locally and independently but so can the keys. Thus, DDFE works in a setting where both the data and the authority are decentralized. In DDFE for some functionality f, the setup step is localized and run independently by users, letting them generate their private and public keys individually. During encryption, the set of users with whom a given input or key object should be combined can be chosen dynamically. In more detail, each party can specify the set of parties with which its input may be combined, a label that controls which values should be considered together and the input $\mathbf{z}_i$ itself. Similarly, every user can also generate a key object which specifies the set of parties with which the key may be combined, and a key vector $\mathbf{c}_i$. For decryption, the ciphertexts and keys from the parties who mutually agree to combine their inputs and keys are put together to compute $f(\{\mathbf{z}_i\}_i, \{\mathbf{c}_j\}_j)$.

Prior Work. We summarize the state of the art below.

The AWS Functionality. For the AWS functionality, even the weakest multi-input notion, namely MIFE is not known to the best of our knowledge. We note Abdalla *et al.*[AGW20] did propose a multi-party extension to their FE for AWS. However, this scheme is a much weaker primitive than the standard notion of MCFE (or even MIFE), since this scheme natively only supports a single ciphertext query per slot. To extend it to the setting of multiple queries, the authors make use of non-interactive MPC to enable the parties to obtain a random secret sharing of 0.

In more detail, while their scheme supports labels, the difference from standard MCFE schemes is that in their scheme each party uses a one-time secret key for each encryption instead of long-term encryption key, and the one-time keys are generated via non-interactive MPC run between the parties. Specifically, their scheme consists of five algorithms $(\mathsf{Setup}, \mathsf{OTSKGen}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$, and $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Dec}$ are the same as those in standard MCFE. $\mathsf{OTSKGen}(1^\lambda)$ is a non-interactive protocol where party i obtains one-time secret key otsk_i as the output of the protocol. $\mathsf{Enc}(\mathsf{otsk}_i, \mathbf{x}_i)$ takes otsk_i and a message $\mathbf{x}_i$ and outputs a ciphertext ct_i for party i. Correctness holds, i.e., decrypting a set $\{\mathsf{ct}_i\}_{i\in[n]}$ of ciphertexts with a secret key for f reveals $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, only when the set of ciphertexts are generated under the one-time secret keys $\{\mathsf{otsk}_i\}_{i\in[n]}$ derived from a single running of $\mathsf{OTSKGen}(1^\lambda)$. The one-time secret-key can be used only once for encryption, otherwise security does not hold any more. Thus, this notion is even weaker than the variant of MCFE with one-time labeling restriction [CDG+18a] and in particular, does not imply MIFE.

AB-MIFE, MCFE and DDFE. The best known attribute-based MIFE scheme is for the inner product functionality is by Abdalla *et al.*[ACGU20]. Moreover, in the AB-MIFE construction by Abdalla *et al.*, the ABE attribute $\mathbf{y}_i$[1] associated with the i^{th} slot is fixed

[1] In their notation, the embedding of access control policy and attribute are swapped to the ciphertext-policy setting – thus, for them $\mathbf{y}_i$ is an access control policy.

in the setup phase and must remain the same for all ciphertexts, instead of being chosen dynamically by the encryptor for each encryption. For MCFE [ABG19] as well as DDFE [CDSG+20], the largest achievable function class is linear functions (or inner products), albeit with function hiding [AGT21b].

Multi-Input Attribute Based Encryption. An attribute based MIFE scheme implies a multi-input attribute based encryption scheme as a special case. We recall that in an MIABE scheme, encryptor i encodes a secret message m_i together with an attribute $\mathbf{y}_i$. The function key encodes a circuit g so that decryption outputs $(m_1, \ldots, m_n)$ if $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = 1$. The generalization to multi-input predicate encryption additionally allows hiding the attributes $\mathbf{y}_i$. In this setting, we gave constructions for circuits in NC_1 and P under various lattice-based assumptions as described in Chapters 3 and 4. Additionally, as described before, Francati *et al.*[FFMV23] also provided a multi-input predicate encryption scheme for a conjunction of predicates. Their construction supports the class P and is based on the Learning With Errors problem. Moreover, if the arity of the function is restricted to a constant, their security game also supports user corruption. However, their construction does not support collusions, which is one of the most important and technically challenging aspects of designing attribute based encryption schemes.

5.2 OUR RESULTS

In this work, we significantly extend the reach of multi-input functional encryption schemes by providing the first AB-MIFE, MCFE and DDFE schemes that support the AWS functionality. Our constructions satisfy the standard (selective) indistinguishability based security and rely on the matrix DDH assumption on bilinear groups. We discuss each of these contributions below.

AB-MIFE for AWS: We provide the first attribute based MIFE for the AWS functionality. In our AB-MIFE, each encryptor can choose an attribute $\mathbf{y}_i$ specific to its AWS input

$\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$, the key generator can choose an access control policy g_i along with its AWS function h_i for $i \in [n]$ and decryption computes:

$$f((\mathbf{y}_1, \{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}), \ldots, (\mathbf{y}_n, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}))$$
$$= \begin{cases} \sum_{i\in[n]} \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle & (g_1(\mathbf{y}_1) = \cdots = g_n(\mathbf{y}_n) = 0) \\ \bot & (\text{otherwise}) \end{cases}$$

Here, $\mathbf{y}_i, \mathbf{x}_{i,j}$ are public while $\mathbf{z}_{i,j}$ is private, and g_i, h_i belong to arithmetic branching programs (ABP). We note that the number of slots N_i for $i \in [n]$ can be unbounded, and chosen by the encryptor dynamically.

Connection with Multi-Input Attribute-Based Encryption We observe that this functionality also implies the notion of multi-input attribute-based encryption (MIABE) defined in Chapter 3 for a conjunction of predicates represented as ABP. Thus, MIABE implied from MIFE for AWS supports the functionality $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = \bigwedge(g_i(\mathbf{y}_i) = 0)$, where each g_i is an ABP.

In contrast, the constructions of MIABE in Chapters 3 and 4 support an arbitrary $g \in \mathsf{NC}_1$ but only outputs a fixed message whereas construction in this chapter supports the AWS functionality. Additionally, it also supports a stronger security model which allows user corruption, while in our MIABE constructions this is not possible since each party shares the same master secret key. Further, by applying our MIPE compiler to the results in this chapter, we obtain a multi-input predicate encryption scheme for constant arity, albeit *without* support for user corruptions, due to the design of the compiler.

Comparing with the work of Francati *et al.*[FFMV23]. As discussed before, their MIABE construction supports no collusions and user corruptions only for constant (not polynomial) arity. In contrast, our construction of AB-MIFE supports unbounded

Work	Arity	Corruption	Collusion	Function Class	Assumption
[FFMV23]	Poly	No	No	Conjunctions in P	LWE
[FFMV23]	Constant	Yes	No	Conjunctions in P	LWE
Chapter 3	2	No	Yes	NC_1	Koala and LWE
Chapter 4	Constant	No	Yes	NC_1	evasive LWE
Chapter 4	Constant	No	Yes	P	evasive and tensor LWE
This Chapter (MIABE)	Poly	Yes	Yes	Conjunctions in NC_1	Matrix DDH
This Chapter (MIPE)	Constant	No	Yes	Conjunctions in NC_1	Matrix DDH and LWE

Table 5.1: Comparison with related work in MIABE and MIPE.We consider CPA-1 sided security for the comparison with [FFMV23].

collusions, as well as user corruptions for *polynomial* arity. However, our construction, being based on pairings, only supports the function class NC_1 while they support P. Additionally our construction supports computation of the expressive AWS functionality while theirs just recovers a fixed message (i.e. our scheme is an FE not an ABE). In the setting of MIPE, our construction does not support corruption but does support collusions, while theirs achieves the opposite. Please see Table 5.1 for a detailed comparison.

Multi-Client FE for AWS We construct the first MCFE for Attribute-Weighted Sums, which generalizes MIFE described above. In more detail, each encryptor can choose input $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$ together with a label L_i, the key generator can choose ABPs $\{f_i\}_{i\in[n]}$ and decryption computes:

$$f(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$$

as long as all the L_i are equal. This is the first MCFE that supports a functionality beyond inner products to the best of our knowledge. Moreover, the number of slots N_i allowed to each party i are unbounded, though the number of parties n is bounded – this feature was

not achieved by prior MCFE schemes for inner products as far as we are aware.

Dynamic Decentralized FE for AWS Next, we extend our MCFE to the much more challenging setting of DDFE. For the setting of AWS, the i^{th} encryptor chooses a set of users $\mathcal{U}_{M,i}$ with whom its input may be combined, some label L_i to constrain which values should be considered together, aside from its AWS inputs $\{\mathbf{z}_{i,j}\}_{j\in[N_i]}$ which are private and $\{\mathbf{x}_{i,j}\}_{j\in[N_i]}$ which are public. For key generation, the i^{th} user also chooses a set of users $\mathcal{U}_{K,i}$ and a set of ABPs $\bar{f}_i = \{f_j\}_{j\in\mathcal{U}_{K,i}}$. If all the sets $\mathcal{U}_{M,i}$ and $\mathcal{U}_{K,i}$ match up (to some $\mathcal{U}'_K$) and if the labels in all n ciphertext slots are equal, then decryption computes the AWS functionality. Formally, for $k_i = (\bar{f}_i, \mathcal{U}_{K,i})$ and $m_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i)$, the functionality computes:

$$f'(\{i, k_i\}_{i\in\mathcal{U}'_K}, \{i, m_i\}_{i\in\mathcal{U}'_M}) = \begin{cases} \sum_{i\in\mathcal{U}'_K} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle & \text{if the conditions below are satisfied} \\ \perp & \text{otherwise} \end{cases}$$

The conditions are:

1. $\mathcal{U}'_K = \mathcal{U}'_M$ and $\forall\, i \in \mathcal{U}'_K, \mathcal{U}_{K,i} = \mathcal{U}_{M,i} = \mathcal{U}'_K$.
2. $\forall_{i,i'\in\mathcal{U}'_K}$, $\bar{f}_i = \bar{f}_{i'}$ and $L_i = L_{i'}$.

We summarize prior work in Table 5.2. Please see Appendix 5.A for a more detailed summary. We explain our functionalities in the framework of multi-party FE [AGT21b] in Appendix 5.B.

5.2.1 New Applications

Our attribute-based MIFE enables several new and exciting applications that were not possible before. Let us begin with the example for AWS suggested by [AGW20], of computing the average age of smokers who have lung cancer. In this case, the access control layer on top of the MIFE can capture the willingness of a user to even participate in such a study involving their medical data. For example, perhaps a user is willing to

Work	(Pub, Pri) CT	Key	Functionality
MIFE [AGT22]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \otimes \mathbf{z} \rangle$
AB-MIFE[ACGU20]	$(\bot, \mathbf{z}_i)$	$\{y_i, \mathbf{c}_i\}_{i \in S}$	$\bigwedge_{i \in S} f_i(y_i) \cdot \sum_{i \in S} \langle \mathbf{z}_i, \mathbf{c}_i \rangle$
AB-MIFE, Sec. 5.6	$((\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j), \{\mathbf{z}_{i,j}\}_j)$	$\{g_i, h_i\}_{i \in [n]}$	$\bigwedge_i (g_i(\mathbf{y}_i) = 0) \cdot \sum_{i \in [n]} \sum_{j \in [N_i]} h_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$
MCFE [CDG+18b; ABG19]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$
MCFE, Sec. 5.7	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in [n]}$	$\sum_{i \in [n]} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$
DDFE, [CDSG+20; AGT21a]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$
DDFE, Sec. 5.8	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in S}$	$\sum_{i \in S} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$

Table 5.2: Prior state of the art and our results. We do not consider function hiding or MCFE schemes with only one time labels. Above, we denote $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ or $\mathbf{z} = (\mathbf{z}_i)_{i \in S}$. S is some subset of authorized users for a given key. A function f_i is a monotone span programs fixed in setup. Functions f_i, g_i, h_i are arithmetic branching programs chosen in key generation.

participate in this computation if certain criteria are satisfied, for instance if the study is being performed by doctors with certain specializations. Moreover, these criteria can be different for different users. This is exactly the kind of access control that an ABE system is designed to enforce. The required criteria can be specified by each user using its attribute $\mathbf{y}_i$ while the key holder's input g_i must encode her privileges so that she learns the AWS output only if $g_i(\mathbf{y}_i)$ is satisfied for all $i \in [n]$.

In the context of MIABE, we recall the example of the medical researcher from Chapter 3: a doctor is treating Covid patients and desires to understand the relation between Covid and other medical conditions such as asthma or cancer, each of which is treated at different locations. The records of a given patient are encrypted independently and stored in a central repository, and the doctor can be given a key that filters stored (encrypted) records according to criteria such as condition = 'Covid' and condition = 'asthma' and age group ='60-80' and enables decryption of these. Note that AB-MIFE can already support the conjunction of predicates and suffices to enable the functionality of the above example. Moreover, in addition to supporting decryption of messages as in MIABE, AB-MIFE will even allow computing some aggregates on the private data, something

beyond the capability of MIABE.

For MCFE and DDFE, generalizing inner products to AWS is clearly meaningful – all applications of AWS in the single input setting are meaningful in the setting with multiple users, with the additional expressiveness offered by MCFE and DDFE. For instance, in DDFE, the number of users who can participate in a computation is unbounded and moreover, users can join dynamically – this is useful in real world applications such as the examples involving patients in the lung cancer study or users in the minority group discussed earlier.

5.3 TECHNICAL OVERVIEW

Recap of AGW Our starting point is the functional encryption scheme by Abdalla, Gong, and Wee [AGW20], henceforth AGW, which provides the first construction supporting the AWS functionality for ABP from standard assumptions on bilinear maps. In more detail, the encryptor[2] computes a ciphertext encoding $\{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}$ where N is unbounded, $\mathbf{x}_j$ are public and $\mathbf{z}_j$ are private, the key generator computes a secret key encoding an ABP f and decryption recovers $\sum_{j\in[N]}\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$. At a high level, their construction proceeds in two steps: (i) construct a single slot scheme, i.e. $N = 1$, which supports computation of $\langle f(\mathbf{x}), \mathbf{z}\rangle$, (ii) extend this to support unbounded N by running N instances of the single slot scheme, and cleverly handling leakage and size blowup that occurs along the way. As discussed by AGW, step (i) can be achieved by adapting a framework by Wee [Wee17], and the main conceptual and technical novelty lies in achieving step (ii), especially in supporting unbounded N.

We review step (ii) next, as the ideas herein form the basis of our multi-input constructions. As discussed above, the first idea to handle $N > 1$ is to simply run N instances of the single slot scheme but this evidently does not work, since it would allow the decryptor to learn partial sums $\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$ which are not revealed by the ideal functionality. To

[2]Here we discuss the single input construction of AGW, the multi-input construction is discussed later.

address this leakage, the single slot scheme is extended to handle "randomization offsets", namely to add masking values $w_j r$ to the partial sums, where w_j are sampled randomly by the encryptor such that $\sum w_j = 0$, and r is sampled randomly by the key generator. These masking values hide intermediate partial sums $\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle$, but when the partial sums are added, we recover $\sum_{j \in [N]} \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle$ as desired.

To make the secret key size independent of N, AGW construct a hybrid argument over the N slots, collecting "partial sums" along the way – the details of this technique are not relevant for our purposes. They achieve selective simulation based security from the standard k-linear assumption over bilinear groups.

They then extend this construction to a setting where the N slots can be owned by N independent parties – to enforce the constraint that $\sum_{j \in [N]} w_j = 0$, they make use of a non-interactive MPC protocol where the parties communicate to generate these shares prior to each encryption. While this construction provides a first feasibility result for FE supporting the AWS functionality in the multi-party setting, it falls short of achieving the standard notion of MCFE in many important ways:

1. The MPC step introduces an additional round of interaction prior to each encryption[3] – this violates the primary demand of non-interaction that is placed on FE.

2. The ciphertexts constructed in different "iterations", i.e. generated via different runs of the MPC cannot talk to each other, thus failing to satisfy the main functionality requirement of even an MIFE scheme, which explicitly requires supporting such combinations. In more detail, consider a two slot MIFE scheme, where the first slot ciphertexts encode $\mathbf{x}^j$ for $j \in [Q_1]$, the second slot ciphertexts encode $\mathbf{y}^i$ for $i \in [Q_2]$ and the secret key encodes some function f. Then MIFE explicitly requires that $f(\mathbf{x}^j, \mathbf{y}^i)$ should be computable for any pair j, i. Indeed, the standard notion of MCFE *generalizes* MIFE by additionally supporting labels in each ciphertext that dictate how ciphertexts may be combined. The multi-party scheme of AGW does not imply an MIFE.

3. The security game of the multi-client AGW construction does *not* handle the multi-challenge setting, which is the main technical challenge in any MIFE or MCFE construction. Indeed, handling the multi-challenge setting would *disallow*

[3] This can done in a offline phase, but then places a bound on the number of ciphertexts which can be computed.

the usage of simulation security due to an impossibility result by Boneh, Sahai and Waters [BSW11] – since the AGW constructions satisfy simulation security, any generalization to the multi-input setting must necessarily take a different route.

Thus, the question of even constructing an MIFE for AWS, let alone generalizations to AB-MIFE, MCFE and DDFE, is completely open. We provide an outline of the AGW construction in Figure 5.1.

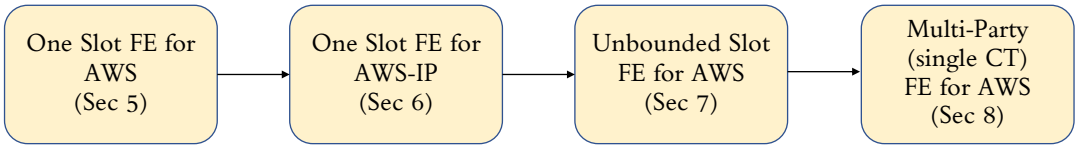

Figure 5.1: Construction Outline of AGW Multi-Client Scheme. All constructions satisfy simulation security. The multi-client scheme only supports a single ciphertext in each slot (and uses MPC to support many).

Building MIFE for AWS In an MIFE for AWS, we have n parties, where the ciphertext computed by the i^{th} party embeds inputs $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$ where N_i is unbounded, the secret key embeds a set of ABPs $\{f_i\}_{i\in[n]}$, and decryption computes

$$f(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$$

Recall that $\mathbf{x}_i$ is public while $\mathbf{z}_i$ is private, i.e., a ciphertext only hides $\mathbf{z}_i$.

A natural idea would be to begin with the multi-party[4] construction of AGW and try to get rid of the MPC. In fact, removing the usage of MPC is not very difficult by using PRFs to compute a secret sharing of 0 for any given label[5], but this would still only lead to a weak variant of MCFE which has the so called "one time label" restriction.

[4]Since the AGW construction does not satisfy the standard definition of MCFE in several important ways as discussed above, we refer to their construction as a multi-party construction, in the sense of [AGT21b].

[5]Consider the 3 party case. Let us say that parties have PRF keys (k_1, k_2), (k_2, k_3), and (k_3, k_1) respectively. Then we can use the fact for every label L, $F(k_1, L) + F(k_2, L), -F(k_2, L) + F(k_3, L), -F(k_3, L) - F(k_1, L)$ are pseudorandom shares of 0 [KDK11; ABG19].

Intuitively, an MCFE with a one time label restriction, as the name suggests, allows each label to be used only one time for each input; this primitive therefore no longer implies MIFE. Handling combinations of multiple ciphertexts is the core functionality of MIFE and forms the basis for most applications, so the one time label restriction is quite a significant limitation. Indeed, in the inner product setting, early constructions of MCFE suffered from the one time label restriction [CDG+18a] and were upgraded to full-fledged MCFE by follow-up work using nontrivial ideas [CDG+18b; ABG19].

Another point to note is the handling of unbounded slots for each client. Concretely, let us say there are n clients, and the i^{th} one chooses N_i (unbounded) internal slots for their data. Now, the AGW multi-party construction can easily handle unbounded N by instantiating $\sum_{i\in[n]} N_i$ nominal clients and having each client internally handle N_i of these. This trick does not work out of the box anymore in the MIFE setting due to the requirement of supporting combinations of all ciphertexts across slots.

Our Approach Taking a step back, a natural approach is to ask whether existing transformations of FE from the single to multi-input setting for the inner product functionality can help us overcome the challenges faced in designing this generalization for AWS. Towards this approach, we observe that all IP-MIFE (or IP-MCFE) schemes in the literature are constructed by (explicitly or implicitly) running an IPFE scheme in parallel for each input and handling leakage issues along the way. At a high level, these works can be classified into two categories based on which property of the underlying IPFE is required in the security proof: 1) ciphertext homomorphism, e.g., [AGRW17; CDG+18b; ACF+18; ABG19] or 2) function-hiding security, e.g., [DOT18; Tom19; AGT21b].

While IPFE schemes have ciphertext homomorphism (a ciphertext is a group element, and adding ciphertexts of $\mathbf{x}_1$ and $\mathbf{x}_2$ results in a ciphertext of $\mathbf{x}_1 + \mathbf{x}_2$), this is not the case in FE for AWS due to public inputs for ABPs. Since ABPs are not linear functions, it is

hopeless to equip an FE scheme for AWS with ciphertext homomorphism. It is worth noting that the reason that the AB-MIFE scheme in [ACGU20] can handle only a limited form of access control, i.e., only secret keys are associated with attributes, and access control is done between the attributes and the public fixed policy, comes from the fact that their scheme relies on the former approach and cannot associate ciphertexts with attributes or a policy as the case of the single input AB-FE schemes.

Fortunately, the latter approach is not ruled out, and indeed, we show that it can be made to work even for the AWS functionality. We observe that works in the latter category use function-hiding security of the underlying scheme to obtain function-hiding in the resultant IP-MIFE scheme. In this work, however, we will use function-hiding for a completely different purpose – to transform a singe-input scheme into a multi-input scheme *without* relying on ciphertext homomorphism of the underlying scheme. In particular, we will not achieve function-hiding security in our final MIFE for AWS scheme.

Recap: Construction of IP-MIFE from IPFE Our starting point is therefore the FE to MIFE transformation for inner products by Datta, Okamoto and Tomida [DOT18] (henceforth DOT), which we describe next. Recall that IP-MIFE supports functions $f : (\mathbb{Z}_p^d)^n \to G_T$ specified by $(\mathbf{c}_1, \ldots, \mathbf{c}_n) \in (\mathbb{Z}_p^d)^n$ and defined as $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = [\sum_{i\in[n]} \langle \mathbf{x}_i, \mathbf{c}_i \rangle]_T$. While the DOT IP-MIFE scheme is a direct construction based on pairings, it can be viewed as a generic construction from a function-hiding FE scheme for inner product (IPFE) as described in the next paragraph. Recall that in an IPFE scheme, the ciphertext and secret key are associated with $\mathbf{x} \in \mathbb{Z}_p^d$ and $\mathbf{c} \in \mathbb{Z}_p^d$ respectively, and decryption reveals $[\langle \mathbf{x}, \mathbf{c} \rangle]_T$. The function-hiding property guarantees that the secret key hides $\mathbf{c}$ along with hiding $\mathbf{x}$ in the ciphertext.

Let iFE = (iSetup, iEnc, iKeyGen, iDec) be a function-hiding IPFE scheme. Then the IP-MIFE scheme is constructed as follows. Setup generates master secret keys

$\mathsf{iMSK}_1, \ldots, \mathsf{iMSK}_n \leftarrow \mathsf{iSetup}(1^\lambda)$ and sets $\mathsf{ek}_i = \mathsf{iMSK}_i, \mathsf{msk} = \{\mathsf{iMSK}_i\}_{i\in[n]}$. Encryption of $\mathbf{x}_i$ for slot i computes $\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_i, (\mathbf{x}_i, 1))$ and outputs $\mathsf{ct}_i = \mathsf{iCT}_i$. Key generation, given input $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$, randomly chooses $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$ such that $\sum_{i\in[n]} r_i = 0$, computes $\mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, (\mathbf{c}_i, r_i))$ for $i \in [n]$, and outputs $\mathsf{sk} = \{\mathsf{iSK}_i\}_{i\in[n]}$. Decryption outputs $\sum_{i\in[n]} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_i) = [\sum\langle \mathbf{x}_i, \mathbf{c}_i\rangle]_T$, since $\sum r_i = 0$. Here, the random element r_i is used to hide partial decryption values $\langle \mathbf{x}_i, \mathbf{c}_i\rangle$.

Let us now turn our attention to the security proof. Since we need neither function hiding nor adaptive security for the multi-input scheme in our purpose, we can make the proof much simpler than that by DOT as follows. We will denote $\mathsf{iEnc}(\mathsf{iMSK}_i, \mathbf{v})$ and $\mathsf{iKeyGen}(\mathsf{iMSK}_i, \mathbf{v})$ by $\mathsf{iCT}_i[\mathbf{v}]$ and $\mathsf{iSK}_i[\mathbf{v}]$, respectively. Now, in the original game, the adversary is given $\mathsf{iCT}_i[(\mathbf{x}_i^{j,\beta}, 1)]$ for the j-th challenge message $(\mathbf{x}_i^{j,0}, \mathbf{x}_i^{j,1})$ and $\{\mathsf{iSK}_i[(\mathbf{c}_i^\ell, r_i^\ell)]\}_{i\in[n]}$ for the ℓ-th secret key of $(\mathbf{c}_1^\ell, \ldots, \mathbf{c}_n^\ell)$, where β is the challenge bit. Thus, the goal of the proof is to delete the information of β from the ciphertexts in an indistinguishable manner. In what follows, we omit index ℓ for conciseness since all secret keys can be handled in the same manner.

The security proof uses two hybrids. In the first hybrid, the j-th ciphertext for slot i is changed to $\mathsf{iCT}_i[(\mathbf{x}_i^{j,0}, 1)]$ while all secret keys are changed to $\{\mathsf{iSK}_i[(\mathbf{c}_i, r_i + \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i\rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i\rangle)]\}_{i\in[n]}$ for *all* i, j. The indistinguishability of the original game and the first hybrid follows from the security of the function-hiding IPFE scheme and the following constraint:

$$\langle \mathbf{x}_i^{j,\beta}, \mathbf{c}_i\rangle - \langle \mathbf{x}_i^{j,0}, \mathbf{c}_i\rangle = \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i\rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i\rangle \quad \text{for all } i, j \tag{5.1}$$

which follows from the fact that the adversary can inherently learn $\langle \mathbf{x}_i^{j,\beta}, \mathbf{c}_i\rangle - \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i\rangle$ from challenge queries (originally observed in [AGRW17, page 4]). In the second hybrid, all secret keys are changed to $\{\mathsf{iSK}_i[(\mathbf{c}_i, r_i)]\}_{i\in[n]}$ for all i, which readily follows from

the fact that the two distributions are equivalent:

$$\{(r_1, \ldots, r_n) : r'_1, \ldots, r'_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r'_i = 0, \quad r_i = r'_i + \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle\}$$
$$\{(r_1, \ldots, r_n) : r_1, \ldots, r_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r_i = 0\}$$

This is because we have that $\sum_{i\in[n]}(\langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle) = 0$ due to the admissibility condition on the queries. At this point, the advantage of the adversary is 0 since its view contains no information about β.

Generalizing the FE to MIFE to support AWS Next, we show how to generalize the FE to MIFE compiler of DOT to handle the AWS functionality. In this step, we make use of an insight developed by AGW to handle unbounded slots, namely, to leverage a (single input) FE scheme that supports *unbounded-slot* AWS together with randomization offsets. In more detail, a ciphertext is associated with $(\mathbf{v}, \mathbf{p}) \in \mathcal{X} \times \mathbb{Z}_p^m$, a secret key is associated with $(F, \mathbf{q}) \in \mathcal{F} \times \mathbb{Z}_p^m$, and decryption reveals $[F(\mathbf{v}) + \langle \mathbf{p}, \mathbf{q} \rangle]_T$. Here, we assume that $\mathcal{F}$ is a set of functions belong to unbounded-slot AWS and $\mathcal{X}$ is its input space, but observe that the argument below can be applied to any function classes. For security, we require that both $\mathbf{p}, \mathbf{q}$ are hidden. In what follows, we call this functionality AWS with inner product (AWSw/IP).

We emphasize that while this is also the functionality achieved by AGW [AGW20[Sec 6]], the security achieved by these is quite different: our construction must satisfy partially function hiding *indistinguishability based* security, while theirs satisfies simulation based security without function hiding. Additionally, our construction will support the additional inner product functionality with respect to unbounded-slot AWS, while AGW support it for only single-slot AWS.

Suppose we have a partially function-hiding FE scheme for the AWSw/IP functionality, denoted as $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$. Then, we can construct MIFE

that supports functions $F : (\mathcal{X})^n \to G_T$ specified by $(F_1, \ldots, F_n) \in \mathcal{F}^n$ and defined as $F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = [\sum_{i\in[n]} F_i(\mathbf{v}_i)]_T$ from aFE by following the template of DOT, as described next. Looking ahead, F can be instantiated to capture either AWS or attribute based AWS to obtain MIFE for AWS or AB-MIFE for AWS respectively. For instance, for MIFE, we set $\mathbf{v}_i = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$, $F = (f_1, \ldots, f_n)$ where f_i are ABPs, and $F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$.

Construction 5.1 (MIFE for AWS)**.** $\mathsf{Setup}(1^\lambda)$ It outputs

$\mathsf{ek}_i = \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and $\mathsf{msk} = \{\mathsf{aMSK}_i\}_i$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{v}_i)$ It outputs $\mathsf{ct}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}_i, 1))$.

$\mathsf{KeyGen}(\mathsf{msk}, (F_1, \ldots, F_n))$ It outputs $\mathsf{sk} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$ s.t. $\sum r_i = 0$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (F_i, r_i))$.

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It outputs $\sum_{i\in[n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum F_i(\mathbf{v}_i)]_T$.

The security proof is essentially the same as in the case of IP-MIFE, discussed above. We use the following two hybrids: in the first hybrid, the j-th ciphertext for slot i is changed from $\mathsf{aCT}_i[(\mathbf{v}^{j,\beta}, 1)]$ to $\mathsf{aCT}_i[(\mathbf{v}^{j,0}, 1)]$ while all secret keys are changed from $\{\mathsf{aSK}_i[(F_i, r_i)]\}_{i\in[n]}$ to $\{\mathsf{aSK}_i[(F_i, r_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0}))]\}_{i\in[n]}$. In this step, we leverage the important observation that a constraint similar to Eq.(5.1) holds in MIFE for the function class we consider, where the final output is the summation of the output of each slot. Specifically, we have $F_i(\mathbf{v}_i^{j,\beta}) - F_i(\mathbf{v}_i^{j,0}) = F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})$ for all i, j. Hence we can use the function-hiding security of aFE to change the second element of the function in secret keys from r_i to $r_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})$ in a indistinguishable manner. In the second hybrid, we bring back all secret keys to the form $\{\mathsf{aSK}_i[(F_i, r_i)]\}_{i\in[n]}$. This transition is possible as the case of IP-MIFE, that is, we use the fact that the following distributions are equivalent:

$$\{(r_1, \ldots, r_n) : r'_1, \ldots, r'_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r'_i = 0, \quad r_i = r'_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})\}$$

$$\{(r_1, \ldots, r_n) : r_1, \ldots, r_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r_i = 0\}$$

which follows from the query condition $\sum_{i\in[n]}(F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})) = 0$. At this point, the advantage of the adversary is 0.

Partial Function Hiding FE for AWS with Inner Product It remains to construct the single input, unbounded slot FE scheme for the AWSw/IP functionality which satisfies partial function hiding. As discussed, the AGW scheme achieves simulation-based security but not function hiding. Our idea of extending AGW to function-hiding to the multi-challenge setting is to design AGW using a function-hiding IPFE scheme, which is inspired by the constructions of ABE for ABP and FE for AWS from (slotted) function-hiding IPFE in [LL20a; DP21].

Recall that AGW first constructs a one-slot scheme that can handle randomization offsets, the construction of which basically follows the ABE scheme by [Wee17], and then converts it to an unbounded-slot scheme in a modular manner. The spirit of our construction follows their blueprint, that is, we first construct a function-hiding one-slot scheme that can handle randomization offsets using a function-hiding IPFE scheme, and then convert it to a unbounded-slot scheme. However, we present the unbounded construction directly since later we will need to extend this to attribute based FE for AWSw/IP, and the modular construction does not apply to that setting. To see this, note that in an attribute-based FE for AWSw/IP, an attribute is associated with an unbounded-slot message and how to deconstruct the attribute for a one-slot message is unclear.

The key building block of [LL20a; DP21] is the arithmetic key garbling scheme (AKGS), which is specialized for constructing attribute-based encryption schemes. In our work, however, we use the (extended) partially-garbling scheme (PGS) for ABP [IW14; AGW20] together with function-hiding IPFE since PGS is more suitable for FE that computes

ABPs and AWSs directly. Informally, it uses an algorithm $\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t})$ that takes an ABP $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_1}$, a public string $\mathbf{x} \in \mathbb{Z}_p^{n_0}$, private strings $\mathbf{z} \in \mathbb{Z}_p^{n_1}$ and $\delta \in \mathbb{Z}_p$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$, and outputs

$$\mathbf{L} = (\langle \mathbf{L}_1 \mathbf{t}, \mathbf{x}' \rangle + \delta, \langle \mathbf{L}_2 \mathbf{t}, \mathbf{x}' \rangle, \ldots, , \langle \mathbf{L}_s \mathbf{t}, \mathbf{x}' \rangle, \mathbf{z}[1] + \langle \mathbf{L}_{s+1} \mathbf{t}, \mathbf{x}' \rangle, \ldots, \mathbf{z}[n_1] + \langle \mathbf{L}_t \mathbf{t}, \mathbf{x}' \rangle)$$

where $\mathbf{x}' = (\mathbf{x}, 1)$, and $s, t \in \mathbb{N}, \mathbf{L}_i \in \mathbb{Z}_p^{(n_0+1)\times(t-1)}$ are deterministically computed from f. The algorithm pgb satisfies:

Corrrectness we can efficiently compute a vector $\mathbf{b}_{f,\mathbf{x}} \in \mathbb{Z}_p^t$ from $f, \mathbf{x}$ such that $\langle \mathbf{L}, \mathbf{b}_{f,\mathbf{x}} \rangle = \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta$;

Security we can efficiently simulate the distribution of $\mathbf{L}$ over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$ from $(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta)$.

Then, we can construct FE for $\mathsf{AWSw/IP}$ as follows. Let iFE be a function-hiding IPFE scheme as above.

Construction 5.2 (FE for $\mathsf{AWSw/IP}$)**.** $\mathsf{Setup}(1^\lambda)$ It outputs $(\mathsf{pp}, \mathsf{msk}) = (\mathsf{iPP}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$.

$\mathsf{Enc}(\mathsf{msk}, (\{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}, \mathbf{p}))$ It chooses $u_1, \ldots, u_N, w_1, \ldots, w_N \leftarrow \mathbb{Z}_p$ s.t. $\sum_{j\in[N]} w_j = 0$. It defines

$$\mathbf{X}_j = \begin{cases} (u_j \mathbf{x}'_j, \mathbf{z}_j, w_j, \mathbf{p}, \ 0^\rho) & (j = 1) \\ (u_j \mathbf{x}'_j, \mathbf{z}_j, w_j, 0^m, 0^\rho) & (j > 1) \end{cases}$$

and computes $\mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, \mathbf{X}_j)$ for $j \in [N]$. It outputs $\mathsf{ct} = (\{\mathbf{x}_j, \mathsf{iCT}_j\}_j)$. Note that the last ρ entries are used only for the security proof.

$\mathsf{KeyGen}(\mathsf{msk}, (f, \mathbf{q}))$ It chooses $r \leftarrow \mathbb{Z}_p$, $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$ and computes $\mathbf{L}_1, \ldots, \mathbf{L}_t$ from f.

It defines

$$\mathbf{Y}_j = \begin{cases} (\mathbf{L}_j\mathbf{t}, 0^{n_1}, \ \ r, \ \mathbf{q}, \ \ 0^{\rho}) & (j = 1) \\ (\mathbf{L}_j\mathbf{t}, 0^{n_1}, \ \ 0, 0^{m}, 0^{\rho}) & (1 < j \leq s) \\ (\mathbf{L}_j\mathbf{t}, \mathbf{e}_{j-s}, 0, 0^{m}, 0^{\rho}) & (s < j \leq t) \end{cases} . \tag{5.2}$$

where $\mathbf{e}_i$ is one-hot vector with the i-th element being 1. Finally it computes $\mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, \mathbf{Y}_j)$ for $j \in [t]$ and outputs $\mathsf{sk} = (f, \{\mathsf{iSK}_j\}_j)$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$ It computes $[d_{j,\ell}]_T = \mathsf{iDec}(\mathsf{iCT}_j, \mathsf{iSK}_\ell)$ for all $j \in [N], \ell \in [t]$ and $\mathbf{b}_{f,\mathbf{x}}$ describe above. It outputs $\sum_{j\in[N]} \sum_{\ell\in[t]} [\mathbf{b}_{f,\mathbf{x}}[\ell] \cdot d_{j,\ell}]_T$.

In decryption, it follows that

$$(d_{j,1}, \ldots, d_{j,t}) = \begin{cases} \mathsf{pgb}(f, \mathbf{x}_j, \mathbf{z}_j, rw_j + \langle \mathbf{p}, \mathbf{q} \rangle; u_j\mathbf{t}) & (j = 1) \\ \mathsf{pgb}(f, \mathbf{x}_j, \mathbf{z}_j, rw_j; u_j\mathbf{t}) & (j > 1) \end{cases}$$

and thus $\sum_{j\in[N]} \sum_{\ell\in[t]} \mathbf{b}_{f,\mathbf{x}}[\ell] \cdot d_{j,\ell} = \sum_{j\in[N]} (\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + rw_j + (j = 1)\langle \mathbf{p}, \mathbf{q} \rangle) = \sum_{j\in[N]} \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle$. Roughly speaking, the partially function-hiding security of this scheme follows from the following observations:

- Thanks to the function-hiding property of iFE, what the adversary can learn from ct and sk is $\{[(d_{j,1}, \ldots, d_{j,t})]_T\}_{j\in[N]}$.
- The random tape $\{[u_j\mathbf{t}]_T\}_{j\in[N]}$ used to compute $\{d_{j,\ell}\}$ looks random under the SXDH assumption, and $(d_{j,1}, \ldots, d_{j,t})$ for each j appear to be generated by a fresh random tape.
- Thanks to the security of the PGS, the only information about $(\{\mathbf{z}_j\}, \mathbf{p}, \mathbf{q})$ contained in $(d_{j,1}, \ldots, d_{j,t})$ is $\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + rw_j + (j = 1)\langle \mathbf{p}, \mathbf{q} \rangle$.
- Under the SXDH assumption, $\{[rw_j]_T\}_{j\in[N]}$ looks random with the constraint that the summation of these is $[0]_T$. Thus, the only information about $(\{\mathbf{z}_j\}, \mathbf{p}, \mathbf{q})$ in $\{[(d_{j,1}, \ldots, d_{j,t})]_T\}_{j\in[N]}$ is $\sum\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle$.

We remark that we can easily modify the scheme such that Enc and KeyGen take vectors $\mathbf{p}$ and $\mathbf{q}$ as a vector of group elements. We will use this property later in the overview for DDFE for AWS.

Attribute-Based MIFE for AWS Next, we explain how to make the above MIFE for AWS construction attribute-based. At a high level, we do the following: (i) Make FE for AWSw/IP Attribute-Based, (ii) Use the FE to MIFE compiler discussed above to "lift" this to AB-MIFE for AWS. We expand on these below.

Step 1: Make FE for AWSw/IP Attribute-Based We extend FE for AWSw/IP such that it incorporates an ABP predicate which controls decryption, similarly to attribute-based encryption. Specifically, we add a public vector $\mathbf{y}$ to the message in the ciphertext and a public ABP g to the function in the secret key, and allow decryption only when $g(\mathbf{y}) = 0$.

A naive idea is to define $\mathbf{x}'_j = (\mathbf{y}, \mathbf{x}_j), \mathbf{z}'_j = (v, \mathbf{z}_j)$ and $f'(\mathbf{x}') = (a \cdot g(\mathbf{y}), f(\mathbf{x}))$ where $a, v \leftarrow \mathbb{Z}_p$ and use $\{\mathbf{x}'_j, \mathbf{z}'_j\}$ and f' as inputs for encryption and key generation of FE for AWSw/IP, respectively. Note that f' is an ABP if f, g in turn are ABPs. Then, decryption outputs $[\sum_{j\in[N]}(av \cdot g(\mathbf{y}) + \langle f(\mathbf{x}_j), \mathbf{z}_j\rangle) + \langle \mathbf{p}, \mathbf{q}\rangle]_T$. Since $[av \cdot g(\mathbf{y})]_T$ looks random if $g(\mathbf{y}) \neq 0$ under the SXDH assumption, the decryptor can learn $[\sum_{j\in[N]}\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle + \langle \mathbf{p}, \mathbf{q}\rangle]_T$ only when $g(\mathbf{y}) = 0$.

However, this idea does not work since a needs to be provided in the clear in the secret key for decryption, and this disallows the reliance on the SXDH assumption (recall that we need f and $\mathbf{x}$ to compute $\mathbf{b}_{f,\mathbf{x}}$ in the decryption of the FE for AWSw/IP). To avoid this, we directly embed a in $\mathbf{Y}_{s+1}$ so that we can perform decryption without the knowledge of a. Concretely, we define f' as $f'(\mathbf{x}') = (g(\mathbf{y}), f(\mathbf{x}))$ instead of $f'(\mathbf{x}') = (a \cdot g(\mathbf{y}), f(\mathbf{x}))$ and define $\mathbf{Y}_{s+1} = (\mathbf{L}_{s+1}, \underline{a}\mathbf{e}_1, 0, 0^m, 0^\rho)$ in Eq. (5.2). Then, the decryption result is the same as the naive construction, which follows from the correctness of the PGS, but f' does not contain information about a in this construction.

The proof of function-hiding security of this scheme is inspired from the proof in AGW [AGW20, Section 7], but with the following key differences: 1) we need to prove IND-based function-hiding security in the secret-key multi-challenge setting while AGW

proves SIM-based security in the public-key setting (thus not function-hiding); 2) Our scheme is attribute-based while AGW is not. Hence we need to handle secret key queries that cannot decrypt some challenge ciphertexts, and for such ciphertexts the function values in $\beta = 0$ and $\beta = 1$ can be different (β is the challenge bit). Please see Section 5.5.2 for details.

Step 2: AB-MIFE for AWS Suppose we have an FE scheme aFE where a ciphertext is associated with $(c, \mathbf{v}, \mathbf{p})$ while a secret key is associated with $(k, F, \mathbf{q})$, and decryption reveals $[F(\mathbf{v}) + \langle \mathbf{p}, \mathbf{q} \rangle]_T$ if $\mathsf{P}(c, k) = 1$ for some predicate P and $\perp$ otherwise. We also assume that aFE is partially function-hiding, so that the ciphertext hides (a part of) $\mathbf{v}$ and $\mathbf{p}$, and the secret key hides $\mathbf{q}$.

At first glance, it seems that we can construct AB-MIFE for $\mathcal{F}'$ from aFE by using Construction 5.1, where $\mathcal{F}'$ consists of functions $F' : (C \times \mathcal{X})^n \to G_T$ specified by $((k_1, F_1), \ldots, (k_n, F_n)) \in (\mathcal{K} \times \mathcal{F})^n$ and defined as

$$F'((c_1, \mathbf{v}_1), \ldots, (c_n, \mathbf{v}_n)) = \begin{cases} [\sum_{i \in [n]} F_i(\mathbf{v}_i)]_T & \mathsf{P}(c_i, k_i) = 1 \text{ for all } i \\ \perp & \text{otherwise} \end{cases}$$

Note that AB-MIFE for AWS corresponds to the case where $c = \mathbf{y}, \mathbf{v} = \{\mathbf{x}_j, \mathbf{z}_j\}_{j \in [N]}$, $k = g$ where g is an ABP, $\mathsf{P}(c, k) = 1$ iff $g(\mathbf{y}) = 0$, and F is specified by an ABP f and defined as $F(\mathbf{v}) = \sum \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle$. We can also observe that aFE for the above setting corresponds to attribute-based FE for AWSw/IP.

However the above construction is insufficient due to the following reason. Let us consider a two-input scheme where an adversary obtains ciphertexts of $(c_1^1, \mathbf{v}_1^1)$ and $(c_1^2, \mathbf{v}_1^2)$ for slot 1 (denoted by $\mathsf{ct}_1^1, \mathsf{ct}_1^2$), a ciphertext of $(c_2^1, \mathbf{v}_2^1)$ for slot 2 (denoted by ct_2^1), and a secret key for $((k_1, F_1), (k_2, F_2))$ (denoted by sk) such that $\mathsf{P}(c_1^j, k_1) = 1$ for both $j \in \{1, 2\}$ while $\mathsf{P}(c_2^1, k_2) = 0$. Note that ct_i^j denotes the j-th ciphertext for slot i. In this case, the adversary should not obtain any information about private inputs, since the predicate

of slot 2 is never satisfied. However, the adversary can learn $[F_1(\mathbf{v}_1^2) - F_1(\mathbf{v}_1^1)]_T$ in this construction (recall that it can learn $[F_1(\mathbf{v}_1^1) + r_1]_T$ and $[F_1(\mathbf{v}_1^2) + r_1]_T$ by decryption of aFE in slot 1). This is leakage which we need to avoid.

An important fact is that this leakage is inherent if the adversary additionally obtains a ciphertext of $(c_2^2, \mathbf{v}_2^2)$ for slot 2 (denoted by ct_2^2) such that $\mathsf{P}(c_2^2, k_2) = 1$. This is because it can learn $[F_1(\mathbf{v}_1^2) - F_1(\mathbf{v}_1^1)]_T$ by $\mathsf{Dec}(\mathsf{ct}_1^2, \mathsf{ct}_2^2, \mathsf{sk}) - \mathsf{Dec}(\mathsf{ct}_1^1, \mathsf{ct}_2^2, \mathsf{sk})$. By generalizing this observation, it turns out that such leakage appears only when the adversary obtains an illegitimate secret key, which cannot decrypt any combinations of ciphertexts that the adversary has. More formally, we say a secret key for $((k_1, F_1), \ldots, (k_n, F_n))$ is illegitimate if there exists slot i and the adversary does not have a ciphertext of $(c_i, *)$ for slot i such that $\mathsf{P}(c_i, k_i) = 1$. In other words, the above construction is secure in the model where the adversary never asks for illegitimate secret keys – we refer to this notion as security against legitimate keys.

Achieving Security against Any Keys We next show how to remove this restriction and achieve security against any keys starting with a scheme secure against legitimate keys. Our idea is to encrypt all secret keys and allow the adversary to decrypt only legitimate secret keys. We achieve such a construction by leveraging an n-out-of-n secret sharing scheme and an attribute-based encryption scheme ABE for the *dual predicate* of P, denoted by $\overline{\mathsf{P}}$. Note that $\overline{\mathsf{P}} : \mathcal{K} \times C \to \{0, 1\}$ is defined as $\overline{\mathsf{P}}(k, c) = 1 \Leftrightarrow \mathsf{P}(c, k) = 1$. We describe this conversion next.

Let $\mathsf{wmFE} = (\mathsf{wmSetup}, \mathsf{wmEnc}, \mathsf{wmKeyGen}, \mathsf{wmDec})$ be an $\mathsf{AB\text{-}MIFE}$ scheme for $\mathcal{F}'$ secure against legitimate keys. The setup algorithm generates n master secret keys $\mathsf{abMSK}_1, \ldots, \mathsf{abMSK}_n$ of ABE and sets $(\mathsf{abMSK}_i, \mathsf{wmEK}_i)$ as an encryption key for slot i. Encryption of $(c_i, \mathbf{v}_i)$ for slot i is the same as wmEnc except that it appends a secret key of ABE for c_i to wmCT_i. Key generation of $\{k_i, F_i\}$ runs $\mathsf{wmSK} \leftarrow \mathsf{wmKeyGen}(\mathsf{wmMSK}, \{k_i, F_i\})$, secret shares wmSK to $\sigma_1, \ldots, \sigma_n$, encrypts σ_i with

attribute k_i to abCT_i by ABE, and outputs $\{\mathsf{abCT}_i\}$. In this construction, observe that the adversary cannot obtain illegitimate secret keys. Recall that in $\mathsf{AB\text{-}MIFE}$ for AWS, an ABE scheme for $\overline{\mathsf{P}}$ corresponds to ciphertext-policy ABE (CP-ABE) for ABPs, which was recently proposed by Lin and Luo [LL20b].

We observe that this security against legitimate vs. any keys in the context of $\mathsf{AB\text{-}MIFE}$ can be seen as generalization of security against complete vs. incomplete (or zero vs. multiple) queries in the context of MIFE [AGRW17]. Recall that incomplete queries refers to the case where an adversary does not make a ciphertext query for some inputs. Therefore, in the context of plain MIFE, all secret keys become illegitimate if the adversary makes incomplete queries and legitimate otherwise. On the other hand, in the context of $\mathsf{AB\text{-}MIFE}$, whether each secret key become legitimate or illegitimate crucially depends on which attributes are queried in both ciphertext and secret-key queries, and thus the situation is much more complex. This is why we need an advanced primitive, namely, ABE to upgrade the security of $\mathsf{AB\text{-}MIFE}$ while MIFE secure against complete queries can be upgraded to that secure against incomplete queries using only symmetric key encryption.

Security under Corruptions The above transformation works only in the secret-key setting where the adversary cannot corrupt encryption keys. Intuitively, this comes from the fact that there exist ABPs that never evaluates to 0 (we call such ABPs null ABPs). For the transformation to work in the corruption model, we require the underlying CP-ABE scheme to have the property that the adversary cannot decrypt ciphertexts for null ABPs even if it obtains the master secret key. However, in the only known CP-ABE scheme by [LL20b], the master secret key has the ability to decrypt ciphertexts for null ABPs. Actually, such a CP-ABE scheme implies witness encryption for NP relations verifiable in NC1, and it seems quite challenging to obtain one from standard assumptions.

To circumvent this problem, we introduce wildcards for access-controlling functionality

similarly to [FFMV23]. That is, for the wildcard input $\star$ and all ABPs (including null ABPs) g, we always have $g(\star) = 0$. In this functionality, the adversary that corrupts i-th input can admissibly generate a ciphertext for slot i that satisfies the i-th predicate for all secret keys, and the leakage of the master secret key of the CP-ABE scheme due to corruption does not cause the problem. As observed in [FFMV23], multi-input ABE with corruptions implies witness encryption, but their constructions does not imply it due to the use of wildcards. Ours also does not imply it for the same reason.

To allow our AB-MIFE scheme to support wildcards, the underlying AB-FE scheme for AWSw/IP also needs to have the wildcard functionality. This modification is quite simple: just setting $v = 0$ (see step 1 above) in encryption with the wildcard attribute suffices.

Comparison with [NPP22] Very recently, Nguyen, Phan and Pointcheval proposed an attribute-based MCFE scheme for inner product (see Table 5.3 for precise functionality). Their scheme is in the weaker MCFE model where each label can be used only once per input, and does not imply standard MIFE for the same function class. In [NPP22, Remark 16], they informally state that we can apply 1) the technique in [CDG+18b] to convert their scheme into the MCFE in the stronger notion and 2) All-or-Nothing Encapsulation [CDSG+20] to achieve security against incomplete queries. We believe that both claims are false.

Regarding item 1, as we discussed previously, the technique in [CDG+18b] to remove the one-time restriction requires ciphertext homomorphism of the underlying scheme. However, their underlying single client scheme is not ciphertext homomorphic, and thus how to use the technique in [CDG+18b] is unclear. Regarding item 2, as discussed above, in the context of the *attribute-based* setting, All-or-Nothing Encapsulation would be insufficient to achieve full-fledged security in the AB-MIFE setting, which can handle only the issue of complete vs. incomplete queries in the non-attribute-based setting. Hence, their result does not appear to imply AB-MIFE scheme as claimed.

Multi-Client FE for AWS To construct MCFE for AWS, we follow the blueprint by [AGT21b] where they construct an MCFE scheme for inner products from a function-hiding IPFE scheme. Roughly speaking, we replace the function-hiding IPFE scheme in their scheme with our FE scheme for AWSw/IP. However, following this approach leads to obstacles in the security proof, to handle which, we need to modify their blueprint. To see this, first consider the MCFE construction for AWS that is obtained by applying their blueprint straightforwardly to our setting. Let $H : \{0,1\}^* \to G_1$ be a hash function and aFE be a FE scheme for AWSw/IP. The scheme is given as follows:

Construction 5.3 (Candidate MCFE for AWS)**.** $\mathsf{Setup}(1^\lambda)$ It outputs $\mathsf{ek}_i = \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and $\mathsf{msk} = \{\mathsf{aMSK}_i\}_i$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{v}_i, L)$ It outputs $\mathsf{ct}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}_i, [(v_L, 0)]_1))$ where $[v_L]_1 = H(L)$.

$\mathsf{KeyGen}(\mathsf{msk}, (F_1, \ldots, F_n))$ It outputs $\mathsf{sk} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$ s.t. $\sum r_i = 0$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (F_i, [(r_i, 0)]_2))$.

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It outputs $\sum_{i\in[n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum F_i(\mathbf{v}_i)]_T$.

Let us try to prove the security of this MCFE candidate similarly to Construction 5.1. In what follows, we denote $\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}, [\mathbf{p}]_1))$ and $\mathsf{aKeyGen}(\mathsf{aMSK}_i, (F, [\mathbf{q}]_2))$ by $\mathsf{aCT}_i[\mathbf{v}, \mathbf{p}]$ and $\mathsf{aSK}_i[F, \mathbf{q}]$, respectively. To see why the security proof does not work in this construction, considering the simple case suffices where an adversary queries only one challenge ciphertext for each slot after which it makes secret-key queries adaptively. In the original game, the adversary is given $\mathsf{aCT}_i[\mathbf{v}_i^\beta, (v_L, 0)]$ for a challenge message $(\mathbf{v}_i^0, \mathbf{v}_i^1, L)$ and $\{\mathsf{aSK}_i[F_i, (r_i, 0)]\}_{i\in[n]}$ for a secret key of $(F_1, \ldots, F_n)$. In the first hybrid, the ciphertext for slot i is changed to $\mathsf{aCT}_i[\mathbf{v}^0, (0, 1)]$ while all secret keys are changed to $\{\mathsf{aSK}_i[F_i, (r_i, v_L r_i + F_i(\mathbf{v}_i^\beta) - F_i(\mathbf{v}_i^0))]\}_{i\in[n]}$. The indistinguishability of the original game and the hybrid follows from the partially function-hiding security of aFE. The

next step will be to change $[v_L r_i]_2$ to $[\widetilde{r_i}]_2$ where $\widetilde{r_i}$ is a random element in $\mathbb{Z}_p$ such that $\sum \widetilde{r_i} = 0$. If we can show this indistinguishability, $\widetilde{r_i}$ absorbs the term $F_i(\mathbf{v}_i^\beta) - F_i(\mathbf{v}_i^0)$ and we can conclude the proof, but this is not the case. This is because the adversary can compute $[v_L]_1$ by the hash function, and thus we cannot use the SXDH assumption in G_2.

We solve this by modifying Construction 5.3 as follows: Let $\mathsf{PRF} : \{0,1\}^* \to \mathbb{Z}_p$ be a pseudorandom function with a key space $\mathcal{K}$.

Construction 5.4 (MCFE for AWS)**.** $\mathsf{Setup}(1^\lambda)$ It chooses $\mathsf{K}_{i,j} \leftarrow \mathcal{K}$ for $i, j \in [n], i < j$, and sets $\mathsf{K}_{i,j} = \mathsf{K}_{j,i}$ for $j < i$. It outputs $\mathsf{ek}_i = (\mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda), \{\mathsf{K}_{i,j}\}_{i \neq j})$ for $i \in [n]$ and $\mathsf{msk} = \{\mathsf{aMSK}_i\}_i$.

$\mathsf{Enc}(\mathsf{ek}_i, x_i, L)$ It outputs $\mathsf{ct}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (x, [(v_{L,i}, 0)]_1))$ where $v_{L,i} = \sum_{j \in [n] \setminus \{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L)$.

$\mathsf{KeyGen}(\mathsf{msk}, (f_1, \ldots, f_n))$ It outputs $\mathsf{sk} = \{\mathsf{aSK}_i\}_{i \in [n]}$ where $r \leftarrow \mathbb{Z}_p$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (f_i, [(r, 0)]_2))$.

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It outputs $\sum_{i \in [n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum f_i(x_i)]_T$.

This construction is inspired by the MIFE scheme in [AGRW17], in which the randomizing term v_i in ciphertexts are generated in the setup phase (not by a PRF). Note that this is enough for MIFE, but in our case we extend the technique to generate the term $v_{L,i}$ for each label on the fly via PRF to handle an exponentially large number of labels. Observe that $\sum_{i \in [n]} v_{L,i} = 0$ for all L and correctness holds. Such a usage of PRF in MCFE was first introduced by [ABG19], but again their MCFE construction requires ciphertext homomorphism and is not applicable to AWS functionality. This is why we devise a new construction based on DOT combining ideas from [AGRW17; ABG19].

In this construction, the above proof strategy works. At a high level, this is due to the

following reasons:

- $\{v_{L,i}\}$ looks random with the constraint $\sum v_{L,i} = 0$ for each label if the PRF is secure.
- In contrast to Construction 5.3, the adversary cannot compute $v_{L,i}$ publicly.

In fact, Construction 5.4 is a secure MCFE scheme for AWS. For a detailed description, we refer the reader to Section 5.7.

Dynamic Decentralized FE for AWS Given an MCFE scheme for AWS, we now convert it to DDFE using the template provided by [AGT21b]. The high-level idea of the blueprint is to allow parties in the system to generate an independent MCFE instance in Construction 5.4 for each user set $\mathcal{U}$ by using a PRF on the fly. First, each party joins the system dynamically by generating a key K_i of a pseudorandom function (PRF) as a master secret key. In encryption and key generation for party set $\mathcal{U}$, party $i \in \mathcal{U}$ generates $\mathsf{aMSK}_{i,\mathcal{U}} = \mathsf{aSetup}(1^\lambda; \mathsf{PRF}^{\mathsf{K}_i}(\mathcal{U}))$, which is unique to $(i, \mathcal{U})$. For key generation of $(\{f_i\}, \mathcal{U})$, party i computes a common random element $r_{i,\mathcal{U}} = H(\{f_i\}, \mathcal{U})$ by a hash function and outputs $\mathsf{aSK}_{i,\mathcal{U}}$ as KeyGen in Construction 5.4. In encryption of $(x_i, \mathcal{U}, L)$, party i generates $\mathsf{K}_{i,j}$ via non-interactive key exchange, and outputs $\mathsf{aCT}_{i,\mathcal{U}}$ in the same manner as Enc in Construction 5.4. Observe that $\mathsf{aCT}_{i,\mathcal{U}}/\{\mathsf{aSK}_{i,\mathcal{U}}\}_{i\in\mathcal{U}}$ is a valid ciphertext/secret key of the MCFE scheme in Construction 5.4 with respect to $\mathcal{U}$. For more details, please see Section 5.8.

We provide an overview of our constructions in Figure 5.2.

Organisation of the chapter. The rest of the chapter is organised as follows. We provide the necessary preliminaries in Section 5.4. In Section 5.5, we provide construction for attribute based FE for AWS with IP, which is used as a building block in our other constructions. In Section 5.6, we provide our construction for AB-MIFE for AWS. We construct MCFE for AWS in Section 5.7. In Section 5.8, we provide our construction for DDFE for AWS. In Appendix 5.B, we define different primitives studied in this chapter

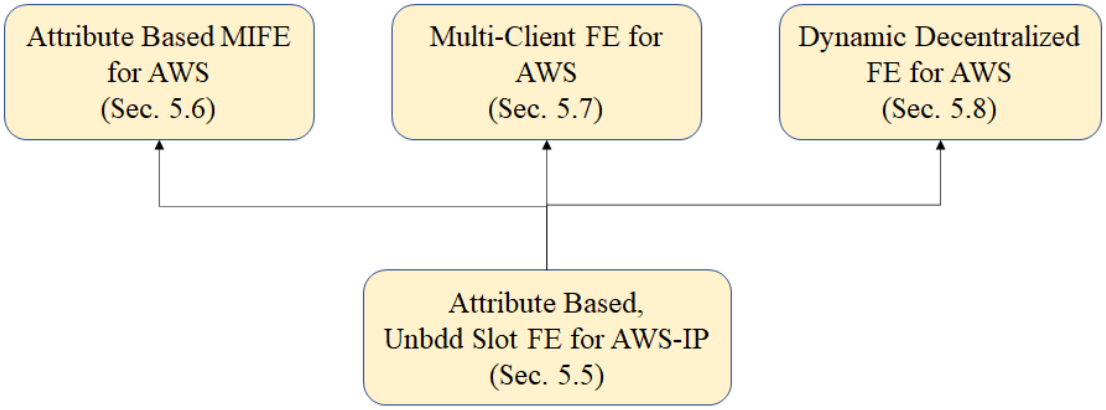

Figure 5.2: Outline of our constructions. For the implication to MCFE and DDFE, the underlying construction for AWSw/IP need not be attribute based.

in terms of multi-party FE.

5.4 PRELIMINARIES

Notation used in this chapter Any vector $\mathbf{v}$ is by default a column vector and $\mathbf{v}^\top$ is a row vector. For vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ denotes the vector concatenation as row vectors *regardless of* whether each $\mathbf{v}_i$ is a row or column vector. For a matrix $\mathbf{A} = (a_{j,\ell})_{j,\ell}$ over $\mathbb{Z}_p$, $[\mathbf{A}]_i$ denotes a matrix over G_i whose (j,ℓ)-th entry is $g_i^{a_{j,\ell}}$, and we use this notation for vectors and scalars similarly. We use addition for the group operation in every group in bilinear groups. For vectors $\mathbf{a} \in \mathbb{Z}_p^n$ and $\mathbf{b} \in G^n$ where G is a cyclic group of order p, we abuse the notation of inner product and denote $\sum_{i\in[n]} \mathbf{a}[i][\mathbf{b}[i]]$ by $\langle \mathbf{a}, [\mathbf{b}] \rangle$. For a matrix $\mathbf{M} \in \mathbb{Z}_p^{a\times b}$ and vectors $\mathbf{a} \in \mathbb{Z}_p^a, \mathbf{b} \in \mathbb{Z}_p^b$, we denote a vector $\mathbf{m}$ such that $\langle \mathbf{a} \otimes \mathbf{b}, \mathbf{m} \rangle = \mathbf{a}^\top \mathbf{M} \mathbf{b}$ by $\mathbf{t}(\mathbf{M})$. By 0^n, for $n \in \mathbb{N}$, we represent the vector $(0, \ldots, 0) \in \mathbb{Z}^n$.

5.4.1 Computation Models

Definition 5.1 (Arithmetic Branching Programs (ABPs))**.** An arithmetic branching program $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$ is defined by a prime p, a directed acyclic graph (V, E), two special vertices $v_0, v_1 \in V$, and a labeling function $\sigma : E \to \mathcal{F}^{\mathsf{Affine}}$, where $\mathcal{F}^{\mathsf{Affine}}$

consists of all affine functions $g : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$. The size of f is the number of vertices $|V|$. Given an input $\mathbf{x} \in \mathbb{Z}_p^{n_0}$ to the ABP, we can assign a $\mathbb{Z}_p$ element to edge $e \in E$ by $\sigma(e)(\mathbf{x})$. Let P be the set of all paths from v_0 to v_1. Each element in P can be represented by a subset of E. The output of the ABP on input $\mathbf{x}$ is defined as $\sum_{E' \in P} \prod_{e \in E'} \sigma(e)(\mathbf{x})$. We can extend the definition of ABPs for functions $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_1}$ by evaluating each output in a coordinate-wise manner and denote such a function class by $\mathcal{F}_{n_0,n_1}^{\mathsf{ABP}}$.

Note that we can convert any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs are a stronger computational model than all of the above.

5.4.2 Basic Cryptographic Notions

Definition 5.2 (Bilinear Groups)**.** Let $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of bilinear groups. Bilinear groups $\mathbb{G}_\lambda$=$(p, G_1, G_2, G_T, g_1, g_2, e)$ are specified by a prime p, cyclic groups G_1, G_2, G_T of order p, generators g_1 and g_2 of G_1 and G_2 respectively, and a bilinear map $e : G_1 \times G_2 \to G_T$, which has two properties.

- (Bilinearity): $\forall h_1 \in G_1, h_2 \in G_2, a, b \in \mathbb{Z}_p, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- (Non-degeneracy): For g_1 and g_2, $g_T = e(g_1, g_2)$ is a generator of G_T.

In what follows, we omit the index λ from $\mathbb{G}_\lambda$ and abuse notation by denoting a family of bilinear groups $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ also by $\mathbb{G}$ if it is clear in the context.

Definition 5.3 ($\mathcal{D}_{j,k}$-MDDH Assumption [EHK+17])**.** Let $\{\mathbb{G}\}$ be a family of bilinear groups. For $j > k$, let $\mathcal{D}_{j,k}$ be a matrix distribution over matrices in $\mathbb{Z}_p^{j \times k}$, which outputs a full-rank matrix with overwhelming probability. We can assume that, wlog, the first k rows of a matrix chosen from $\mathcal{D}_{j,k}$ form an invertible matrix. We consider the following distribution: $\mathbf{A} \leftarrow \mathcal{D}_{j,k}$, $\mathbf{m} \leftarrow \mathbb{Z}_p^k$, $\mathbf{k}_0 = \mathbf{A}\mathbf{m}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^j$, $P_{i,\beta} = (\mathbb{G}, [\mathbf{A}]_i, [\mathbf{k}_\beta]_i)$. We say that the $\mathcal{D}_{j,k}$-MDDH assumption holds with respect to $\{\mathbb{G}\}$ if, for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_{j,k}\text{-}\mathsf{MDDH}} = \max_{i \in \{1,2\}} |\Pr[1 \leftarrow \mathcal{A}(P_{i,0})] - \Pr[1 \leftarrow \mathcal{A}(P_{i,1})]| \leq \mathrm{negl}\,.$$

In what follows, we denote $\mathcal{D}_{k+1,k}$ by $\mathcal{D}_k$. Note that the well-known k-Lin assumption

can be captured as the $\mathcal{D}_k$-MDDH assumption.

Uniform Distribution. Let $\mathcal{U}_{j,k}$ be a uniform distribution over $\mathbb{Z}_p^{j\times k}$. Then, the following holds with tight reductions: $\mathcal{D}_k$-MDDH $\Rightarrow$ $\mathcal{U}_k$-MDDH $\Rightarrow$ $\mathcal{U}_{j,k}$-MDDH. We denote $\mathcal{D}_k$-MDDH by MDDH$_k$.

Random Self-Reducibility. We can obtain arbitrarily many instances of the $\mathcal{D}_{j,k}$-MDDH problem from a single instance. For any $n \in \mathbb{N}$, we define the following distribution: $\mathbf{A} \leftarrow \mathcal{D}_{j,k},\ \mathbf{M} \leftarrow \mathbb{Z}_p^{k\times n},\ \mathbf{K}_0 = \mathbf{AM},\ \mathbf{K}_1 \leftarrow \mathbb{Z}_p^{j\times n},\ P_{i,\beta} = (\mathbb{G}, [\mathbf{A}]_i, [\mathbf{K}_\beta]_i)$. The n-fold $\mathcal{D}_{j,k}$-MDDH assumption is similarly defined to the $\mathcal{D}_{j,k}$-MDDH assumption. Then, the n-fold $\mathcal{D}_{j,k}$-MDDH assumption is implied by the $\mathcal{D}_{j,k}$-MDDH assumption with security loss of $\min\{n, j-k\}$.

Definition 5.4 (Secret Sharing Scheme)**.** A (n out of n) secret sharing scheme consists of Share and Rec.

$\mathsf{Share}(s, n)$ It takes a secret $s \in \{0,1\}^m$ and a number of shares n and outputs shares $\sigma_1, \ldots, \sigma_n \in \{0,1\}^m$.

$\mathsf{Rec}(\sigma_1, \ldots, \sigma_n)$ It takes shares $\sigma_1, \ldots, \sigma_n \in \{0,1\}^m$ and outputs a bit string s'.
A secret sharing scheme has two properties.

Correctness For all $n, m \in \mathbb{N}, s \in \{0,1\}^m$,

$$\Pr[\mathsf{Rec}(\sigma_1, \ldots, \sigma_n) = s : \sigma_1, \ldots, \sigma_n \leftarrow \mathsf{Share}(s, n)] = 1.$$

Security For all $n, m \in \mathbb{N}, s \in \{0,1\}^m, S \subsetneq [n]$, the following distributions are identical:

$$\{\{\sigma_i\}_{i\in S} : \sigma_1, \ldots, \sigma_n \leftarrow \mathsf{Share}(s, n)\} \quad \text{and} \quad \{\{\sigma_i\}_{i\in S} : \sigma_1, \ldots, \sigma_n \leftarrow \{0,1\}^m\}$$

Definition 5.5 (Non-interactive key exchange (NIKE))**.** A NIKE scheme for shared key space $\mathcal{K}$ consists of the three algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ It takes a security parameter 1^λ and outputs a public parameter pp.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$ It takes pp and outputs a public key pk and the corresponding secret key sk.

$\mathsf{SharedKey}(\mathsf{pk}, \mathsf{sk}) \to \mathsf{K}$ It takes pk and sk and deterministically outputs a shared key $\mathsf{K} \in \mathcal{K}$.

Correctness. A NIKE scheme is correct if, for all $\lambda \in \mathbb{N}$, we have

$$\Pr\left[\mathsf{K}_{i,j} = \mathsf{K}_{j,i} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathsf{K}_{i,j} = \mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j) \\ \mathsf{K}_{j,i} = \mathsf{SharedKey}(\mathsf{pk}_j, \mathsf{sk}_i) \end{array}\right] = 1.$$

Security. We say a NIKE scheme is IND-secure if, for all stateful PPT adversaries $\mathcal{A}$, we have

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\},\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathcal{S} \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathcal{CS}, (i', j') \leftarrow \mathcal{A}(\{\mathsf{pk}_i\}_{i \in \mathcal{S}}) \text{ where } i', j' \in \mathcal{S} \backslash \mathcal{CS} \text{ and } i' \neq j' \\ \mathsf{K}^0_{i',j'} = \mathsf{SharedKey}(\mathsf{pk}_{i'}, \mathsf{sk}_{j'}),\ \mathsf{K}^1_{i',j'} \leftarrow \mathcal{K} \\ \beta' \leftarrow \mathcal{A}(\{\mathsf{sk}_i\}_{i \in \mathcal{CS}}, \mathsf{K}^\beta_{i',j'}) \end{array}\right] \leq 1/2 + \mathrm{negl}\,.$$

Definition 5.6 (Partial Garbling Scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$)**.** We use the following partial garbling scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ [IW14] (please see Definition 5.1) for the construction of our FE schemes. A partial garbling scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ consists of the four algorithms. Note that lgen and rec are deterministic algorithms while pgb and pgb^* are probabilistic algorithms.

$\mathsf{lgen}(f)$ It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ and outputs $\mathbf{L}_1, \ldots, \mathbf{L}_t \in \mathbb{Z}_p^{(n_0+1)\times(t-1)}$ where t depends on f.

$\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t})$ Let $\mathbf{x}'^\top = (\mathbf{x}, 1)$. It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$. It then outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_s \mathbf{t}, \mathbf{z}[1] + \mathbf{x}'^\top \mathbf{L}_{s+1} \mathbf{t}, \ldots, \mathbf{z}[n_1] + \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $s = t - n_1$ and $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

$\mathsf{pgb}^*(f, \mathbf{x}, \mu; \mathbf{t})$ It takes $\mu \in \mathbb{Z}_p$ and $f, \mathbf{x}, \mathbf{t}$ as above and outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t} + \mu, \mathbf{x}'^\top \mathbf{L}_2 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

$\mathsf{rec}(f, \mathbf{x})$ It takes $f, \mathbf{x} \in \mathbb{Z}_p^{n_0}$ and outputs $\mathbf{d}_{f,\mathbf{x}} \in \mathbb{Z}_p^t$.
The concrete description of $\mathsf{lgen}, \mathsf{rec}$ that satisfy the following properties is found in [AGW20, Appendix A]. We slightly modify the format of the output of lgen from [AGW20] for convenience in our construction, but note that they are essentially the same.

Correctness. The garbling scheme is correct if for all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}, \mathbf{t} \in \mathbb{Z}_p^{t-1}$, we have

$$\langle \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}), \mathsf{rec}(f, \mathbf{x}) \rangle = \langle f(\mathbf{x}), \mathbf{z} \rangle.$$

Security. The garbling scheme is secure if for all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}$, the following distributions are statistically close:

$$\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) \quad \text{and} \quad \mathsf{pgb}^*(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z} \rangle; \mathbf{t})$$

where the random tape is chosen over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Extension of Partial Garbling Scheme. We can construct an additional partial garbling algorithm pgb^+ with the following properties [AGW20, Appendix A.

$\mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t})$ Let $\mathbf{x}'^\top = (\mathbf{x}^\top, 1)$. It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$, $\mathbf{x} \in \mathbb{Z}_p^{n_0}$, $\mathbf{z} \in \mathbb{Z}_p^{n_1}$, $\delta \in \mathbb{Z}_p$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$. It then outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t} + \boxed{\delta}, \mathbf{x}'^\top \mathbf{L}_2 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_s \mathbf{t}, \mathbf{z}[1] + \mathbf{x}'^\top \mathbf{L}_{s+1} \mathbf{t}, \ldots, \mathbf{z}[n_1] + \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $s = t - n_1$ and $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

Correctness. For all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$, $\mathbf{x} \in \mathbb{Z}_p^{n_0}$, $\mathbf{z} \in \mathbb{Z}_p^{n_1}$, $\mathbf{t} \in \mathbb{Z}_p^{t-1}$, we have

$$\langle \mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}), \mathsf{rec}(f, \mathbf{x}) \rangle = \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta.$$

Security. For all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$, $\mathbf{x} \in \mathbb{Z}_p^{n_0}$, $\mathbf{z} \in \mathbb{Z}_p^{n_1}$, the following distributions are statistically close:

$$\mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}) \quad \text{and} \quad \mathsf{pgb}^*(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta; \mathbf{t})$$

where the random tape is chosen over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Linearlity. Observe that pgb^+ is affine in $\mathbf{z}[1]$, $\mathbf{t}$, δ, and pgb^* is affine in μ.

5.4.3 Variants of Functional Encryption

Definition 5.7 (Attribute-Based Encryption (ABE))**.** Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a predicate where $\mathcal{X}$ and $\mathcal{Y}$ denote ciphertext-attribute and key-attribute spaces. An attribute-based encryption (ABE) scheme for a predicate family P consists of four algorithms:

$\mathsf{Setup}(1^\lambda)$ It takes a security parameter 1^λ, and outputs a public key pk and a master secret key msk. The other algorithms implicitly take pk.

$\mathsf{Enc}(x, M)$ It takes pk, an attribute $x \in \mathcal{X}$ and a message $M \in \mathcal{M}$ as inputs, and outputs a ciphertext ct. (Note that we let $\mathcal{M}$ be specified in pk.)

$\mathsf{KeyGen}(\mathsf{msk}, y)$ It takes pk, msk, and an attribute $y \in \mathcal{Y}$ as inputs, and outputs a secret key sk.

$\mathsf{Dec}(\mathsf{ct}_x, \mathsf{sk}_y)$ It takes pk, ct and sk as inputs, and outputs a message M' or a symbol $\perp$.

Correctness. An ABE scheme is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 1$, and $M \in \mathcal{M}$, we have

$$\Pr\left[M = M' : \begin{array}{l} (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \kappa) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(x, M) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, y) \\ M' = \mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) \end{array}\right] = 1.$$

Security. An ABE scheme is selectively secure in the multi-challenge setting if it satisfies the following condition. That is, the advantage of $\mathcal{A}$ defined as follows is negligible in λ for all stateful PPT adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \kappa) \\ \{x^j, M^k_0, M^j_1\}_{j \in [q_c]} \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathsf{ct}^j \leftarrow \mathsf{Enc}(x^j, M^j_\beta) \ \text{ for } j \in [q_c] \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\{\mathsf{ct}^j\}_{j \in [q_c]}) \end{array}\right] - \frac{1}{2}$$

where all $\{y^\ell\}_{\ell \in [q_k]}$ on which $\mathcal{A}$ queries KeyGen must satisfy $\mathsf{P}(x^j, y^\ell) = 0$.

Definition 5.8 (Secret-Key Functional Encryption)**.** Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X} \to \mathcal{Z}$. A secret-key functional encryption (SK-FE) scheme for $\mathcal{F}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda)$ It takes a security parameter 1^λ and outputs a public parameter pp, and a master secret key msk. The other algorithms implicitly take pp.

$\mathsf{Enc}(\mathsf{msk}, x)$ It takes msk and $x \in \mathcal{X}$ and outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ It takes msk and $f \in \mathcal{F}$, and outputs a secret key sk.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$ It takes ct and sk, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An SK-FE scheme is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N},\ x \in \mathcal{X},\ f \in \mathcal{F}$, we have

$$\Pr\left[f(x) = \mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) : \begin{array}{l} (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \end{array}\right] = 1.$$

Security. We consider the case where each $x \in \mathcal{X}$ consists of a public part x_{pub} and a private part x_{priv}, i.e., $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})$, and each $f \in \mathcal{F}$ consists of a public part f_{pub} and a private part f_{priv}, i.e., $f = (f_{\mathsf{pub}}, f_{\mathsf{priv}})$. An SK-FE scheme is selectively partially function-hiding if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QEnc}^\beta(),\mathsf{QKeyGen}^\beta()}(\mathsf{pp}) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}$$

where $x^{j,\beta} = (x^j_{\mathsf{pub}}, x^{j,\beta}_{\mathsf{priv}})$, $f^\beta = (f_{\mathsf{pub}}, f^\beta_{\mathsf{priv}})$, $\mathsf{QEnc}^\beta(x^0, x^1)$ returns $\mathsf{Enc}(\mathsf{msk}, x^\beta)$, and $\mathsf{QKeyGen}^\beta(f^0, f^1)$ returns $\mathsf{KeyGen}(\mathsf{msk}, f^\beta)$. The admissible adversary's queries must satisfy the following condition:

1. $\mathcal{A}$ cannot query QEnc^β after querying $\mathsf{QKeyGen}^\beta$ even once.
2. If (x^0, x^1) is included in the query to QEnc^β and (f^0, f^1) is queried to $\mathsf{QKeyGen}^\beta$, then $f^0(x^0) = f^1(x^1)$.

5.5 ATTRIBUTE-BASED FE FOR ATTRIBUTE-WEIGHTED SUMS WITH INNER PRODUCT

In this section, we present an attribute-based FE for attribute-weighted sums with inner product (AB-FE for AWSw/IP). In Appendix 5.B, we show how it can be captured using the notation of MPFE. We will need the following definitions.

Definition 5.9 (Inner Product Functional Encryption)**.** Inner product functional encryption (IPFE) is a class of secret-key functional encryption (SK-FE) that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = G_1^m$ be a message space. Let $\mathcal{F} = G_2^m$ be a family of functions, where $f = [\mathbf{c}]_2 \in \mathcal{F}$ represents the function $f : \mathcal{X} \to G_T$ defined as $f([\mathbf{x}]_1) = [\langle \mathbf{x}, \mathbf{c} \rangle]_T$ where $\mathbf{x}, \mathbf{c} \in \mathbb{Z}_p^m$ are both private inputs.

A function-hiding IPFE scheme can be constructed from the MDDH assumption [Tom19, Appendix A].

Definition 5.10 (FE for AWSw/IP)**.** An FE scheme for attribute-weighted sums with inner product (AWSw/IP) is a class of SK-FE that supports the following functionality. Let $\mathbb{G}$ be bilinear groups of order p. Let $\mathcal{X} = \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i \times G_1^m$ be a message space. Let $\mathcal{F} = \mathcal{F}_{n_0,n_1}^{\mathsf{ABP}} \times G_2^m$ (see Definition 5.1 for $\mathcal{F}_{n_0,n_1}^{\mathsf{ABP}}$) be a family of functions, where $f' = (f, [\mathbf{q}]_2) \in \mathcal{F}$ represents the function $f' : \mathcal{X} \to G_T$ defined as

$$f'((\{\mathbf{x}_i, \mathbf{z}_i\}_{i \in [N]}, [\mathbf{p}]_1)) = [\sum_{i \in [N]} \langle f(\mathbf{x}_i), \mathbf{z}_i \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T$$

where $\{\mathbf{x}_i\}, f$ are public elements while $\{\mathbf{z}_i\}, [\mathbf{p}]_1, [\mathbf{q}]_2$ are private elements.

Definition 5.11 (AB-FE for AWSw/IP)**.** An attribute-based FE scheme for attribute-weighted sums with inner product (AB-FE for AWSw/IP) is a class of SK-FE that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = (\mathbb{Z}_p^{n_0'} \cup \{\star\}) \times \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i \times G_1^m$ be a message space. Let $\mathcal{F} = \mathcal{F}_{n_0',1}^{\mathsf{ABP}} \times \mathcal{F}_{n_0,n_1}^{\mathsf{ABP}} \times G_2^m$ be a family of functions, where $f = (g, h, [\mathbf{q}]_2) \in \mathcal{F}$ represents the function $f : \mathcal{X} \to G_T$ defined as

$$f((\mathbf{y}, \{\mathbf{x}_j, \mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1)) = \begin{cases} [\sum_{j \in [N]} \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T & g(\mathbf{y}) = 0 \vee \mathbf{y} = \star \\ \bot & g(\mathbf{y}) \neq 0 \end{cases}$$

where $\mathbf{y}, \{\mathbf{x}_i\}, g, h$ are public elements while $\{\mathbf{z}_i\}, [\mathbf{p}]_1, [\mathbf{q}]_2$ are private elements. For notational convenience, we define $g(\star) = 0$ for all ABPs g (even for ABPs g such that $g(\mathbf{y}) \neq 0$ for all $\mathbf{y} \in \mathbb{Z}_p^{n_0'}$).

Remark 9. As explained in the introduction, we need the wildcard functionality to make our AB-MIFE scheme secure in the corruption model. This is why we define the functionality of AB-FE for AWS such that it also supports wildcards, which we will use to construct our our AB-MIFE scheme as a building block.

5.5.1 Construction

Let k be the parameter for the MDDH_k assumption. Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme with the vector length being $k(n_0' + n_0 + 3) + n_1 + 2m + 2$. The last $m + 2$ elements are used for only the security proof. Let $(\mathsf{lgen}, \mathsf{pgb}, \mathsf{pgb}^+, \mathsf{pgb}^*, \mathsf{rec})$ be a partially garbling scheme for ABPs (Definition 5.6). Then our AB-FE scheme for AWSw/IP is described below.

$\mathsf{Setup}(1^\lambda)$ It runs $\mathsf{iPP}, \mathsf{iMSK} \leftarrow \mathsf{iSetup}(1^\lambda)$ and outputs $(\mathsf{pp}, \mathsf{msk}) = (\mathsf{iPP}, \mathsf{iMSK})$.

$\mathsf{Enc}(\mathsf{msk}, (\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}, [\mathbf{p}]_1))$ It samples $\mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{w}_1, \ldots, \mathbf{w}_{N-1} \leftarrow \mathbb{Z}_p^k$ and sets $\mathbf{w}_N = -\sum_{j\in[N-1]} \mathbf{w}_j$. If $\mathbf{y}' = \star$, it sets $\mathbf{y} = 0^{n_0'}$ and $\mathbf{v} = 0^k$, otherwise it sets $\mathbf{y} = \mathbf{y}'$ and $\mathbf{v} \leftarrow \mathbb{Z}_p^k$. Then, it defines

$$\boldsymbol{\chi}_j^\top = (\mathbf{y}, \mathbf{x}_j, 1), \quad \mathbf{X}_j = \begin{cases} (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j, \mathbf{w}_j, \mathbf{v}, \ \mathbf{p}, \ 0^{m+2}) & (j = 1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j, \mathbf{w}_j, 0^k, 0^m, 0^{m+2}) & (j > 1) \end{cases}$$

and computes $\mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{X}_j]_1)$ for all $j \in [N]$. It outputs $\mathsf{ct} = (\mathbf{y}, \{\mathbf{x}_j, \mathsf{iCT}_j\}_{j\in[N]})$.

$\mathsf{KeyGen}(\mathsf{msk}, (g, h, [\mathbf{q}]_2))$ It samples $\mathbf{r}, \mathbf{s} \leftarrow \mathbb{Z}_p^k$ and defines an ABP $\phi : \mathbb{Z}_p^{n_0'+n_0} \to \mathbb{Z}_p^{1+n_1}$ as $\phi((\mathbf{y}, \mathbf{x})) = (g(\mathbf{y}), h(\mathbf{x}))$ for $\mathbf{y} \in \mathbb{Z}_p^{n_0'}, \mathbf{x} \in \mathbb{Z}_p^{n_0}$. It computes $\mathbf{L}_1, \ldots, \mathbf{L}_t \leftarrow$

$\mathsf{lgen}(\phi)$ and $\mathbf{T} \leftarrow \mathbb{Z}_p^{(t-1)\times k}$ and defines

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r}, \ 0^k, \mathbf{q}, \ 0^{m+2}) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m, 0^{m+2}) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \ \mathbf{s}, \ 0^m, 0^{m+2}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m, 0^{m+2}) & (s+1<j) \end{cases}$$

where s is the parameter of the partial garbling scheme defined in Definition 5.6. It computes $\mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\mathbf{Y}_j]_2)$ for all $j \in [t]$ and outputs $\mathsf{sk} = (\phi, \{\mathsf{iSK}_j\}_{j\in[t]})$.

$\mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$ It parse ct, sk as $(\mathbf{y}, \{\mathbf{x}_j, \mathsf{iCT}_j\}_{j\in[N]})$ and $(\phi, \{\mathsf{iSK}_j\}_{j\in[t]})$, respectively. It outputs $\perp$ if $g(\mathbf{y}') \neq 0$. Otherwise, it computes $[d_{j,\ell}]_T = \mathsf{iDec}(\mathsf{iCT}_j, \mathsf{iSK}_\ell)$ for $j \in [N], \ell \in [t]$ and outputs

$$[d]_T = \sum_{j\in[N]} \langle [\mathbf{d}_j]_T, \mathsf{rec}(\phi, (\mathbf{y}, \mathbf{x}_j)) \rangle.$$

where $\mathbf{d}_j = (d_{j,1}, \ldots, d_{j,t})$.

Correctness. In decryption, due to the correctness of iFE, we have

$$\mathbf{d}_j = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}, \mathbf{x}_j), (\langle \mathbf{s}, \mathbf{v}\rangle, \mathbf{z}_j), \langle \mathbf{r}, \mathbf{w}_j\rangle + \langle \mathbf{p}, \mathbf{q}\rangle; \mathbf{T}\mathbf{u}_j) & (j=1) \\ \mathsf{pgb}^+(\phi, (\mathbf{y}, \mathbf{x}_j), (\ \ 0, \ \ \mathbf{z}_j), \langle \mathbf{r}, \mathbf{w}_j\rangle \qquad\quad ; \mathbf{T}\mathbf{u}_j) & (j>1) \end{cases}$$

where $\mathbf{v} = \mathbf{0}$ if $\mathbf{y}' = \star$. Thanks to the correctness of the partial garbling scheme, we have

$$\langle \mathbf{d}_j, \mathsf{rec}(\phi, (\mathbf{y}, \mathbf{x}_j)) \rangle = \begin{cases} \langle \mathbf{s}, \mathbf{v}\rangle g(\mathbf{y}) + \langle h(\mathbf{x}_j), \mathbf{z}_j\rangle + \langle \mathbf{r}, \mathbf{w}_j\rangle + \langle \mathbf{p}, \mathbf{q}\rangle & (j=1) \\ \langle h(\mathbf{x}_j), \mathbf{z}_j\rangle + \langle \mathbf{r}, \mathbf{w}_j\rangle & (j>1) \end{cases}$$

In the above, we use the fact that $\phi((\mathbf{y}, \mathbf{x}_j)) = (g(\mathbf{y}), h(\mathbf{x}_j)) \in \mathbb{Z}_p^{1+n_1}$. Hence $d = \sum_{j\in[N]} \langle h(\mathbf{x}_j), \mathbf{z}_j\rangle + \langle \mathbf{p}, \mathbf{q}\rangle$ if $g(\mathbf{y}') = 0$, since $\sum_{j\in[N]} \langle \mathbf{r}, \mathbf{w}_j\rangle = 0$.

Remark 10. A partially-hiding FE scheme for AWSw/IP can be obtained from a partially-hiding AB-FE scheme for AWSw/IP by setting $n'_0 = 0$ and g as the constant function that outputs 0.

5.5.2 Security

We argue security via the following theorem.

Theorem 5.5. *If* iFE *is function-hiding, the partial garbling scheme is secure, and the* MDDH_k *assumption holds in* $\mathbb{G}$*, then the proposed AB-FE scheme for AWSw/IP is partially function-hiding as per Definition 5.8.*

Proof. We prove the theorem via a series of hybrid games H^β_ℓ for $\ell \in [q_c]$ where q_c is the number of ciphertext queries by the adversary. We show that $\mathsf{H}^\beta_s \approx_c \mathsf{H}^\beta_1 \approx_c \cdots \approx_c \mathsf{H}^\beta_{q_c} \approx_c \mathsf{H}^\beta_f$, where H^β_s for $\beta \in \{0,1\}$ is the original security game. Intuitively, in H^β_ℓ, we program the vectors $\mathbf{X}_j$ and $\mathbf{Y}_j$ in the ciphertexts and secret keys queried by the adversary such that the challenge ciphertexts in the first ℓ queries decrypt to $\sum\langle h(\mathbf{x}_j), \mathbf{z}^0_j\rangle + \langle \mathbf{p}^0, \mathbf{q}^0\rangle$ while the rest of ciphertexts decrypts to $\sum\langle h(\mathbf{x}_j), \mathbf{z}^\beta_j\rangle + \langle \mathbf{p}^\beta, \mathbf{q}^\beta\rangle$. Then, in the last hybrid, the adversary obtains no information about β, and we can conclude the proof.

Recall that in H^β_s the challenger replies

$$\mathsf{Enc}(\mathsf{msk}, (\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}^\beta_j\}_{j\in[N]}, [\mathbf{p}^\beta]_1)) \quad \text{for } \mathsf{QEnc}^\beta(\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}^0_j, \mathbf{z}^1_j\}_{j\in[N]}, [\mathbf{p}^0]_1, [\mathbf{p}^1]_1)$$

$$\mathsf{KeyGen}(\mathsf{msk}, (g, h, [\mathbf{q}^\beta]_2)) \quad \text{for } \mathsf{QKeyGen}^\beta(g, h, [\mathbf{q}^0]_2, [\mathbf{q}^1]_2)$$

where Enc and KeyGen work as specified in Section 5.5.1. The hybrid H^β_ℓ is the same as H^β_s except the way of defining $\mathbf{X}_j$ in Enc and $\mathbf{Y}_j$ in KeyGen in the replies for ciphertext

and secret-key queries. Specifically, $\mathbf{X}_j$ in the ℓ'-th ciphertext query is defined as

$$\begin{cases} \mathbf{X}_j^{\ell'} = \begin{cases} (\chi_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^0}, \mathbf{w}_j, \mathbf{v},\ 0^m, \underline{\mathbf{p}^0}, 0^2) & (j=1) \\ (\chi_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^0}, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases} & (\ell' \le \ell) \\ \mathbf{X}_j^{\ell'} = \begin{cases} (\chi_j \otimes \mathbf{u}_j, \mathbf{z}_j^\beta, \mathbf{w}_j, \mathbf{v},\ \mathbf{p}^\beta,\ 0^{m+2}) & (j=1) \\ (\chi_j \otimes \mathbf{u}_j, \mathbf{z}_j^\beta, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases} & (\ell' > \ell) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1},\quad \mathbf{r},\ 0^k, \mathbf{q}^\beta, \underline{\mathbf{q}^0}, 0^2) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1},\quad 0^k, 0^k, 0^m, 0^{m+2}) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1},\quad 0^k,\ \mathbf{s},\ 0^m, 0^{m+2}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^{m+2}) & (s+1<j) \end{cases}$$

The hybrid H_f^β is the same as $\mathsf{H}_{q_c}^\beta$ except that $\mathbf{Y}_1$ for all queries are defined as

$$\mathbf{Y}_1 = (\mathsf{vec}(\mathbf{L}_1\mathbf{T}), 0^{n_1}, \mathbf{r}, 0^k, \underline{0^m}, \mathbf{q}^0, 0^2)$$

Note that the advantage of the adversary is 0 in H_f^β since it does not obtain the information of β. Hence, the theorem holds from Lemmas 5.6 and 5.7. ∎

Lemma 5.6. $\mathsf{H}_{q_c}^\beta \approx_c \mathsf{H}_f^\beta$ *if* iFE *is function-hiding.*

Proof. Observe that in $\mathsf{H}_{q_c}^\beta$, $\mathbf{X}_j$ is defined as

$$\mathbf{X}_j = \begin{cases} (\chi_j \otimes \mathbf{u}_j, \mathbf{z}_j^0, \mathbf{w}_j, \mathbf{v},\ 0^m, \mathbf{p}^0, 0^2) & (j=1) \\ (\chi_j \otimes \mathbf{u}_j, \mathbf{z}_j^0, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases}$$

for all queries to QEnc. Therefore, for all queries to QEnc and QKeyGen, we have

$$\langle \mathbf{X}_j, \mathbf{Y}_1 \rangle = \chi_j^\top \mathbf{L}_1 \mathbf{T} \mathbf{u}_j + \langle \mathbf{w}_j, \mathbf{r} \rangle + \langle \mathbf{p}^0, \mathbf{q}^0 \rangle$$

in *both* $\mathsf{H}_{q_c}^\beta$ and H_f^β. Hence, the indistinguishability of $\mathsf{H}_{q_c}^\beta$ and H_f^β readily follows from

the function-hiding security of iFE. ■

Lemma 5.7. *Let* $\mathsf{H}_0^\beta = \mathsf{H}_s^\beta$. *For all* $\ell \in [q_c]$, *we have* $\mathsf{H}_{\ell-1}^\beta \approx_c \mathsf{H}_\ell^\beta$.

Proof. What this lemma asserts is that for the ℓ-th ciphertext ct^ℓ and any secret key sk, the cases where decryption of these reveals $\gamma^\beta = \sum\langle h(\mathbf{x}^\ell), \mathbf{z}^{\ell,\beta}\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle$ or $\perp$ and where it reveals $\gamma^0 = \sum\langle h(\mathbf{x}^\ell), \mathbf{z}^{\ell,0}\rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0\rangle$ or $\perp$ are indistinguishable. Note that $\gamma^\beta = \gamma^0 = \gamma$ due to the query condition. As in [AGW20], our goal is the hybrid where ct^ℓ is simulatable without $\mathbf{z}^\beta, \mathbf{p}^{\ell,\beta}$, and sk is simulatable from γ. At this point, we can use the equality $\gamma^\beta = \gamma^0$ to switch the β-system to the 0-system through the following two equivalent hybrids.

$\mathsf{H}_{\ell,1}^\beta$ This hybrid is the same as $\mathsf{H}_{\ell-1}^\beta$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}_j^\ell = \begin{cases} (\underline{\quad 0^{n_0 k}, \quad 0^{n_1}, 0^k, \ 0^k, 0^m}, 0^m, 1, 0) & (j = 1) \\ (\chi_j \otimes \mathbf{u}_j, \underline{0^{n_1}}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (j > 1) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \ 0^k, \mathbf{q}^\beta, \ \mathbf{q}^0, \underline{d_j}, 0) & (j = 1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m, \underline{d_j}, 0) & (1 < j \le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \mathbf{s}, \ 0^m, 0^m, \underline{d_j}, 0) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^m, \underline{d_j}, 0) & (s+1 < j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{v} \leftarrow \mathbb{Z}_p$ (if $\mathbf{y}'^\ell \neq \star$), $\widetilde{v} = 0$ (if $\mathbf{y}'^\ell = \star$) and

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{v}g(\mathbf{y}^\ell) + \sum_{j\in[N^{(\ell)}]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta}\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \langle \mathbf{w}_1^\ell, \mathbf{r}\rangle; \widetilde{\mathbf{t}}). \quad (5.3)$$

$\mathsf{H}^{\beta}_{\ell,2}$ This hybrid is the same as $\mathsf{H}^{\beta}_{\ell,1}$ except that $(d_1, \ldots, d_t)$ in Eq. (5.3) is defined as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), \widetilde{v} g(\mathbf{y}^\ell) + \sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,0}_j \rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0 \rangle + \langle \mathbf{w}^\ell_1, \mathbf{r} \rangle; \widetilde{\mathbf{t}}).$$

We prove that $\mathsf{H}^{\beta}_{\ell-1} \approx_c \mathsf{H}^{\beta}_{\ell,1} = \mathsf{H}^{\beta}_{\ell,2} \approx_c \mathsf{H}^{\beta}_{\ell}$. We can see that $\mathsf{H}^{\beta}_{\ell,1} = \mathsf{H}^{\beta}_{\ell,2}$ by considering the two cases. If $g(\mathbf{y}'^\ell) = 0$ (that is, $g(\mathbf{y}^\ell) = 0$ or $\widetilde{v} = 0$), we have $\sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,\beta}_j \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle = \sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,0}_j \rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0 \rangle$ due to the admissibility of the adversary. Otherwise, the term $\widetilde{v} g(\mathbf{y}^\ell)$ is uniformly distributed in $\mathbb{Z}_p$ and works as a one-time pad.

Proving $\mathsf{H}^{\beta}_{\ell-1} \approx_c \mathsf{H}^{\beta}_{\ell,1}$ and $\mathsf{H}^{\beta}_{\ell,2} \approx_c \mathsf{H}^{\beta}_{\ell}$ are similar, and we prove only the former. To this end, we introduce further intermediate hybrids $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}, \ldots, \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,5}$ for $\nu \in [N^{(\ell)}]$ and show that $\mathsf{H}^{\beta}_{\ell-1} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,1,1} \approx_c \cdots \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,1,5} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,2,1} \approx_c \cdots \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,N^{(\ell)},5} = \mathsf{H}^{\beta}_{\ell,1}$. Intuitively, what we are doing in these steps is to move the information of $\mathbf{z}^{\ell,\beta}_\nu$ from $\mathbf{X}^\ell_\nu$ to $\{\mathbf{Y}_j\}_{j \in [t]}$ step by step for $\nu \in [N^\ell]$. Each hybrid is defined as follows.

$\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ This hybrid is the same as $\mathsf{H}^{\beta}_{\ell-1}$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}^\ell_j = \begin{cases} (\quad 0^{n_0 k}, \quad 0^{n_1}, 0^k, \ 0^k, 0^m, 0^m, 1, 0) & (j = 1) \\ (\chi_j \otimes \mathbf{u}_j, 0^{n_1}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (1 < j < \nu) \\ (\underline{\quad 0^{n_0 k}, \quad 0^{n_1}, 0^k}, \ 0^k, 0^m, 0^m, 0, \underline{1}) & (1 < j = \nu) \\ (\chi_j \otimes \mathbf{u}_j, \mathbf{z}^\beta_j, \ \mathbf{w}_j, \ 0^k, 0^m, 0^m, 0, 0) & (\nu < j) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \ 0^k, \mathbf{q}^\beta, \mathbf{q}^0, \underline{d_j, d'_j}) & (j = 1) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m, \underline{d_j, d'_j}) & (1 < j \leq s) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad 0^k, \ \mathbf{s}, \ 0^m, 0^m, \underline{d_j, d'_j}) & (j = s + 1) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^m, \underline{d_j, d'_j}) & (s + 1 < j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{\nu} \leftarrow \mathbb{Z}_p$ (if $\mathbf{y}'^{\ell} \neq \star$), $\widetilde{\nu} = 0$ (if $\mathbf{y}'^{\ell} = \star$) and

$$(d_1, \ldots, d_t) = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), (\langle \mathbf{s}, \mathbf{v}^\ell \rangle, \mathbf{z}_1^{\ell,\beta}), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle \\ \qquad + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \mathbf{Tu}_1^\ell) & (\nu = 1) \\ \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{\nu} g(\mathbf{y}^\ell) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle \\ \qquad + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \widetilde{\mathbf{t}}) & (\nu > 1) \end{cases}$$

$$(d_1', \ldots, d_t') = \begin{cases} \mathbf{0} & (\nu = 1) \\ \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_\nu^\ell), (0, \mathbf{z}_\nu^{\ell,\beta}), \langle \mathbf{w}_\nu^\ell, \mathbf{r} \rangle; \mathbf{Tu}_\nu^\ell) & (\nu > 1) \end{cases}$$

$\widehat{\mathsf{H}}^\beta_{\ell,\nu,2}$ This hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ except that d_i, d_i' for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), (\underline{\widetilde{\nu}}, \mathbf{z}_1^{\ell,\beta}), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \underline{\widetilde{\mathbf{t}}}) & (\nu = 1) \\ \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{\nu} g(\mathbf{y}^\ell) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle \\ + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \underline{\widetilde{r}_1}; \widetilde{\mathbf{t}}) & (\nu > 1) \end{cases}$$

$$(d_1', \ldots, d_t') = \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_\nu^\ell), (0, \mathbf{z}_\nu^{\ell,\beta}), \underline{\widetilde{r}_\nu; \widetilde{\mathbf{t}}'}) \quad (\nu > 1)$$

where $\widetilde{\mathbf{t}}, \widetilde{\mathbf{t}}' \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{r}_1, \leftarrow \mathbb{Z}_p$ and $\widetilde{r}_\nu = -\widetilde{r}_1 - \sum_{i \in [N^\ell] \setminus \{1,\nu\}} \langle \mathbf{w}_i^\ell, \mathbf{r} \rangle$.

$\widehat{\mathsf{H}}^\beta_{\ell,\nu,3}$ This hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,\nu,2}$ except that d_i, d_i' for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \underline{\mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{\nu} g(\mathbf{y}^\ell) + \langle h(\mathbf{x}_1^\ell), \mathbf{z}_1^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \widetilde{\mathbf{t}})} \quad (\nu = 1)$$

$$(d_1', \ldots, d_t') = \underline{\mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_\nu^\ell), \langle h(\mathbf{x}_\nu^\ell), \mathbf{z}_\nu^{\ell,\beta} \rangle + \widetilde{r}_\nu; \widetilde{\mathbf{t}}')} \quad (\nu > 1)$$

$\widehat{\mathsf{H}}^\beta_{\ell,\nu,4}$ For $\nu = 1$, this hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,1,3}$. Otherwise, this hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,\nu,3}$ except that d_i, d_i' for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{\nu} g(\mathbf{y}^\ell) + \sum_{j \in [\nu]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \widetilde{r}_1; \widetilde{\mathbf{t}}) \quad (\nu > 1)$$

$$(d'_1, \ldots, d'_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_\nu), \cancel{\langle h(\mathbf{x}^\ell_\nu), \mathbf{z}^{\ell,\beta}_\nu \rangle} + \widetilde{r}_\nu; \widetilde{\mathbf{t}}') \quad (\nu > 1)$$

$\widehat{\mathsf{H}}^\beta_{\ell,\nu,5}$ For $\nu = 1$, this hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,1,4}$. Otherwise, this hybrid is the same as $\widehat{\mathsf{H}}^\beta_{\ell,\nu,4}$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}^\ell_j = \begin{cases} (\quad 0^{n_0 k}, \quad 0^{n_1}, 0^k, \ 0^k, 0^m, 0^m, 1, 0) & (j = 1) \\ (\chi_j \otimes \mathbf{u}_j, 0^{n_1}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (1 < j \leq \nu) \\ (\chi_j \otimes \mathbf{u}_j, \mathbf{z}^\beta_j, \ \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (\nu < j) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \ 0^k, \mathbf{q}^\beta, \mathbf{q}^0, \underline{d_j, 0}) & (j = 1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m, \underline{d_j, 0}) & (1 < j \leq s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \mathbf{s}, \ 0^m, 0^m, \underline{d_j, 0}) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^m, \underline{d_j, 0}) & (s+1 < j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$ and

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), \widetilde{\nu} g(\mathbf{y}^\ell) + \sum_{j \in [\nu]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,\beta}_j \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \underline{\langle \mathbf{w}^\ell_1, \mathbf{r} \rangle}; \widetilde{\mathbf{t}})$$

Thanks to Theorems 5.8 to 5.12, Theorem 5.7 holds. ■

Lemma 5.8. *Let* $\widehat{\mathsf{H}}^\beta_{\ell,0,5} = \mathsf{H}^\beta_{\ell-1}$. *For all* $\nu \in [N^{(\ell)}]$, *we have* $\widehat{\mathsf{H}}^\beta_{\ell,\nu-1,5} \approx_c \widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ *if* iFE *is function-hiding.*

Proof. Observe that the difference of $\widehat{\mathsf{H}}^\beta_{\ell,\nu-1,5}$ and $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ is described as the two cases:

$\nu = 1$ In this case, $\mathbf{X}^\ell_1$ in the ℓ-th ciphertext and $\mathbf{Y}_j$ in all the secret keys in $\widehat{\mathsf{H}}^\beta_{\ell,0,5} = \mathsf{H}^\beta_{\ell-1}$ are defined as follows:

$$\mathbf{X}^\ell_1 = \quad (\chi_1 \otimes \mathbf{u}_1, \quad \mathbf{z}^\beta_1, \quad \mathbf{w}_1, \mathbf{v}, \ \mathbf{p}^\beta, \ 0^{m+2})$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^\beta,\ 0^{m+2}) & (j=1, \ell=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^\beta,\ \mathbf{q}^0, 0^2) & (j=1, \ell>1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m,\ 0^{m+2}) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m,\ 0^{m+2}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m,\ 0^{m+2}) & (s+1<j) \end{cases}$$

while the corresponding vectors in $\widehat{\mathsf{H}}^\beta_{\ell,1,1}$ are defined as

$$\mathbf{X}^\ell_1 = \quad (\underline{\quad 0^{n_0 k}, \quad 0^{n_1}, \quad 0^k, 0^k}, 0^m, 0^m,\ 1,\ 0)$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^\beta,\ \mathbf{q}^0,\ \underline{d_j}, 0) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m,\ 0^m, \underline{d_j}, 0) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m,\ 0^m, \underline{d_j}, 0) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m,\ 0^m, \underline{d_j}, 0) & (s+1<j) \end{cases}$$

where $(d_1, \ldots, d_t) = \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), (\langle \mathbf{s}, \mathbf{v}^\ell\rangle, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \langle \mathbf{w}^\ell_1, \mathbf{r}\rangle; \mathbf{T}\mathbf{u}^\ell_1)$. It is not hard to see that for all secret keys and j, $\langle \mathbf{X}^\ell_1, \mathbf{Y}_j\rangle$ in $\mathsf{H}^\beta_{\ell-1}$ and that in $\widehat{\mathsf{H}}^\beta_{\ell,1,1}$ are both equal to d_j. Hence, thanks to the function-hiding security of iFE, the two hybrids are indistinguishable. Note that the second to last entry of $\mathbf{X}_j$ other than $\mathbf{X}^\ell_1$ is 0, and thus the change of $\mathbf{Y}_j$ does not affect the other vectors.

$\nu > 1$ In this case, $\mathbf{X}^\ell_\nu$ in the ℓ-th ciphertext and $\mathbf{Y}_j$ in all the secret keys in $\widehat{\mathsf{H}}^\beta_{\ell,\nu-1,5}$ are defined as follows:

$$\mathbf{X}^\ell_\nu = \quad (\chi_\nu \otimes \mathbf{u}_\nu, \quad \mathbf{z}^\beta_\nu, \qquad \mathbf{w}_\nu, 0^k, 0^m, 0^m,\ 0,\ 0)$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r},\ 0^k, \mathbf{q}^\beta,\ \mathbf{q}^0,\ d_j, 0) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m,\ 0^m, d_j, 0) & (1 < j \le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \mathbf{s},\ 0^m,\ 0^m, d_j, 0) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m,\ 0^m, d_j, 0) & (s+1 < j) \end{cases}$$

while the corresponding vectors in $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ are defined as

$$\mathbf{X}^\ell_\nu = \quad (\underline{\quad 0^{n_0 k} \quad}, 0^{n_1}, \quad 0^k, 0^k, 0^m,\ 0^m, 0,\ \underline{1}\ )$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r},\ 0^k, \mathbf{q}^\beta,\ \mathbf{q}^0,\ d_j, \underline{d'_j}) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m,\ 0^m, d_j, \underline{d'_j}) & (1 < j \le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \mathbf{s},\ 0^m,\ 0^m, d_j, \underline{d'_j}) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m,\ 0^m, d_j, \underline{d'_j}) & (s+1 < j) \end{cases}$$

where $(d'_1, \ldots, d'_t) = \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_\nu), (0, \mathbf{z}^{\ell,\beta}_\nu), \langle \mathbf{w}^\ell_\nu, \mathbf{r}\rangle; \mathbf{Tu}^\ell_\nu)$. It is not hard to see that for all secret keys and j, $\langle \mathbf{X}^\ell_\nu, \mathbf{Y}_j\rangle$ in $\widehat{\mathsf{H}}^\beta_{\ell,\nu-1,5}$ and that in $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ are both equal to d'_j. Hence, thanks to the function-hiding security of iFE, the two hybrids are indistinguishable. Note that the last entry of $\mathbf{X}_j$ other than $\mathbf{X}^\ell_\nu$ is 0, and thus the change of $\mathbf{Y}_j$ does not affect the other vectors.

■

Lemma 5.9. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1} \approx_c \widehat{\mathsf{H}}^\beta_{\ell,\nu,2}$ if the MDDH_k assumption holds in $\mathbb{G}$.*

Proof. Observe that the difference of $\widehat{\mathsf{H}}^\beta_{\ell,\nu,1}$ and $\widehat{\mathsf{H}}^\beta_{\ell,\nu,2}$ is described as the two cases:

$\nu = 1$ In this case, d_j in $\mathbf{Y}_j$ in $\widehat{\mathsf{H}}^\beta_{\ell,1,1}$ is generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), (\langle \mathbf{s}, \mathbf{v}^\ell\rangle, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \langle \mathbf{w}^\ell_1, \mathbf{r}\rangle; \mathbf{Tu}^\ell_1)$$

while the corresponding terms in $\widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ is generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), (\underline{\widetilde{\nu}}, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta} \rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r} \rangle; \underline{\widetilde{\mathbf{t}}}).$$

Observe that $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1} = \widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ if $\mathbf{y}'^{\ell} = \star$. Hence, we focus on the case $\mathbf{y}'^{\ell} \neq \star$. Recall that pgb^{+} is efficiently computable if the random tape is given as group elements due to its linearity. Hence, $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ is reduced to

$$[\{\mathbf{T}^{\kappa}, \mathbf{s}^{\kappa}, \mathbf{T}^{\kappa}\mathbf{u}_1, \langle \mathbf{s}^{\kappa}, \mathbf{v}^{\ell} \rangle\}_{\kappa \in [q_k]}]_2 \approx_c [\{\mathbf{T}^{\kappa}, \mathbf{s}^{\kappa}, \widetilde{\mathbf{t}}^{\kappa}, \widetilde{\nu}^{\kappa}\}_{\kappa \in [q_k]}]_2$$

where q_k is the number of queries to $\mathsf{QKeyGen}^{\beta}$, which are essentially what the MDDH_k assumption asserts. Thanks to the linearity of pgb^{+}, this reduction is efficient.

$\nu > 1$ In this case, d_j and d'_j in $\mathbf{Y}_j$ in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ are generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^{*}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), \widetilde{\nu} g(\mathbf{y}^{\ell}) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}^{\ell}_j), \mathbf{z}^{\ell,\beta}_j \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta} \rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r} \rangle; \widetilde{\mathbf{t}})$$

$$(d'_1, \ldots, d'_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_{\nu}), (0, \mathbf{z}^{\ell,\beta}_{\nu}), \langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r} \rangle; \mathbf{T}\mathbf{u}^{\ell}_{\nu})$$

while the corresponding term in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ are generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^{*}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), \widetilde{\nu} g(\mathbf{y}^{\ell}) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}^{\ell}_j), \mathbf{z}^{\ell,\beta}_j \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta} \rangle + \underline{\widetilde{r}_1}; \widetilde{\mathbf{t}})$$

$$(d'_1, \ldots, d'_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_{\nu}), (0, \mathbf{z}^{\ell,\beta}_{\nu}), \underline{\widetilde{r}_{\nu}; \widetilde{\mathbf{t}'}})$$

The indistinguishability between $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ and $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ can be shown by two steps. The first step changes the random tape in pgb^{+} from $\mathbf{T}\mathbf{u}_{\nu}$ to $\widetilde{\mathbf{t}'}$. This can be proven in the same way as the case $\nu = 1$. The second step changes $\langle \mathbf{w}^{\ell}_1, \mathbf{r} \rangle$ and $\langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r} \rangle$ to $\widetilde{r}_1$ and $\widetilde{r}_{\nu}$. In this step, we would like to prove that

$$\begin{aligned}
&(\{[\mathbf{r}^{\kappa}]_2, [\langle \mathbf{w}^{\ell}_1, \mathbf{r}^{\kappa} \rangle]_2, [\langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r}^{\kappa} \rangle]_2\}_{\kappa \in [q_k]}, \{\mathbf{w}^{\ell}_j\}_{j \in [N^{\ell}] \setminus \{1,\nu\}}) \\
&\quad \approx_c (\{[\mathbf{r}^{\kappa}]_2, [\widetilde{r}^{(\kappa)}_1]_2, [\widetilde{r}^{(\kappa)}_{\nu}]_2\}_{\kappa \in [q_k]}, \{\mathbf{w}^{\ell}_j\}_{j \in [N^{\ell}] \setminus \{1,\nu\}})
\end{aligned}$$

where $\mathbf{r}^\kappa \leftarrow \mathbb{Z}_p^k$, $\mathbf{w}_1^\ell, \ldots, \mathbf{w}_{N^{(\ell)}}^\ell \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{j\in[N^{(\ell)}]} \mathbf{w}_j^\ell = \mathbf{0}$, and $\widetilde{r}_1^{(\kappa)}, \widetilde{r}_\nu^{(\kappa)} \leftarrow \mathbb{Z}_p$ s.t. $\widetilde{r}_1^{(\kappa)} + \widetilde{r}_\nu^{(\kappa)} + \sum_{j\in[N^{(\ell)}]\setminus\{1,\nu\}} \langle \mathbf{w}_j^\ell, \mathbf{r}^\kappa\rangle = \mathbf{0}$. It is not hard to see that the following indistinguishability suffices to prove the above indistinguishability:

$$([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, [\mathbf{A}\mathbf{m}_2]_2, \mathbf{m}_3, \ldots, \mathbf{m}_d) \approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2, [\mathbf{s}_2]_2, \mathbf{m}_3, \ldots, \mathbf{m}_d)$$

where $d > 1, n$ are any natural numbers, $\mathbf{A} \leftarrow \mathbb{Z}_p^{n\times k}$, $\mathbf{m}_1, \ldots, \mathbf{m}_d \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{i\in[d]} \mathbf{m}_i = \mathbf{0}$, and $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^n$ s.t. $\mathbf{s}_1 + \mathbf{s}_2 + \sum_{i\in\{3,d\}} \mathbf{A}\mathbf{m}_i = \mathbf{0}$. It is easy to see that they are distributed the same if $n \le k$, so we consider the case $n > k$. The above relation can be rewritten as

$$\begin{aligned}&([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, [-\mathbf{A}\mathbf{m}_1 - \mathbf{b}]_2, \mathbf{m}_3, \ldots, \mathbf{m}_d)\\ &\approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2, [-\mathbf{s}_1 - \mathbf{b}]_2, \mathbf{m}_3, \ldots, \mathbf{m}_d)\end{aligned}$$

where $\mathbf{b} = \sum_{i\in\{3,d\}} \mathbf{A}\mathbf{m}_i$. Hence, this is implied by the MDDH_k assumption, which assert that $([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2) \approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2)$. Thanks to the linearity of pgb^+ and pgb^*, this reduction is efficient.

■

Lemma 5.10. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^\beta_{\ell,\nu,2} \approx_s \widehat{\mathsf{H}}^\beta_{\ell,\nu,3}$ if the partial garbling scheme is secure.*

Proof. The lemma readily follows from the security of the extension of the partial garbling scheme. ■

Lemma 5.11. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^\beta_{\ell,\nu,3} = \widehat{\mathsf{H}}^\beta_{\ell,\nu,4}$.*

Proof. Recall that $\widetilde{r}_1$ and $\widetilde{r}_\nu$ are randomly distributed such that $\widetilde{r}_1 + \widetilde{r}_\nu = -\sum_{i\in[N^\ell]\setminus\{1,\nu\}} \langle \mathbf{w}_i^\ell, \mathbf{r}\rangle$. Therefore, $\widetilde{r}'_1 = \widetilde{r}_1 + \langle h(\mathbf{x}_\nu^\ell), \mathbf{z}_\nu^{\ell,\beta}\rangle$ and $\widetilde{r}'_\nu = \widetilde{r}_\nu - \langle h(\mathbf{x}_\nu^\ell), \mathbf{z}_\nu^{\ell,\beta}\rangle$ are also randomly distributed such that $\widetilde{r}'_1 + \widetilde{r}'_\nu = -\sum_{i\in[N^\ell]\setminus\{1,\nu\}} \langle \mathbf{w}_i^\ell, \mathbf{r}\rangle$. By applying this replacement, we can see that both hybrids are identical. ■

Lemma 5.12. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,4} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,5}$ if* iFE *is function-hiding, the partial garbling scheme is secure, and the MDDH_k assumption holds in $\mathbb{G}$.*

Proof. The proof of this lemma is similar to that of $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu-1,5} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,3}$. ■

5.6 ATTRIBUTE-BASED MIFE FOR ATTRIBUTE-WEIGHTED SUMS

In this section, we present our AB-MIFE for AWS in two steps as discussed in Section 5.1. In Appendix 5.B, we show how it can be captured in the context of MPFE.

Definition 5.12 (Multi-Input Functional Encryption)**.** Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X}^n \to \mathcal{Z}$.[6] An MIFE scheme for $\mathcal{F}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$ It takes a security parameter 1^λ and a number 1^n of slots, and outputs a public parameter pp, encryption keys $\{\mathsf{ek}_i\}_{i\in[n]}$, a master secret key msk. The other algorithms implicitly take pp.

$\mathsf{Enc}(\mathsf{ek}_i, x_i)$ It takes ek_i and $x_i \in \mathcal{X}$ and outputs a ciphertext ct_i.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ It takes msk and $f \in \mathcal{F}$, and outputs a secret key sk.

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It takes $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and sk, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An MIFE scheme is correct if it satisfies the following condition. For all

[6] In general, the domain of each slot can be different, i.e., f can be defined as $f : \mathcal{X}_1 \cdots \times \mathcal{X}_n \to \mathcal{Z}$. In this work, however, we only handle the case where $\mathcal{X}_i = \mathcal{X}$ for all $i \in [n]$.

$\lambda, n \in \mathbb{N},\ (x_1, \ldots, x_n) \in \mathcal{X}^n,\ f \in \mathcal{F}$, we have

$$\Pr\left[d = f(x_1, \ldots, x_n) : \begin{array}{l} (\mathsf{pp}, \{\mathsf{ek}_i\}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i) \ \text{ for } i \in [n] \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ d = \mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk}) \end{array}\right] = 1.$$

Security. We consider the case where each $x_i \in \mathcal{X}$ consists of a public part $x_{i,\mathsf{pub}}$ and a private part $x_{i,\mathsf{priv}}$, i.e., $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}})$. An MIFE scheme is selectively partially-hiding if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{pp}, \{\mathsf{ek}_i\}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}(), \mathsf{QEnc}^\beta(), \mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pp}) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}$$

where $\mathsf{QCor}(i)$ outputs ek_i, and $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ outputs $\mathsf{Enc}(\mathsf{ek}_i, x_i^\beta)$. Let $q_{c,i}$ be the numbers of queries of the forms of $\mathsf{QEnc}^\beta(i, *, *)$. Let $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = [n] \backslash \mathcal{HS}$. Then, the admissible adversary's queries must satisfy the following conditions.

- For $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ must satisfy $x_i^0 = x_i^1$.
- For $i \in \mathcal{HS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ must satisfy $x_{i,\mathsf{pub}}^0 = x_{i,\mathsf{pub}}^1$.
- $f(x_1^0, \ldots, x_n^0) = f(x_1^1, \ldots, x_n^1)$ for all sequences $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f)$ that satisfy the two conditions:
  - For all $i \in [n]$, [$\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ is queried and $i \in \mathcal{HS}$] or [$x_i^0 = x_i^1 \in \mathcal{X}_i$ and $i \in \mathcal{CS}$].
  - $\mathsf{KeyGen}(\mathsf{msk}, f)$ is queried.
- The adversary must make all queries to QCor and QEnc in one shot. That is, first it outputs $(\mathcal{CS}, \{i, x_i^0, x_i^1\})$ and obtains the response: $(\{\mathsf{ek}_i\}_{i \in \mathcal{CS}}, \{\mathsf{Enc}(\mathsf{ek}_i, x_i^\beta)\})$.

Only after the one-shot query, the adversary can query KeyGen adaptively.

We formally define attribute-based MIFE scheme for attribute-weighted sums and its security.

Definition 5.13 (AB-MIFE for AWS)**.** Attribute-based MIFE for Attribute-Weighted Sums (AB-MIFE for AWS) is a class of MIFE (Definition 5.12) that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = (\mathbb{Z}_p^{n_0'} \cup \{\star\}) \times \bigcup_{i\in\mathbb{N}}(\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$ be a message space. Let $\mathcal{F} = (\mathcal{F}^{\mathsf{ABP}}_{n_0',1} \times \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^n$ be a family of functions, where $((g_1, h_1), \ldots, (g_n, h_n)) \in \mathcal{F}$ represents the function $f : \mathcal{X}^n \to G_T$ defined as

$$f((\mathbf{y}_1, \{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}), \ldots, (\mathbf{y}_n, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]})) = \begin{cases} [\sum_{i\in[n]} \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle]_T & (g_i(\mathbf{y}_i) = 0 \text{ for all } i \in [n]) \\ \bot & (\text{otherwise}) \end{cases}$$

where $\mathbf{y}_i, \mathbf{x}_{i,j}$ are public inputs while $\mathbf{z}_{i,j}$ is a private input, and we always have $g_i(\star) = 0$.

Definition 5.14 (Security of AB-MIFE for AWS)**.** We say that an AB-FE scheme for AWS satisfies security against *legitimate* keys if the scheme is secure against adversaries that follows the condition defined below in addition to the conditions defined in Definition 5.12. Let $(\mathcal{CS}, \{i, x_i^{\ell,0}, x_i^{\ell,1}\}_{i\in[n],\ell\in[q_{c,i}]}, \{f^\eta\}_{\eta\in[q_k]})$ be the query of the adversary, where q_k is the number of queries to KeyGen, $x_i^{\ell,\beta} = (\mathbf{y}_i^\ell, \{\mathbf{x}_{i,j}^\ell, \mathbf{z}_{i,j}^{\ell,\beta}\}_{j\in[N_i^\ell]})$ and $f^\eta = \{g_i^\eta, h_i^\eta\}_{i\in[n]}$. We say that f^η is legitimate if for all $i \in \mathcal{HS}$, there exists $\ell_i' \in [q_{c,i}]$ such that $g_i^\eta(\mathbf{y}_i^{\ell_i'}) = 0$. In security against legitimate keys, f^η must be legitimate for all $\eta \in [q_k]$. In contrast, we say that an AB-FE scheme for AWS satisfies security against *any* keys if the scheme is secure against adversaries that follows just the condition defined in Definition 5.12.

5.6.1 Construction

Let aFE = (aSetup, aEnc, aKeyGen, aDec) be an FE scheme for AB-FE for AWSw/IP. Then our AB-MIFE scheme for AWS is described as follows.

$\mathsf{Setup}(1^\lambda, 1^n)$ It runs $\mathsf{aPP}_i, \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and outputs

$$\mathsf{pp} = \{\mathsf{aPP}_i\}_{i\in[n]},\ \mathsf{ek}_i = \mathsf{aMSK}_i \ \text{ for } i \in [n],\ \mathsf{msk} = \{\mathsf{ek}_i\}_{i\in[n]}.$$

$\mathsf{Enc}(\mathsf{ek}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}))$ It outputs

$$\mathsf{ct}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, [1]_1)).$$

$\mathsf{KeyGen}(\mathsf{msk}, \{g_i, h_i\}_{i\in[n]})$ It samples $r_1, \ldots, r_{n-1} \leftarrow \mathbb{Z}_p$, sets $r_n = -\sum_{i\in[n-1]} r_i$, and outputs $\mathsf{sk} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where

$$\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i]_2)).$$

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It parse $\mathsf{ct}_i, \mathsf{sk}$ as $\mathsf{aCT}_i, \{\mathsf{aSK}_i\}_{i\in[n]}$, respectively. If there exists i such that $g_i(\mathbf{y}_i) \neq 0$, it outputs $\perp$. Otherwise, it computes $[d_i]_T = \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ for $i \in [n]$, and outputs $[d]_T = \sum_{i\in[n]} [d_i]_T$.

Correctness. Due to the correctness of aFE, we have

$$d_i = \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + r_i$$

Hence $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$ since $\sum_{i\in[n]} r_i = 0$.

5.6.2 Security

The proposed scheme is secure against legitimate keys as stated by the following theorem.

Theorem 5.13. *If* aFE *is partially function-hiding, then the proposed AB-MIFE scheme for AWS satisfies security against legitimate keys as per Definition 5.14.*

Proof. We prove the theorem via two hybrids H_1^β and H_2^β. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta = \mathsf{H}_2^\beta$,

where H_s^β for $\beta \in \{0,1\}$ is the original security game. Recall that in H_s^β the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}, [1]_1)) \quad \text{and} \quad \{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, respectively, where

$$x_i^\beta = (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}) \quad \text{and} \quad f = \{g_i, h_i\}_{i\in[n]}.$$

The hybrid H_1^β is the same as H_s^β except that for all $i \in \mathcal{HS}$, the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \underline{\mathbf{z}_{i,j}^0}\}_{j\in[N_i]}, [1]_1)) \quad \text{and}$$
$$\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i + \underline{\sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta}\rangle - \sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,0}\rangle}]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, respectively, where $\{\mathbf{x}_{i,j}^{\ell_i}, \mathbf{z}_{i,j}^{\ell_i,0}, \mathbf{z}_{i,j}^{\ell_i,1}\}_{j\in[N_i^{\ell_i}]}$ are the components of the ℓ_i-th challenge message, and $\ell_i = \min\{\ell' \in [q_{c,i}] \mid g_i(\mathbf{y}_i^{\ell'}) = 0\}$ for $i \in \mathcal{HS}$. Since all secret keys are legitimate, such ℓ_i always exists for each key query.

The hybrid H_2^β is the same as H_1^β except that for all $i \in \mathcal{HS}$, the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^0\}_{j\in[N_i]}, [1]_1)) \quad \text{and}$$
$$\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i + \underline{\sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta}\rangle - \sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,0}\rangle}]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, respectively. Note that the advantage of the adversary is 0 in H_2^β since it does not obtain the information of β. Hence the theorem follows from Theorems 5.14 and 5.15. ∎

Lemma 5.14. *We have* $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta$ *if* aFE *is partially function-hiding.*

Proof. Let $q_{c.i}$ be the number of the ciphertext queries for slot i, and q_k be the number

of the secret key queries. For $\mu \in [q_{c,i}]$ and $\nu \in [q_k]$, let aCT_i^μ be the μ-th challenge ciphertext for slot i, and aSK_i^ν be the i-th element of the ν-th secret key. For $j \in \{s, 1\}$, let $\delta_{j,i}^{\mu,\nu} = \mathsf{aDec}(\mathsf{aCT}_i^\mu, \mathsf{aSK}_i^\nu)$ be the decryption value in H_j^β. Then, what we need to prove is $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ for all $i \in \mathcal{HS}, \mu \in [q_{c,i}], \nu \in [q_k]$.

This can be proven by the three cases.

- If $g_i^\nu(\mathbf{y}_i^\mu) \neq 0$, we have $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu} = \perp$ for all $i \in \mathcal{HS}, \mu \in [q_{c,i}], \nu \in [q_k]$.
- If $\mu = \ell_i$, we have $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu} = \sum_{j \in [N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta} \rangle$ for all $i \in \mathcal{HS}, \nu \in [q_k]$.
- If $\mu > \ell_i$ and $g_i^\nu(\mathbf{y}_i^\mu) = 0$, we have $\delta_{s,i}^{\mu,\nu} = \sum_{j \in [N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,\beta} \rangle$ and

$$\delta_{1,i}^{\mu,\nu} = \sum_{j \in [N_i^{\ell_i}]} (\langle h_i^\nu(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,0} \rangle + \langle h_i^\nu(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta} \rangle - \langle h_i^\nu(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,0} \rangle)$$

for all $i \in \mathcal{HS}, \nu \in [q_k]$. We can prove $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ as follows. Due to the admissibility of the adversary, we have

$$\sum_{k \in \mathcal{HS}} \sum_{j \in [N_k^{\ell_k}]} \langle h_k^\nu(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,0} \rangle = \sum_{k \in \mathcal{HS}} \sum_{j \in [N_k^{\ell_k}]} \langle h_k^\nu(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,1} \rangle \tag{5.4}$$

$$\begin{aligned} &\sum_{j \in [N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,0} \rangle + \sum_{k \in \mathcal{HS} \setminus \{i\}} \sum_{j \in [N_k^{\ell_k}]} \langle h_k^\nu(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,0} \rangle \\ = &\sum_{j \in [N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,1} \rangle + \sum_{k \in \mathcal{HS} \setminus \{i\}} \sum_{j \in [N_k^{\ell_k}]} \langle h_k^\nu(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,1} \rangle \end{aligned} \tag{5.5}$$

We can readily obtain $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ by subtracting Eq.(5.4) from Eq.(5.5) in the third case.

■

Lemma 5.15. $\mathsf{H}_1^\beta = \mathsf{H}_2^\beta$.

Proof. From Eq.(5.4), the following distributions are identical:

$$\left\{ (r_1, \ldots, r_n) \ : \ r_1, \ldots, r_{n-1} \leftarrow \mathbb{Z}_p, \ r_n = -\sum_{i \in [n-1]} r_i \right\} \quad \text{and}$$

$$\left\{ (r_1, \ldots, r_n) \ : \begin{array}{l} r'_1, \ldots, r'_{n-1} \leftarrow \mathbb{Z}_p,\ r'_n = -\sum_{i \in [n-1]} r'_i \\ r_i = \begin{cases} r'_i + \sum_{j \in [N_i^{\ell_i}]} (\langle h_i^{\nu}(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta} \rangle - \langle h_i^{\nu}(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,0} \rangle) & (i \in \mathcal{HS}) \\ r'_i & (i \in \mathcal{CS}) \end{cases} \end{array} \right\}$$

Hence H_1^{β} and H_2^{β} are identically distributed. ■

5.6.3 Amplifying security against *Any Keys*

In this section, we present how to convert an AB-MIFE scheme for AWS with security against legitimate keys to one with security against any keys. In the conversion, we use a ciphertext-policy ABE (CP-ABE) scheme for ABP and a (n-out-of-n) secret sharing scheme. A CP-ABE scheme for ABP with wildcards is an ABE scheme (Definition 5.7) that supports predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ where $\mathcal{X} = \mathcal{F}_{n_0,1}^{\mathsf{ABP}}$, $\mathcal{Y} = \mathbb{Z}_p^{n_0} \cup \{\star\}$, and for $g \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$, P is defined as

$$\mathsf{P}(g, \mathbf{y}) = \begin{cases} 1 & g(\mathbf{y}) = 0 \\ 0 & g(\mathbf{y}) \neq 0 \end{cases}$$

A CP-ABE scheme for ABP with wildcards is easily obtained from the CP-ABE scheme for ABP in [LL20b] just by setting the master secret key as the secret key for the wildcard.

Construction Let $\mathsf{wmFE} = (\mathsf{wmSetup}, \mathsf{wmEnc}, \mathsf{wmKeyGen}, \mathsf{wmDec})$ be an AB-MIFE scheme for AWS with security against legitimate keys, $\mathsf{ABE} = (\mathsf{abSetup}, \mathsf{abEnc}, \mathsf{abKeyGen}, \mathsf{abDec})$ be an CP-ABE scheme for ABP, and $(\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme. Then, an AB-MIFE scheme for AWS can be constructed as shown below.

$\mathsf{Setup}(1^{\lambda}, 1^n)$ It runs $\mathsf{wmPP}, \{\mathsf{wmEK}_i\}_{i \in [n]}, \mathsf{wmMSK} \leftarrow \mathsf{wmSetup}(1^{\lambda})$ and abPK_i,

$\mathsf{abMSK}_i \leftarrow \mathsf{abSetup}(1^\lambda)$ for $i \in [n]$. It outputs $\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in[n]}, \mathsf{msk}$ as follows:

$$\mathsf{pp} = (\mathsf{wmPP}, \{\mathsf{abPK}_i\}_{i\in[n]}), \quad \mathsf{ek}_i = (\mathsf{wmEK}_i, \mathsf{abMSK}_i), \quad \mathsf{msk} = \mathsf{wmMSK}$$

$\mathsf{Enc}(\mathsf{ek}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}))$ It outputs $\mathsf{ct}_i = (\mathsf{wmCT}_i, \mathsf{abSK}_i)$ where

$$\mathsf{wmCT}_i \leftarrow \mathsf{wmEnc}(\mathsf{wmEK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]})), \quad \mathsf{abSK}_i \leftarrow \mathsf{abKeyGen}(\mathsf{abMSK}_i, \mathbf{y}_i)$$

$\mathsf{KeyGen}(\mathsf{msk}, \{g_i, h_i\}_{i\in[n]})$ It outputs sk as follows:

$$\mathsf{wmSK} \leftarrow \mathsf{wmKeyGen}(\mathsf{wmMSK}_i, \{g_i, h_i\}_{i\in[n]}), \quad (\sigma_1, \ldots, \sigma_n) \leftarrow \mathsf{Share}(\mathsf{wmSK}, n)$$
$$\mathsf{abCT}_i \leftarrow \mathsf{abEnc}(g_i, \sigma_i) \text{ for } i \in [n], \quad \mathsf{sk} = \{\mathsf{abCT}_i\}_{i\in[n]}$$

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It parse $\mathsf{ct}_i, \mathsf{sk}$ as $(\mathsf{wmCT}_i, \mathsf{abSK}_i), \{\mathsf{abCT}_i\}_{i\in[n]}$, respectively. If there exists i such that $g_i(\mathbf{y}_i) \neq 0$, it outputs $\perp$. Otherwise, it outputs $[d]_T$ as follows:

$$\sigma'_i = \mathsf{abDec}(\mathsf{abCT}_i, \mathsf{abSK}_i) \text{ for } i \in [n], \quad \mathsf{wmSK}' = \mathsf{Rec}(\sigma'_1, \ldots, \sigma'_n)$$
$$[d]_T = \mathsf{wmDec}(\mathsf{wmCT}_1, \ldots, \mathsf{wmCT}_n, \mathsf{wmSK}')$$

Correctness and Security. Due to the correctness of ABE, $\sigma'_1, \ldots, \sigma'_n$ are valid shares of wmSK for $\{g_i, h_i\}_{i\in[n]}$. Thus, thanks to the correctness of wmFE, we have $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$.

We argue security via the following theorem.

Theorem 5.16. *If* wmFE *has security against legitimate keys,* ABE *is selectively secure, and the secret sharing scheme is secure, then the proposed scheme satisfies security against any keys, i.e., selectively partially-hiding security in Definition 5.12.*

Proof. We prove the theorem via three hybrids $\mathsf{H}_1^\beta, \mathsf{H}_2^\beta, \mathsf{H}_3^\beta$. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta = \mathsf{H}_2^\beta \approx_c \mathsf{H}_3^\beta$, where H_s^β for $\beta \in \{0,1\}$ is the original security game. Let us call a secret key with which all the combinations of challenge ciphertexts decrypt to $\perp$ a illegitimate key. For each illegitimate key for $\{g_i, h_i\}_{i\in[n]}$, there exists $i' \in \mathcal{HS}$ such that $g_{i'}(\mathbf{y}_{i'}^\ell) \neq 0$ for all $\ell \in [q_{c,i'}]$.

In H_1^β, we change the replies to the illegitimate-secret-key queries. Specifically, $\mathsf{abCT}_{i'}$ in sk is generated as $\mathsf{abCT}_{i'} \leftarrow \mathsf{abEnc}(g_{i'}, 0^m)$ instead of $\mathsf{abCT}_{i'} \leftarrow \mathsf{abEnc}(g_{i'}, \sigma_{i'})$, where m is the bit-length of a share. We can easily observe that $\mathsf{H}_s \approx_c \mathsf{H}_1$ due to the security of the CP-ABE scheme for ABP.

In H_2^β, we change the replies to the illegitimate-secret-key queries. Specifically, $\sigma_1, \ldots, \sigma_n$ is generated as $\sigma_i \leftarrow \{0,1\}^m$ for $i \in [n]$ instead of being generated by the sharing algorithm. $\mathsf{H}_1 = \mathsf{H}_2$ directly follows from the security of the secret sharing scheme.

In H_3^β, we change the challenge ciphertexts. Instead of replying $\mathsf{Enc}(\mathsf{ek}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}))$ to ciphertext queries, the challenger replies $\mathsf{Enc}(\mathsf{ek}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^0\}_{j\in[N_i]}))$ for all the queries. $\mathsf{H}_2 \approx_c \mathsf{H}_3$ directly follows from the security of wmFE. Note that the advantage of the adversary is 0 in H_3^β since it does not obtain the information of β. ■

5.7 MULTI-CLIENT FE FOR ATTRIBUTE-WEIGHTED SUMS

We define multi-client functional encryption, which basically follows the definition in [ABG19]. The essential difference from the definition in [ABG19] is that we add the definition of selective security.

Definition 5.15 (Multi-Client Functional Encryption)**.** Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X}^n \to \mathcal{Z}$. Let $\mathcal{L}$ be a label space. An MCFE scheme for $\mathcal{F}$ and $\mathcal{L}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$ It takes a security parameter 1^λ and a number 1^n of slots, and outputs a public parameter pp, encryption keys $\{\mathsf{ek}_i\}_{i\in[n]}$, a master secret key msk. The other algorithms implicitly take pp.

$\mathsf{Enc}(\mathsf{ek}_i, x_i, L)$ It takes ek_i, an index $i \in [n]$, $x_i \in \mathcal{X}$, and a label L and outputs a ciphertext ct_i.

$\mathsf{KeyGen}(\mathsf{msk}, f)$ It takes msk and $f \in \mathcal{F}$, and outputs a secret key sk.

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It takes $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and sk, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An MCFE scheme is correct if it satisfies the following condition. For all $\lambda, n \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{X}^n$, $f \in \mathcal{F}$, $L \in \mathcal{L}$, we have

$$\Pr\left[d = f(x_1, \ldots, x_n) : \begin{array}{l} (\mathsf{pp}, \{\mathsf{ek}_i\}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, x_i, L) \ \text{ for } i \in [n] \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ d = \mathsf{Dec}(\mathsf{ct}_1, \ldots, , \mathsf{ct}_n, \mathsf{sk}) \end{array} \right] = 1.$$

Security. We consider the case where each $x_i \in \mathcal{X}$ consists of a public part $x_{i,\mathsf{pub}}$ and a private part $x_{i,\mathsf{priv}}$, i.e., $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}})$. An MCFE scheme is xx-yy-partially-hiding (xx $\in$ {sel, sta, adt}, yy $\in$ {any, pos}) if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{pp}, \{\mathsf{ek}_i\}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}(), \mathsf{QEnc}^\beta(), \mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{pp}) \end{array} \right] \leq \frac{1}{2} + \mathrm{negl}$$

where $\mathsf{QCor}(i)$ outputs ek_i, and $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ outputs $\mathsf{Enc}(\mathsf{ek}_i, x_i^\beta, L)$. Let $q_{c,i,L}$

be the numbers of queries of the forms of $\mathsf{QEnc}^{\beta}(i, *, *, L)$. Let $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = [n] \backslash \mathcal{HS}$. Then, the admissible adversary's queries must satisfy the following conditions.

- For $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L)$ must satisfy $x_i^0 = x_i^1$.
- For $i \in \mathcal{HS}$, the queries $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L)$ must satisfy $x_{i,\mathsf{pub}}^0 = x_{i,\mathsf{pub}}^1$.
- $f(x_1^0, \ldots, x_n^0) = f(x_1^1, \ldots, x_n^1)$ for all sequences $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f, L)$ that satisfy the two conditions:
  - For all $i \in [n]$, [$\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L)$ is queried and $i \in \mathcal{HS}$] or [$x_i^0 = x_i^1 \in \mathcal{X}_i$ and $i \in \mathcal{CS}$].
  - $\mathsf{KeyGen}(\mathsf{msk}, f)$ is queried.
- When xx = sta: the adversary cannot query QCor after querying QEnc or KeyGen even once.
- When xx = sel: the adversary must make all queries to QCor and QEnc in one shot. That is, first it outputs $(\mathcal{CS}, \{i, x_i^0, x_i^1, L\})$ and obtains the response: $(\{\mathsf{ek}_i\}_{i \in \mathcal{CS}}, \{\mathsf{Enc}(\mathsf{ek}_i, x_i^{\beta}, L)\})$. Only after the one-shot query, the adversary can query KeyGen adaptively.
- When yy = pos: for each $L \in \mathcal{L}$, either $q_{c,i,L} > 0$ for all $i \in \mathcal{HS}$ or $q_{c,i,L} = 0$ for all $i \in \mathcal{HS}$[7].

First, we formally define MCFE for AWS.

Definition 5.16 (MCFE for Attribute-Weighted Sums)**.** MCFE for Attribute-Weighted Sums (AWS) is a class of MCFE (Definition 5.15) that supports the following functionality. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. Let $\mathcal{X} = \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$ be a message space. Let $\mathcal{F} = (\mathcal{F}_{n_0,n_1}^{\mathsf{ABP}})^n$ be a family of functions, where $(f_1, \ldots, f_n) \in \mathcal{F}$ represents the function $f' : \mathcal{X}^n \to G_T$ defined as

$$f'(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j \in [N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j \in [N_n]}) = [\sum_{i \in [n]} \sum_{j \in [N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle]_T.$$

[7] We can covert a xx-pos-partially-hiding scheme to xx-any-partially-hiding scheme generically [ABG19].

5.7.1 Construction

Let $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ be an FE scheme for $\mathsf{AWSw/IP}$. Let $\mathsf{PRF}^{\mathsf{K}} : \mathcal{L} \to \mathbb{Z}_p^k$ be a PRF with key space $\mathcal{K}$. Let k be the parameter for the MDDH_k assumption. Then construction of our MCFE scheme for AWS is described as follows.

$\mathsf{Setup}(1^\lambda, 1^n)$ It runs $\mathsf{aPP}_i, \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$, chooses $\mathsf{K}_{i,j} \leftarrow \mathcal{K}$ for $i, j \in [n], i < j$, and sets $\mathsf{K}_{i,j} = \mathsf{K}_{j,i}$ for $j < i$. It outputs

$$\mathsf{pp} = \{\mathsf{aPP}_i\}_{i\in[n]},\ \mathsf{ek}_i = (\mathsf{aMSK}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]\setminus\{i\}}) \ \text{ for } i \in [n],\ \mathsf{msk} = \{\mathsf{ek}_i\}_{i\in[n]}.$$

$\mathsf{Enc}(\mathsf{ek}_i, L, x_i = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]})$ It computes $\mathbf{v}_{L,i} = \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L)$ and outputs

$$\mathsf{ct}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, x_i, [(\mathbf{v}_{L,i}, 0)]_1).$$

$\mathsf{KeyGen}(\mathsf{msk}, \{f_i\}_{i\in[n]})$ It samples $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and outputs $\mathsf{sk} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where

$$\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [(\mathbf{s}, 0)]_2).$$

$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathsf{sk})$ It parse $\mathsf{ct}_i, \mathsf{sk}$ as $\mathsf{aCT}_i, \{\mathsf{aSK}_i\}_{i\in[n]}$, respectively. It computes $[d_i]_T = \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ for $i \in [n]$, and outputs $[d]_T = \sum_{i\in[n]} [d_i]_T$.

Correctness. Due to the correctness of aFE, we have

$$d_i = \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \langle \mathbf{v}_{L,i}, \mathbf{s}\rangle$$

Hence $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$ since $\sum_{i\in[n]} \langle \mathbf{v}_{L,i}, \mathbf{s}\rangle = 0$.

5.7.2 Security

We argue security via the following theorem.

Theorem 5.17. *If* aFE *is partially function-hiding, and the* MDDH_k *assumption holds in* $\mathbb{G}$*, then the proposed MCFE scheme for AWS is sel-pos-partially-hiding as per Definition 5.15.*

Proof. We prove the theorem via a series of hybrid games H^β_ℓ for $\ell \in \{0\} \cup [q_L]$ where $q_L = |\{L \mid q_{c,i,L} > 0\}|$ for $i \in \mathcal{HS}$ is the maximum number of labels queried by the adversary. We show that $\mathsf{H}^\beta_s \approx_c \mathsf{H}^\beta_0 \approx_c \mathsf{H}^\beta_1 \approx_c \cdots \approx_c \mathsf{H}^\beta_{q_L}$, where H^β_s for $\beta \in \{0,1\}$ is the original security game. Recall that in H^β_s the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, x_i^\beta, [\mathbf{p}_i]_1)$ and $\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [\mathbf{q}_i]_2)\}_{i\in[n]}$ for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, respectively, where

$$x_i^\beta = \{\mathbf{x}_{i,j}, \mathbf{z}^\beta_{i,j}\}_{j\in[N_i]}, \quad \mathbf{p}_i = (\mathbf{v}_{L,i}, 0), \quad \mathbf{q}_i = (\mathbf{s}, 0).$$

Let $\mathcal{QL} = \{L_1, \ldots, L_{q_L}\}$ be the labels that are queried by the adversary. H^β_0 is the same as H^β_s except that the challenger randomly chooses $\mathbf{v}_{L,i} \in \mathbb{Z}_p^k$ for $i \in \mathcal{HS}, L \in \mathcal{QL}$ such that $\sum_{i\in\mathcal{HS}} \mathbf{v}_{L,i} + \sum_{i\in\mathcal{CS}} \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L) = \mathbf{0}$. The hybrid H^β_ℓ is the same as H^β_0 except that for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ such that $L \in \{L_1, \ldots, L_\ell\}$, the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, \underline{x_i^0}, \mathbf{p}_i)$. Note that the advantage of the adversary is 0 in $\mathsf{H}^\beta_{q_L}$ since it does not obtain the information of β. It is not hard to see that $\mathsf{H}^\beta_s \approx_c \mathsf{H}^\beta_0$ follows from the security of the PRF. Hence, the theorem holds from Theorem 5.18. ■

Lemma 5.18. *Let* $\mathsf{H}^\beta_0 = \mathsf{H}^\beta_s$*. For all* $\ell \in [q_L]$*, we have* $\mathsf{H}^\beta_{\ell-1} \approx_c \mathsf{H}^\beta_\ell$*.*

Proof. To prove the lemma we introduce intermediate hybrids $\mathsf{H}^\beta_{\ell,1}, \mathsf{H}^\beta_{\ell,2}, \mathsf{H}^\beta_{\ell,3}$, which are defined as follows:

$\mathsf{H}^\beta_{\ell,1}$ This hybrid is the same as $\mathsf{H}^\beta_{\ell-1}$ except that the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, x_i^0, [\mathbf{p}_i]_1)$ and $\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [\mathbf{q}_i]_2)\}_{i\in[n]}$ for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, respectively, where

$$\mathbf{p}_i = (\underline{0^k, 1}), \quad \mathbf{q}_i = (\mathbf{s}, \underline{\langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})}) \text{ for } i \in \mathcal{HS}.$$

where $(x^{\kappa,\beta}_{i,L_\ell}, x^{\kappa,0}_{i,L_\ell})$ is the pair of challenge messages in the κ-th query to QEnc of the form $(i, *, *, L_\ell)$.

$\mathsf{H}^\beta_{\ell,2}$ This hybrid is the same as $\mathsf{H}^\beta_{\ell,1}$ except that in the replies for the queries $\mathsf{QEnc}^\beta(i, x^0_i, x^1_i, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathbf{p}_i$ and $\mathbf{q}_i$ is defined as

$$v_{L_\ell,i} \leftarrow \mathbb{Z}_p \quad \text{for } i \in \mathcal{HS} \text{ s.t. } \sum_{i\in\mathcal{HS}} v_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle = 0$$
$$\mathbf{p}_i = (0^k, 1), \quad \mathbf{q}_i = (\mathbf{s}, \underline{v_{L_\ell,i}} + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})) \quad \text{for } i \in \mathcal{HS}.$$

$\mathsf{H}^\beta_{\ell,3}$ This hybrid is the same as $\mathsf{H}^\beta_{\ell,2}$ except that in the replies for the queries $\mathsf{QEnc}^\beta(i, x^0_i, x^1_i, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{msk}, f)$, $\mathbf{p}_i$ and $\mathbf{q}_i$ is defined as

$$v_{L_\ell,i} \leftarrow \mathbb{Z}_p \quad \text{for } i \in \mathcal{HS} \text{ s.t. } \sum_{i\in\mathcal{HS}} v_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle = 0$$
$$\mathbf{p}_i = (0^k, 1), \quad \mathbf{q}_i = (\mathbf{s}, v_{L_\ell,i} + \cancel{f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})}) \quad \text{for } i \in \mathcal{HS}.$$

Thanks to Theorems 5.19 to 5.22, Theorem 5.18 holds. ■

Lemma 5.19. *Let* $\mathsf{H}^\beta_0 = \mathsf{H}^\beta_s$. *For all* $\ell \in [q_L]$, *we have* $\mathsf{H}^\beta_{\ell-1} \approx_c \mathsf{H}^\beta_{\ell,1}$ *if* aFE *is partially function-hiding.*

Proof. Observe that for all aCT_is that the adversary obtains as a reply to the query of the form $\mathsf{QEnc}^\beta(i, x^0_i, x^1_i, L)$ and all aSK_is that it obtains as a reply to the query of the form $\mathsf{KeyGen}(\mathsf{msk}, f = \{f_i\}_{i\in\mathcal{HS}})$, the output of $\mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ in $\mathsf{H}^\beta_{\ell-1}$ and that in $\mathsf{H}^\beta_{\ell,1}$ are equal for all $i \in \mathcal{HS}$. Here, we use the fact that for all $i \in \mathcal{HS}$ and $\kappa \in [q_{c,i,L_\ell}]$, we have

$$f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell}) = f_i(x^{\kappa,\beta}_{i,L_\ell}) - f_i(x^{\kappa,0}_{i,L_\ell}).$$

This is basically obtained by Eq. (5.6) − Eq. (5.7):

$$\sum_{i\in\mathcal{HS}} f_i(x^{1,\beta}_{i,L_\ell}) = \sum_{i\in\mathcal{HS}} f_i(x^{1,0}_{i,L_\ell}) \tag{5.6}$$

$$f_{i'}(x^{\kappa,\beta}_{i',L_\ell}) + \sum_{i\in\mathcal{HS}\setminus\{i'\}} f_i(x^{1,\beta}_{i,L_\ell}) = f_{i'}(x^{\kappa,0}_{i',L_\ell}) + \sum_{i\in\mathcal{HS}\setminus\{i'\}} f_i(x^{1,0}_{i,L_\ell}) \tag{5.7}$$

which follows from the query condition in Definition 5.15. Hence, thanks to the partially function-hiding security of aFE, these hybrids are indistinguishable. ■

Lemma 5.20. *For all $\ell \in [q_L]$, we have $\mathsf{H}^{\beta}_{\ell,1} \approx_c \mathsf{H}^{\beta}_{\ell,2}$ if the MDDH$_k$ holds in $\mathbb{G}$.*

Proof. We would like to prove that

$$\{[\mathbf{s}^\kappa]_2, \{[\langle \mathbf{s}^\kappa, \mathbf{v}_{L_\ell,i}\rangle]_2\}_{i\in\mathcal{HS}}\}_{\kappa\in[q_k]} \approx_c \{[\mathbf{s}^\kappa]_2, \{[v^\kappa_{L_\ell,i}]_2\}_{i\in\mathcal{HS}}\}_{\kappa\in[q_k]}$$

where q_k is the number of queries to QKeyGen, $\mathbf{s}^\kappa \leftarrow \mathbb{Z}_p^k$, $\mathbf{v}_{L_\ell,i} \leftarrow \mathbb{Z}_p^k$ for $i \in \mathcal{HS}$ s.t. $\sum_{i\in\mathcal{HS}} \mathbf{v}_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L) = \mathbf{0}$, and $v^\kappa_{L_\ell,i} \leftarrow \mathbb{Z}_p$ for $i \in \mathcal{HS}$ and $\kappa \in [q_k]$ s.t. $\sum_{i\in\mathcal{HS}} v_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \langle \mathbf{s}^\kappa, \mathbf{v}_{L_\ell,i}\rangle = 0$. It is not hard to see that the following indistinguishability suffices to prove the above indistinguishability:

$$([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, \ldots, [\mathbf{A}\mathbf{m}_d]_2) \approx_c ([\mathbf{A}]_2, [\mathbf{r}_1]_2, \ldots, [\mathbf{r}_d]_2)$$

where $d > 1$, n are any natural numbers, $\mathbf{c}$ is any vectors in $\mathbb{Z}_p^k$, $\mathbf{A} \leftarrow \mathbb{Z}_p^{n\times k}$, $\mathbf{m}_1, \ldots, \mathbf{m}_d \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{i\in[d]} \mathbf{m}_i = \mathbf{c}$, and $\mathbf{r}_1, \ldots, \mathbf{r}_d \leftarrow \mathbb{Z}_p^n$ s.t. $\sum_{i\in[d]} \mathbf{r}_i = \mathbf{A}\mathbf{c}$. It is easy to see that they are distributed the same if $n \le k$, so we consider the case $n > k$. The above relation can be rewritten as

$$\begin{aligned}&([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, \ldots, [\mathbf{A}\mathbf{m}_{d-1}]_2, [\mathbf{A}\mathbf{c} - \sum_{i\in[d-1]} \mathbf{A}\mathbf{m}_i]_2)\\ \approx_c\ &([\mathbf{A}]_2, [\mathbf{r}_1]_2, \ldots, [\mathbf{r}_{d-1}]_2, [\mathbf{A}\mathbf{c} - \sum_{i\in[d-1]} \mathbf{r}_i]_2)\end{aligned}$$

This is implied by the $d-1$-fold MDDH$_k$ assumption, which asserts that

$$[(\mathbf{A}, \mathbf{A}\mathbf{m}_1, \ldots, \mathbf{A}\mathbf{m}_{d-1})]_2 \approx_c [(\mathbf{A}, \mathbf{r}_1, \ldots, \mathbf{r}_{d-1})]_2.$$

■

Lemma 5.21. *For all $\ell \in [q_L]$, we have $\mathsf{H}^{\beta}_{\ell,2} = \mathsf{H}^{\beta}_{\ell,3}$*

Proof. As we see above, Eq. (5.6) holds due to the query condition in Definition 5.15. Thus, $\{v_{L_\ell,i}\}_{i\in[\mathcal{HS}]}$ and $\{v_{L_\ell,i} + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})\}_{i\in[\mathcal{HS}]}$ are both randomly distributed in $\mathbb{Z}_p$ such that the summation of these is equal to $-\sum_{i\in\mathcal{CS}}\langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle$. ■

Lemma 5.22. *For all $\ell \in [q_L]$, we have $\mathsf{H}^{\beta}_{\ell,3} \approx_c \mathsf{H}^{\beta}_{\ell}$ if* aFE *is partially function-hiding and the MDDH_k holds in $\mathbb{G}$.*

Proof. This lemma can be proven in the same way as $\mathsf{H}^{\beta}_{\ell-1} \approx_c \mathsf{H}^{\beta}_{\ell,2}$. ■

5.8 DYNAMIC DECENTRALIZED FE FOR ATTRIBUTE WEIGHTED SUMS

In this section, we present a dynamic decentralized FE scheme for attribute weighted sums (DDFE for AWS). In Appendix 5.B, we show how it can be captured in the context of dynamic MPFE.

5.8.1 Definition

Definition 5.17 (Dynamic Decentralized Functional Encryption)**.** Let $\mathcal{ID}, \mathcal{K}, \mathcal{M}$ be an ID space, a key space, and a message space, respectively. $\mathcal{M}$ consists of a public part $\mathcal{M}_{\mathsf{pub}}$ and a private part $\mathcal{M}_{\mathsf{priv}}$. Let f be a function such that $f : \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{K})^i \times \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{M})^i \to \mathcal{Z}$. A DDFE scheme for f consists of five algorithms.

$\mathsf{Setup}(1^\lambda)$ It takes a security parameter 1^λ and outputs a public parameter pp. The other algorithms implicitly take pp.

$\mathsf{LSetup}(\mathsf{pp})$ It takes pp and outputs local public parameter pk_i and a master secret key msk_i. The following three algorithms implicitly take pk_i.

$\mathsf{Enc}(\mathsf{msk}_i, m)$ It takes msk_i and $m \in \mathcal{M}$, and outputs a ciphertext ct_i.

$\mathsf{KeyGen}(\mathsf{msk}_i, k)$ It takes msk_i and $k \in \mathcal{K}$, and outputs a secret key sk_i.

$\mathsf{Dec}(\{\mathsf{sk}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{ct}_i\}_{i\in\mathcal{U}_M})$ It takes $\{\mathsf{sk}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{ct}_i\}_{i\in\mathcal{U}_M}$ and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$ where $\mathcal{U}_K \subseteq \mathcal{ID}$ and $\mathcal{U}_M \subseteq \mathcal{ID}$ are any sets.

Correctness. A DDFE scheme for f is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $\mathcal{U}_K \subseteq \mathcal{ID}$, $\mathcal{U}_M \subseteq \mathcal{ID}$, $\{i, k_i\}_{i\in\mathcal{U}_K} \in \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{K})^i$, $\{i, m_i\}_{i\in\mathcal{U}_M} \in \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{M})^i$, we have

$$\Pr\left[d = f(\{i,k_i\}_{i\in\mathcal{U}_K}, \{i,m_i\}_{i\in\mathcal{U}_M}) : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{pk}_i, \mathsf{msk}_i \leftarrow \mathsf{LSetup}(\mathsf{pp}) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{msk}_i, m_i) \\ \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, k_i) \\ d = \mathsf{Dec}(\{\mathsf{sk}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{ct}_i\}_{i\in\mathcal{U}_M}) \end{array}\right] = 1.$$

Note that we can consider the case where $\mathcal{U}_K$ and $\mathcal{U}_M$ are multisets as in the original definition in [CDSG+20]. However, we do not consider the case here since it induces ambiguity that can be also found in [CDSG+20][8]. We assume that $\mathbb{N}$ contains 0 here and $(\mathcal{ID}\times\mathcal{K})^0 = \{i, k_i\}_{i\in\emptyset} = \emptyset$. That is, $\mathcal{U}_K$ and $\mathcal{U}_M$ can be an empty set, which corresponds to the case where Dec does not take secret keys/ciphertexts as input.

Security. We define the security of DDFE as follows. A DDFE scheme is xx-yy-partially hiding (xx ∈ {sel, adt}, yy ∈ {sym, asym}) if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\beta \leftarrow \mathcal{A}^{\mathsf{QHonestGen}(),\mathsf{QCor}(),\mathsf{QEnc}^\beta(),\mathsf{QKeyGen}()}(\mathsf{pp}) : \begin{array}{l} \beta \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Each oracle works as follows. For $i \in \mathcal{ID}$, $\mathsf{QHonestGen}(i)$ runs $(\mathsf{pk}_i, \mathsf{msk}_i) \leftarrow \mathsf{LSetup}(\mathsf{pp})$ and returns pk_i. For i such that $\mathsf{QHonestGen}(i)$ was queried, the adversary can make the following queries: $\mathsf{QCor}(i)$ outputs msk_i, $\mathsf{QEnc}^\beta(i, m^0, m^1)$ outputs $\mathsf{Enc}(\mathsf{msk}_i, m^\beta)$, and $\mathsf{QKeyGen}(i, k)$ outputs $\mathsf{KeyGen}(\mathsf{msk}_i, k)$. Note that m^β consists

[8] Concretely, when $\mathcal{U}_K$ is a multiset, and $i' \in \mathcal{U}_K$ has multiplicity 2, how to treat $k_{i'} \in \{k_i\}_{i\in\mathcal{U}_K}$ is unclear.

of the private elements $m^{\beta}_{\mathsf{priv}}$ and the public elements m_{pub}, respectively (we always require that $m^0_{\mathsf{pub}} = m^1_{\mathsf{pub}} = m_{\mathsf{pub}}$ as the public elements are not hidden in ct). Let $\mathcal{S}$ be the set of parties on which $\mathsf{QHonestGen}(i)$ is queried, $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = \mathcal{S} \backslash \mathcal{CS}$. Then, the adversary's queries must satisfy the following conditions.

- There are no sequences $(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^0_i\}_{i\in\mathcal{U}_M}), \{i, m^1_i\}_{i\in\mathcal{U}_M})$ that satisfy all the conditions:
  - For all $i \in \mathcal{U}_K$, [$\mathsf{QKeyGen}(i, k_i)$ is queried] or [$i \in \mathcal{CS}$].
  - For all $i \in \mathcal{U}_M$, [$\mathsf{QEnc}^{\beta}(i, m^0_i, m^1_i)$ is queried] or [$m^0_i = m^1_i \in \mathcal{M}$ and $i \in \mathcal{CS}$].
  - $f(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^0_i\}_{i\in\mathcal{U}_M}) \neq f(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^1_i\}_{i\in\mathcal{U}_M})$.
- When xx = sel: the adversary first generates a set $\mathcal{S}$ of honest users in one shot. After that it makes the corruption, key generation, encryption queries in one shot to obtain $\{\mathsf{msk}_i\}$, $\{\mathsf{KeyGen}(\mathsf{msk}_i, k)\}$, $\{\mathsf{Enc}(\mathsf{ek}_i, m^{\beta})\}$.
- When yy = sym: for $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^{\beta}(i, m^0, m^1)$ must satisfy $m^0 = m^1$[9].

We formally define DDFE for AWS as follows.

Definition 5.18 (DDFE for Attribute Weighted Sum)**.** DDFE for AWS is a class of DDFE (Definition 5.17) where $\mathcal{ID} \subseteq \{0,1\}^*$, $\mathcal{K} = \bigcup_{S\subseteq\mathcal{ID}} (\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S \times S$[10], $\mathcal{M} = \mathcal{X} \times 2^{\mathcal{ID}} \times \mathcal{L}$, where $\mathcal{X} = \bigcup_{i\in\mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$, and supports the following functionality: Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The function f' is defined as follows: for $k_i = (\bar{f}_i, \mathcal{U}_{K,i}) \in \mathcal{K}$, where $\bar{f}_i = \{f_j\}_{j\in\mathcal{U}_{K,i}}$ and $m_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i) \in \mathcal{M}$ (here $\{\mathbf{z}_{i,j}\}_{j\in[N_i]}$ is the private part and $\{\mathbf{x}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i$ are the public parts of m_i),

$$f'(\{i, k_i\}_{i\in\mathcal{U}'_K}, \{i, m_i\}_{i\in\mathcal{U}'_M}) =$$

[9] The symmetric setting captures the case where msk_i can be used to not only encrypt/key generation but also decryption/decoding of $\mathsf{ct}_i/\mathsf{sk}_i$.

[10] An element in $(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S \times S$ is of the form $(\{f_i\}_{i\in S}, S)$. We note that in more precise notation, $(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S$ contains elements of the form $\{i, f_i\}_{i\in S}$, which itself carries information about S, but we explicitly add $\times S$, to keep the notation more intuitive.

$$\begin{cases} [\sum_{i\in\mathcal{U}'_K}\sum_{j\in[N_i]}\langle f_i(\mathbf{x}_{i,j}),\mathbf{z}_{i,j}\rangle]_T & \text{the condition below is satisfied} \\ \bot & \text{otherwise} \end{cases}$$

1. $\mathcal{U}'_K = \mathcal{U}'_M$ and $\forall\, i \in \mathcal{U}'_K, \mathcal{U}_{K,i} = \mathcal{U}_{M,i} = \mathcal{U}'_K$.
2. $\forall_{i,i'\in\mathcal{U}'_K}, \bar{f}_i = \bar{f}_{i'}$ and $L_i = L_{i'}$.

For a building block of DDFE for AWS, we use a class of DDFE called all-or-nothing encryption. Chotard *et al.* showed that sel-sym-IND-secure AoNE can be generically constructed from identity-based encryption [CDSG+20].

Definition 5.19 (All-or-nothing encryption (AoNE))**.** AoNE is a class of DDFE (Definition 5.17) where $\mathcal{ID} = \{0,1\}^*$, $\mathcal{M}_{\mathsf{priv}} = \{0,1\}^L$ for some $L \in \mathbb{N}$, $\mathcal{M}_{\mathsf{pub}} = 2^{\mathcal{ID}} \times \mathcal{L}$, $\mathcal{K} = \emptyset$, $\mathcal{Z} = \{0,1\}^*$. The function f is defined as, for $\mathcal{U}'_K \in 2^{\mathcal{ID}}$ and $\{m_i = (x_i, \mathcal{U}_{M,i}, L_{M,i})\}_{i\in\mathcal{U}'_M}$,

$$f(\{i\}_{i\in\mathcal{U}'_K}, \{i, m_i\}_{i\in\mathcal{U}'_M}) = \begin{cases} \{x_i\}_{i\in\mathcal{U}'_M} & \text{the condition below is satisfied} \\ \bot & \text{otherwise} \end{cases}$$

- $\forall i \in \mathcal{U}'_M, \mathcal{U}'_M = \mathcal{U}_{M,i}$.
- $\exists L_M \in \mathcal{L}, \forall i \in \mathcal{U}'_M, L_{M,i} = L_M$.

This means that KeyGen is unnecessary, and Dec works without taking secret keys as input in AoNE (recall that $\mathcal{U}'_K$ can be an empty set).

5.8.2 Construction

Let aFE = (aSetup, aEnc, aKeyGen, aDec) be an FE scheme for AWSw/IP with the length of the random tape for aSetup being ℓ_a, AoNE = (anGSetup, anLSetup, anEnc, anDec) be an all-or-nothing encryption scheme[11], NIKE = (nSetup, nKeyGen, nSharedKey) be a non-interactive key exchange scheme, $\{\mathsf{PRF}_1^{\mathsf{K}}\} : 2^{\mathcal{ID}} \times \mathcal{L} \to \mathbb{Z}_p^k$,

[11] We use AoNE to encrypt messages from two different spaces. So, either we can use two AoNE schemes with appropriate message spaces or can use padding to make the message spaces same. For simplicity, we present our construction with same AoNE scheme.

$\{\mathsf{PRF}_2^{\mathsf{K}}\} : 2^{\mathcal{ID}} \to \{0,1\}^{\ell_s}$ be families of pseudorandom functions, with key space $\mathcal{K}_1, \mathcal{K}_2$, respectively and $\mathcal{ID}$ denotes an identity space and $H : \{0,1\}^* \to G_2^k$ is a hash function modeled as a random oracle. Our construction of DDFE for AWS is given below.

$\mathsf{GSetup}(1^\lambda)$ On input the security parameter 1^λ, the setup algorithm outputs pp as follows.

$$\mathsf{anPP} \leftarrow \mathsf{anGSetup}(1^\lambda),\ \mathsf{nPP} \leftarrow \mathsf{nSetup}(1^\lambda),\ \mathsf{pp} = (\mathsf{anPP}, \mathsf{nPP}).$$

$\mathsf{LSetup}(\mathsf{pp})$ On input pp, user $i \in \mathcal{ID}$ generates $(\mathsf{pk}_i, \mathsf{msk}_i)$ via the setup algorithm as follows.

$$(\mathsf{nPK}_i, \mathsf{nSK}_i) \leftarrow \mathsf{nKeyGen}(\mathsf{nPP}),\ (\mathsf{anPK}_i, \mathsf{anMSK}_i) \leftarrow \mathsf{anLSetup}(\mathsf{anPP}),\ \mathsf{K}_{i,2} \leftarrow \mathcal{K}_2$$
$$\mathsf{pk}_i = (\mathsf{nPK}_i, \mathsf{anPK}_i),\ \mathsf{msk}_i = (\mathsf{nSK}_i, \mathsf{anMSK}_i, \mathsf{K}_{i,2}).$$

$\mathsf{Enc}(\mathsf{msk}_i, m)$ The encryption algorithm takes as input the public parameters pp, the master secret key msk_i, and an input $m = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j \in N_i} \mathcal{U}_{M,i}, L_i)$ such that $i \in \mathcal{U}_{M,i}$ and outputs ct_i as follows.

$$\mathsf{rt}_i = \mathsf{PRF}_2^{\mathsf{K}_{i,2}}(\mathcal{U}_{M,i}),\ \mathsf{aMSK}_i = \mathsf{aSetup}(1^\lambda; \mathsf{rt}_i),\ \mathsf{K}_{i,j,1} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_i, \mathsf{nPK}_j)$$
$$\mathbf{v}_i = \sum_{\substack{j \in \mathcal{U}_{K,i} \\ i \neq j}} (-1)^{j<i} \mathsf{PRF}_1^{\mathsf{K}_{i,j,1}}(\mathcal{U}_{M,i}, L_i),\ \hat{x}_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j \in N_i}, [\mathbf{v}_i, 0]_1),$$
$$\mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, \hat{x}_i) \tag{5.8}$$
$$\mathsf{anCT}_i \leftarrow \mathsf{anEnc}(\mathsf{anMSK}_i, (\mathsf{aCT}_i, \mathcal{U}_{M,i}, L_i)),\ \mathsf{ct}_i = (\mathsf{anCT}_i, \mathcal{U}_{M,i}, L_i). \tag{5.9}$$

$\mathsf{KeyGen}(\mathsf{msk}_i, k)$ The key generation algorithm takes the master secret key msk_i, and

an input $k = (\{f_j\}_{j\in\mathcal{U}_{K,i}}, \mathcal{U}_{K,i})$ such that $i \in \mathcal{U}_{K,i}$ and outputs sk_i as follows.

$$\mathsf{rt}_i = \mathsf{PRF}_2^{\mathsf{K}_{i,2}}(\mathcal{U}_{K,i}),\ \mathsf{aMSK}_i = \mathsf{aSetup}(1^\lambda; \mathsf{rt}_i)$$
$$[\mathbf{s}]_2 = H(\{f_i\}_{i\in\mathcal{U}_{K,i}}, \mathcal{U}_{K,i}),\ \hat{f}_i = (f_i, [(\mathbf{s}, 0)]_2),\ \mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, \hat{f}_i) \tag{5.10}$$

$$\mathsf{anCT}_i \leftarrow \mathsf{anEnc}(\mathsf{anMSK}_i, (\mathsf{aSK}_i, \mathcal{U}_{K,i}, \{f_j\}_{j\in\mathcal{U}_{K,i}})),\ \mathsf{sk}_i = (\mathsf{anCT}_i, \mathcal{U}_{K,i}, \{f_j\}_{j\in\mathcal{U}_{K,i}}). \tag{5.11}$$

$\mathsf{Dec}(\{\mathsf{sk}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{ct}_i\}_{i\in\mathcal{U}_M})$ The decryption algorithm takes as input the public parameters pp, secret keys $\{\mathsf{sk}_i\}_{i\in\mathcal{U}_K}$, ciphertexts $\{\mathsf{ct}_i\}_{i\in\mathcal{U}_M}$ such that $\mathcal{U} = \mathcal{U}_K = \mathcal{U}_M$ and outputs d as follows. Parse $\mathsf{sk}_i = (\mathsf{anCT}_i, \mathcal{U}_{K,i}, \{f_j\}_{j\in\mathcal{U}_{K,i}})$ and $\mathsf{ct}_i = (\mathsf{anCT}'_i, \mathcal{U}_{M,i}, L_i)$. Compute

$$\{\widetilde{\mathsf{aSK}_i}\}_{i\in\mathcal{U}} = \mathsf{anDec}(\{\mathsf{anCT}_i\}_{i\in\mathcal{U}}),\ \{\widetilde{\mathsf{aCT}_i}\}_{i\in\mathcal{U}} = \mathsf{anDec}(\{\mathsf{anCT}'_i\}_{i\in\mathcal{U}}),$$
$$[d]_T = \prod_{i\in\mathcal{U}} \mathsf{aDec}(\widetilde{\mathsf{aSK}_i}, \widetilde{\mathsf{aCT}_i}).$$

Correctness. Firstly, we observe that if $\mathcal{U}_K = \mathcal{U}_M = \mathcal{U}$, $L_i = L_M$ for all $i \in \mathcal{U}$, where L_M is any label in $\mathcal{L}^2$ and $\{f_j\}_{j\in\mathcal{U}_{K,i}}$ is same in all the ciphertexts input to the decryption algorithm, then

- we have from the correctness of AoNE, $\widetilde{\mathsf{aSK}_i} = \mathsf{aSK}_i$ and $\widetilde{\mathsf{aCT}_i} = \mathsf{aCT}_i$.
- vector $\mathbf{s}$ computed by every user $i \in \mathcal{U}$ is same.

Then from the correctness of NIKE, $\mathsf{K}_{i,j,1} = \mathsf{K}_{j,i,1}$ and hence, $\sum_{i\in\mathcal{U}} \mathbf{v}_i = 0$. Hence, from the correctness of aFE,

$$\prod_{i\in\mathcal{U}} \mathsf{aDec}(\widetilde{\mathsf{aSK}_i}, \widetilde{\mathsf{aCT}_i}) = \prod_{i\in\mathcal{U}} \mathsf{aDec}(\mathsf{aSK}_i, \mathsf{aCT}_i) = \prod_{i\in\mathcal{U}} [\sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \langle \mathbf{s}, \mathbf{v}_i\rangle]_T$$
$$= [\sum_{i\in\mathcal{U}} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \sum_{i\in\mathcal{U}} \langle \mathbf{s}, \mathbf{v}_i\rangle]_T = [\sum_{i\in\mathcal{U}} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i.j}\rangle]_T.$$

5.8.3 Security

We argue security via the following theorem.

Theorem 5.23. *If* $\{\mathsf{PRF}_1^{\mathsf{K}}\}, \{\mathsf{PRF}_2^{\mathsf{K}}\}$ *are families of pseudorandom functions,* NIKE *is IND-secure,* AoNE *is sel-sym-IND-secure, the* MDDH_k *assumption holds in* $\mathbb{G}$*, and* aFE *is function-hiding, then our AWS-DDFE scheme is sel-sym-partially-hiding in the random oracle model as per Definition 5.17.*

Proof. Let $\mathcal{S}$ be the set of parties generated by QHonestGen queries. Let $\mathcal{HS} \subseteq \mathcal{S}$ be the set of uncorrupted parties and $\mathcal{CS} = \mathcal{S} \backslash \mathcal{HS}$. We prove the theorem via a series of hybrids, which are defined as follows.

H_s^β This is the original game. In particular, in response to $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ and $\mathsf{QKeyGen}(i, \{f_j\}_{j\in\mathcal{U}_K}, \mathcal{U}_K)$, where $x_i^b = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^b\}_{j\in[N_i]}$ for $b \in \{0,1\}$, the challenger sets

$$\hat{x}_i = (x_i^\beta, [\mathbf{v}_i, 0]_1), \quad \hat{f}_i = (f_i, [\mathbf{s}, 0]_2),$$

(in eqs. (5.8) and (5.10), respectively). Vectors $\mathbf{v}_i$ and $\mathbf{s}$ are computed as desribed in the construction.

H_1^β In this hybrid, the challenger samples rt_i randomly instead of computing using PRF_2. Indistinguishability between H_s^β and H_1^β follows from the security of PRF_2.

H_2^β We say an encryption query on $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ is incomplete, if there exists $i' \in \mathcal{U}_M$ such that $i' \in \mathcal{HS}$ and no encryption query of the form $(i', \star, \star, \mathcal{U}_M, L_M)$ is made. In this hybrid, in response to all the incomplete encryption queries, anCT_i is computed as $\mathsf{anEnc}.(0, \mathcal{U}_M, L_M)$ (eq. (5.9)). The indistinguishability between H_1^β and H_2^β follows from the security of AoNE.

H_3^β We say a key query on $(i, \bar{f} = \{f_j\}_{j\in\mathcal{U}_K}, \mathcal{U}_K)$ is incomplete if there exist $i' \in \mathcal{U}_K$

such that $i' \in \mathcal{HS}$ and there is no key query of the form $(i', \bar{f}, \mathcal{U}_K)$. In this hybrid, for all the incomplete key queries anCT_i encrypts 0 (eq. (5.11)). The indistinguishability between H_2^β and H_3^β follows from the security of AoNE.

H_f^β In this hybrid, for all the complete encryption queries of the form $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ with $i \in \mathcal{HS}$, the challenger sets $\hat{x}_i = (x_i^0, [\mathbf{v}_i, 0]_1)$. We note that the adversary has zero advantage in this hybrid because its view is independent of β (recall that for $i \in \mathcal{CS}, x_i^0 = x_i^1$.). We show that H_f^β is indistinguishable from H_3^β in Lemma 5.24

Lemma 5.24. *If* $\{\mathsf{PRF}_1^\mathsf{K}\}$ *is a family of pseudorandom functions,* NIKE *is* IND*-secure, the* MDDH_k *assumption holds in* $\mathbb{G}$*, and* aFE *is partially function-hiding, then* $\mathsf{H}_3^\beta \approx \mathsf{H}_f^\beta$ *in the random oracle model.*

Proof. To prove the lemma, we consider the following sub hybrids between H_3^β and H_f^β. Let q_u be the total number of ID sets with complete encryption queries. Let $\{\mathcal{U}_1, \ldots, \mathcal{U}_{q_u}\}$ be some fixed ordering on the ID sets from complete encryption queries and let q_u' be the upper bound on q_u. Then define sub hybrid $\widehat{\mathsf{H}}_j^\beta$ as follows

$\widehat{\mathsf{H}}_j^\beta$ (for $j \in \{0\} \cup [q_u']$). This hybrid is same as H_3^β except that for every complete encryption query of the form $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ such that $i \in \mathcal{HS}$, the challenger sets

$$\hat{x}_i = \begin{cases} (x_i^0, [\mathbf{v}_i, 0]_1) & \text{if } \mathcal{U}_M \in \{\mathcal{U}_1, \ldots, \mathcal{U}_j\} \\ (x_i^\beta, [\mathbf{v}_i, 0]_1) & \text{if } \mathcal{U}_M \in \{\mathcal{U}_{j+1}, \ldots, \mathcal{U}_{q_u}\}, \end{cases}$$

where $\mathcal{U}_j = \{\perp\}$ for $j > q_u$. We observe that $\widehat{\mathsf{H}}_0^\beta = \mathsf{H}_3^\beta$ and $\widehat{\mathsf{H}}_{q_u'}^\beta = \mathsf{H}_f^\beta$. So, now we need to show that for all $j \in [q_u']$, $\widehat{\mathsf{H}}_{j-1}^\beta \approx \widehat{\mathsf{H}}_j^\beta$.

To show this, we let $\{L_{\mathcal{U}_j}^1, \ldots, L_{\mathcal{U}_j}^v\}$ be the set of labels used in complete encryption queries of the form $(\star, \star, \star, \mathcal{U}_j, \star)$. Let $v \leq q_L$. Then define the following sub hybrids:

$\widehat{\mathsf{H}}_{j-1,0}^\beta$ Same as $\widehat{\mathsf{H}}_{j-1}^\beta$.

$\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa}$ (for $\kappa \in [q_L]$). Same as $\widehat{\mathsf{H}}^{\beta}_{j-1}$ except that for every complete encryption query of the form $(i, x_i^0, x_i^1, \mathsf{U}_j, L)$, for $i \in \mathcal{HS}$,

$$\hat{x}_i = \begin{cases} (x_i^0, [\mathbf{v}_i, 0]_1) & \text{if } L \in \{L^1_{\mathcal{U}_j}, \ldots, L^{\kappa}_{\mathcal{U}_j}\} \\ (x_i^{\beta}, [\mathbf{v}_i, 0]_1) & \text{if } L \in \{L^{\kappa+1}_{\mathcal{U}_j}, \ldots, L^{v}_{\mathcal{U}_j}\} \end{cases}$$

We observe that $\widehat{\mathsf{H}}^{\beta}_{j-1,q_L} = \widehat{\mathsf{H}}^{\beta}_{j}$. So now, we need to show that $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa-1} \approx \widehat{\mathsf{H}}^{\beta}_{j-1,\kappa}$, for all $\kappa \in [q_L]$. For this, we further define following sub hybrids between $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa-1}$ and $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa}$:

Let $\mathcal{U}_j^{\mathcal{HS}} = \mathcal{U}_j \cap \mathcal{HS} = \{u_1, \ldots, u_w\}$ and w' be an upper bound on w. Define

$\bar{\mathsf{H}}^{\beta}_{\eta}$ (for $\eta \in [w']$). Same as $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa-1}$, except that for each complete encryption query $\mathsf{QEnc}^{\beta}(u_i, x^0_{u_i}, x^1_{u_i}, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$ and complete key query $\mathsf{QKeyGen}(u_i, \{f_j\}_{j \in \mathcal{U}_j}, \mathcal{U}_j)$, $\hat{x}_{u_i}$ and $\hat{f}_{u_i}$, respectively, are set as follows:

$$\hat{x}_{u_i} = \begin{cases} (x^0_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i \leq \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } \eta < i < w \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i < w \\ (f_{u_i}, [\mathbf{s}, \sum_{l \in [\eta]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

Here, $\Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}} = f_{u_l}(x^{1,\beta}_{u_l}) - f_{u_l}(x^{1,0}_{u_l})$, where 1 in the superscript indicates the first QEnc^{β} query of the form $(u_l, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$. We have, from the admissibility conditions,

- Let $q_{c,u_l,\mathcal{U}_j,L^{\kappa}_{\mathcal{U}_j}}$ be the number of encryption queries of the form $(u_l, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, then $f_{u_l}(x^{\tau,\beta}_{u_l}) - f_{u_l}(x^{\tau,0}_{u_l}) = \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}$ for all $\tau \in [q_{c,u_l,\mathcal{U}_j,L^{\kappa}_{\mathcal{U}_j}}]$, where τ denotes the sequence number of the query of this form (see proof of Lemma 5.19).
- $\sum_{u_l \in (\mathcal{HS} \cap \mathcal{U}_j)} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}} = 0$.

Now we argue the indistinguishability of the sub hybrids. Firstly, we observe the

following:

1. $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa-1} \approx \bar{\mathsf{H}}^{\beta}_{0}$: The only difference between the two hybrids is that for encryption query of the form $\mathsf{QEnc}^{\beta}(u_w, x^0_{u_w}, x^1_{u_w}, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, $\hat{x}_{u_w} = (x^{\beta}_{u_w}, [\mathbf{v}_{u_w}, 0]_1)$ in the former and $\hat{x}_{u_w} = (x^{\beta}_{u_w}, [\mathbf{v}_{u_w}, 1]_1)$ in the latter hybrid. Note that $\hat{f}_{u_w}$ for any key queries of the form $(u_w, \{f_j\}_{j\in[\mathcal{U}_j]}, \mathcal{U}_j)$ is of the form $(f_{u_w}, [\mathbf{s}, 0]_2)$ (notice the last bit being 0) in both the hybrids. Hence, the two hybrids are indistinguishable due to partially function-hiding security of aFE.

2. Similarly, $\bar{\mathsf{H}}^{\beta}_{w'} \approx \widehat{\mathsf{H}}^{\beta}_{j-1,\kappa}$ from aFE security.

So, all that is left is to show that $\bar{\mathsf{H}}^{\beta}_{\eta-1} \approx \bar{\mathsf{H}}^{\beta}_{\eta}$. For this, we first note that the two hybrids differ only in the values of $\hat{x}_{u_\eta}$ and $\hat{f}_{u_w}$ as follows:

In $\bar{\mathsf{H}}^{\beta}_{\eta-1}$:

$$\hat{x}_{u_i} = \begin{cases} (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}, \quad \hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta-1]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

In $\bar{\mathsf{H}}^{\beta}_{\eta}$:

$$\hat{x}_{u_i} = \begin{cases} (x^{0}_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}, \quad \hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

To show indistinguishability, we consider sub hybrids with the following sequence of changes in $\hat{x}_{u_i}$ and $\hat{f}_{u_i}$ for $u_i \in \mathcal{U}^{\mathcal{HS}}_j$.

$\bar{\mathsf{H}}^{\beta}_{\eta-1,1}$ For every complete QEnc^{β} query, sample $K_{u_\eta,u_w,1} (= K_{u_w,u_\eta,1})$ randomly instead of computing from $\mathsf{nSharedKey}$[12]. Indistinguishability from $\bar{\mathsf{H}}^{\beta}_{\eta-1}$ follows from the security of NIKE.

$\bar{\mathsf{H}}^{\beta}_{\eta-1,2}$ For any complete encryption query of the form $(u_i, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, the

[12] this change will happen for all the ID sets, since $K_{u_\eta,u_w,1}$ does not depend on the ID set or the label.

computation of $\mathbf{v}_{u_\eta}$ and $\mathbf{v}_{u_w}$ use random value in place of $\mathsf{PRF}_1^{K_{u_\eta,u_w,1}(\mathcal{U}_j,L^\kappa_{\mathcal{U}_j})}$. Thus, vectors $\mathbf{v}_{u_\eta}$ and $\mathbf{v}_{u_w}$ changes from

$$\begin{aligned}
\mathbf{v}_{u_\eta} &= \sum_{i\in\mathcal{U}_j, i\neq u_\eta} (-1)^{i<u_\eta}\mathsf{PRF}^{K_{u_\eta,i,1}(\mathcal{U}_j,L^\kappa_{\mathcal{U}_j})}, \text{ to}\\
\mathbf{v}_{u_\eta} &= \sum_{i\in\mathcal{U}_j, i\notin\{u_\eta,u_w\}} (-1)^{i<u_\eta}\mathsf{PRF}^{K_{u_\eta,i,1}(\mathcal{U}_j,L^\kappa_{\mathcal{U}_j})} \underline{+\mathbf{t}_{u_\eta,u_w}}\\
\mathbf{v}_{u_w} &= \sum_{i\in\mathcal{U}_j, i\neq u_w} (-1)^{i<w}\mathsf{PRF}^{K_{u_w,i,1}(\mathcal{U}_\ell,L^\kappa_{\mathcal{U}_j})}, \text{ to}\\
\mathbf{v}_{u_w} &= \sum_{i\in\mathcal{U}_j, i\notin\{u_\eta,u_w\}} (-1)^{i<u_w}\mathsf{PRF}^{K_{u_w,i,1}(\mathcal{U}_\ell,L^\kappa_{\mathcal{U}_j})} \underline{-\mathbf{t}_{u_\eta,u_w}}
\end{aligned}$$

where $\mathbf{t}_{u_\eta,u_w}$ is chosen randomly. Indistinguishability from the previous sub hybrid follows from the security of PRF_1.

$\bar{\mathsf{H}}^\beta_{\eta-1,3}$ Change $\hat{x}_{u_\eta}$, $\hat{x}_{u_w}$, $\hat{f}_{u_\eta}$ and $\hat{f}_{u_w}$ as:

$$\hat{x}_{u_i} = \begin{cases} (x^\beta_{u_i}, [\mathbf{v}_{u_i} \underline{+\mathbf{t}_{u_\eta,u_w}}, 1]_1) & \text{if } i=\eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} \underline{-\mathbf{t}_{u_\eta,u_w}}, 1]_1) & \text{if } i=w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, \underline{-\langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w}\rangle}]_2) & \text{if } i=\eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta-1]} \Delta^\beta_{u_l,L^\kappa_{\mathcal{U}_j}} \underline{+\langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w}\rangle}]_2) & \text{if } i=w \end{cases}$$

The indistinguishability follows from the partially function-hiding property of aFE.

$\bar{\mathsf{H}}^\beta_{\eta-1,4}$ Replace $\langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w}\rangle$ with random t_{u_η,u_w}.

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^\beta_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i=\eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i=w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, -\underline{t_{u_\eta,u_w}}]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta-1]} \Delta^{\beta}_{u_l,L^{\kappa}_{\mathcal{U}_j}} + \underline{t_{u_\eta,u_w}}]_2) & \text{if } i = w \end{cases}$$

Indistinguishability between $\bar{\mathsf{H}}^{\beta}_{\eta-1,3}$ and $\bar{\mathsf{H}}^{\beta}_{\eta-1,4}$ follows from the MDDH_k assumption. In more detail, let $\bar{f}^1, \ldots, \bar{f}^{q_k}$ be the functions for which the adversary issues complete key queries of the form $(\star, \star, \mathcal{U}_j)$ queries and let $\mathbf{s}^1, \ldots, \mathbf{s}^{q_k}$ be the corresponding $\mathbf{s}$ vectors (recall that these are computed from the hash function modeled as a random oracle). Then, to argue indistinguishability between the two hybrids, we need to show

$$\{\mathbf{s}^\tau, \langle \mathbf{s}^\tau, \mathbf{t}_{u_\eta,u_w} \rangle\}_{\tau\in[q_k]} \approx \{t^\tau_{u_\eta,u_w}\}_{\tau\in[q_k]},$$

which follows directly from the MDDH_k assumption.

$\bar{\mathsf{H}}^{\beta}_{\eta-1,5}$ Implicitly set $t_{u_\eta,u_w} = t'_{u_\eta,u_w} + \Delta_{u_\eta,L^{\kappa}_{\mathcal{U}_j}}$. That is,

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^{\beta}_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, -\underline{t'_{u_\eta,u_w} - \Delta_{u_\eta,L^{\kappa}_{\mathcal{U}_j}}}]_2) & \text{if } i = \eta \\ (f_{u_w}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^{\beta}_{u_l,L^{\kappa}_{\mathcal{U}_j}} + \underline{t'_{u_\eta,u_w}}]_2) & \text{if } i = w \end{cases}$$

$\bar{\mathsf{H}}^{\beta}_{\eta-1,6}$ Change $\hat{x}_{u_\eta}$ and $\hat{f}_\eta$ as

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^{0}_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, -\underline{t'_{u_\eta,u_w}}]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^{\beta}_{u_l,L^{\kappa}_{\mathcal{U}_j}} + t'_{u_\eta,u_w}]_2) & \text{if } i = w \end{cases}$$

Work	Parties	EK Cor.	Label	(Pub, Pri) CT	Key	Functionality	Function classes of f, g, h
AB-FE ACGU20	1	✓	N/A	$(x, \mathbf{z})$	$(f, \mathbf{c})$	$f(x) \cdot \langle \mathbf{z}, \mathbf{c} \rangle$	$f \in$ MSPs
FE for AWS AGW20	1	✓	N/A	$(\{\mathbf{x}_j\}_j, \{\mathbf{z}_j\}_j)$	f	$\sum_{j\in[N]} f(\mathbf{x}_j)^\top \mathbf{z}_j$	$f \in$ ABPs
MIFE AGT22	n	✓	×	$(\perp, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c}, \mathbf{z} \otimes \mathbf{z} \rangle$	N/A
MIFE AGT22	n	×	✓	$(\perp, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c}, \mathbf{z} \otimes \mathbf{z} \rangle$	N/A
AB-MIFEACGU20	n	✓	×	$(\perp, \mathbf{z}_i)$	$\{y_i, \mathbf{c}_i\}_{i\in S}$	$f(\{y_i\}_{i\in S}) \cdot \sum_{i\in S} \langle \mathbf{z}_i, \mathbf{c}_i \rangle$	$f(\{y_i\}) = \bigwedge_{i\in S} g_i(y_i)$ fixed $g_i \in$ MSPs
AB-MIFE, Sec. 5.6,5.6.3	n	✓	×	$((\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j), \{\mathbf{z}_{i,j}\}_j)$	$\{g_i, h_i\}_{i\in[n]}$	$f(\mathbf{y}) \cdot \sum_{i\in[n]} \sum_{j\in[N_i]} h_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f(\mathbf{y}) = \bigwedge_i (g_i(\mathbf{y}_i) = 0)$ $g_i, h_i \in$ ABPs
MCFE CDG+18b; ABG19	n	✓	✓	$(\perp, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c}, \mathbf{z} \rangle$	N/A
AB-MCFE NPP22	n	✓	OT	$(x_i, \mathbf{z}_i)$	$\{g_i, \mathbf{c}_i\}$	$f(\{x_i\}) \cdot \langle \mathbf{c}, \mathbf{z} \rangle$	$f(\{x_i\}) = \bigwedge_i g_i(x_i)$ $g_i \in$ LSS
MCFE, Sec. 5.7	n	✓	✓	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i\in[n]}$	$\sum_{i\in[n]} \sum_{j\in[N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f_i \in$ ABPs
DDFE, CDSG+20; AGT21b	Unbdd n	✓	✓	$(\perp, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c}, \mathbf{z} \rangle$	N/A
DDFE, Sec. 5.8	Unbdd n	✓	✓	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i\in S}$	$\sum_{i\in S} \sum_{j\in[N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f_i \in$ ABPs

Table 5.3: Prior state of the art and our results. We do not consider function hiding here. Above, we denote $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ or $\mathbf{z} = (\mathbf{z}_i)_{i\in S}$. $S \subseteq [n]$ is some subset of authorized users for a given key. EK Cor. refers to whether an adversary is allowed to obtain encryption keys in the security game. Label refers to the capability of labeling functionality that restricts decryption such that it is allowed only when all labels are equal. OT in label means that each label can be used only one time per input. MSPs/ABPs/LSS stand for monotone span programs/arithmetic branching programs/linear secret sharing. MCFE in a stronger (resp. weaker) notion corresponds to MIFE that satisfies EK Cor. and Label (resp. One-time label).

Indistinguishability follows from the partially function hiding property of aFE. Now, undo the changes in previous steps to get $\bar{\mathsf{H}}^{\beta}_{\eta}$. ■

■

APPENDIX

5.A DETAILED COMPARISON WITH PRIOR WORK

Here we provide a detailed summary of the related prior work in multi-party FE schemes.

5.B MULTI-PARTY FUNCTIONAL ENCRYPTION

In this chapter, we use many classes of functional encryption (FE) such as attribute-based encryption, secret-key functional encryption, multi-input encryption, etc. To capture various notions of FE, Agrawal, Goyal, and Tomida proposed a notion called multi-party functional encryption (MPFE) [AGT21b]. The following definition is verbatim from [AGT21b].

Definition 5.20 (Multi-Party Functional Encryption)**.** Let n_x be the number of ciphertext inputs and n_y be the number of key inputs. Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ be the space of ciphertext inputs and $\mathcal{Y} = \mathcal{Y}_{\mathsf{pub}} \times \mathcal{Y}_{\mathsf{priv}}$ be the space of key inputs. We define two aggregation functions as $\mathsf{Agg}_x : \mathcal{X}^{n_x} \to \mathcal{X}^*$, and $\mathsf{Agg}_y : \mathcal{Y}^{n_y} \to \mathcal{Y}^*$.

An MPFE scheme is defined as a tuple of 4 algorithms/protocols (Setup, KeyGen, Enc, Dec). To suitably capture existing primitives, we define our Setup algorithm/protocol to run in three modes, described next.

Setup modes. The Setup algorithm/protocol can be run in different modes: central, local, or interactive. For $\mathsf{mode} \in \{\mathsf{central}, \mathsf{local}, \mathsf{interactive}\}$, consider the following.

central Here the Setup algorithm is run by one trusted third party which outputs the master secret keys and encryption keys for all users in the system.

local Here it is run independently by different parties without any interaction, and each party outputs its own encryption key and/or master secret key.

interactive Here it is an interactive protocol run by a set of users, at the end of which, each user has its encryption key and/or master secret key. We note that these keys may be correlated across multiple users.

A multi-party functional encryption (MPFE) consists of the following:

$\mathsf{Setup}\left(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y\right)$ This algorithm/protocol can be executed in any one of

the three modes described above.Given input the security parameter, number of ciphertext inputs n_x, number of key inputs n_y and two aggregation functions Agg_x, Agg_y as defined above, this algorithm outputs a set of encryption keys $\{\mathsf{ek}_i\}_{i \le n_x}$, master secret keys $\{\mathsf{msk}_i\}_{i \le n_y}$ and public key pk.

$\mathsf{Enc}\left(\mathsf{pk}, \mathsf{ek}, i, x = \left(x_{\mathsf{pub}}, x_{\mathsf{priv}}\right)\right)$ Given input the public key pk, an encryption key ek, user index $i \in [n_x]$, an input $x = \left(x_{\mathsf{pub}}, x_{\mathsf{priv}}\right)$, this algorithm outputs a ciphertext ct_x.

$\mathsf{KeyGen}\left(\mathsf{pk}, \mathsf{msk}, j, y = \left(y_{\mathsf{pub}}, y_{\mathsf{priv}}\right)\right)$ Given input the public key pk, a master secret key msk, user index $j \in [n_y]$ and a function input $y = \left(y_{\mathsf{pub}}, y_{\mathsf{priv}}\right)$, this algorithm outputs a secret key sk_y.

$\mathsf{Dec}\left(\mathsf{pk}, \{\mathsf{sk}_j\}_{j \le n_y}, \{\mathsf{ct}_i\}_{i \le n_x}\right)$ Given input the public key pk, a set of secret keys $\{\mathsf{sk}_j\}_{j \le n_y}$ and a set of ciphertexts $\{\mathsf{ct}_i\}_{i \le n_x}$, this algorithm outputs a value z or $\perp$.

We remark that in the *local* setup mode, it will be helpful to separate the setup algorithm into a global setup, denoted by Gsetup along with a local setup, denoted by Lsetup, where the former is used only to generate common parameters of the system, such as group descriptions and such.

Correctness. We say that an MPFE scheme is *correct* if, $\forall (n_x, n_y) \in \mathbb{N}^2$, ciphertext inputs $x_i \in \mathcal{X}$ for $i \in [n_x]$, key inputs $y_j \in \mathcal{Y}$ for $j \in [n_y]$, message and function

aggregation circuits Agg_x and Agg_y, it holds that:

$$\Pr\left[z = z' : \begin{array}{l} (\mathsf{pk}, \{\mathsf{ek}_i\}, \{\mathsf{msk}_j\}) \leftarrow \\ \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{ek}_i, i, x_i)\ \forall i \in [n_x] \\ \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{pk}, \mathsf{msk}_j, j, y_j)\ \forall j \in [n_y] \\ z \leftarrow \mathsf{Dec}\left(\mathsf{pk}, \{\mathsf{sk}_j\}_{j \le n_y}, \{\mathsf{ct}_i\}_{i \le n_x}\right) \\ z' = \mathcal{U}\left(\mathsf{Agg}_x(\{x_i\}), \mathsf{Agg}_y(\{y_j\})\right) \end{array}\right] = 1.$$

Recall that $\mathcal{U}$ is the universal circuit with appropriate input and output size.

Indistinguishability based security. Next, we define security of MPFE. The security definition is modelled in a similar fashion to MIFE security [GGG+14, sec. 2.2] while taking into account corruption queries.

For any choice of parameters λ, n_x, n_y, aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$, and master keys $\mathsf{K} = (\mathsf{pk}, \{\mathsf{ek}_i\}_{i\in[n_x]}, \{\mathsf{msk}_j\}_{j\in[n_y]}) \leftarrow \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y)$, we define the following list of oracles:

$\mathsf{QCor}^{\mathsf{K}}(\cdot)$, upon a call to this oracle for any $i \in [n_x]$ or $j \in [n_y]$, the adversary gets the corresponding encryption key ek_i or master secret key msk_j. In the case of a local setup, the adversary could instead also supply the oracle with adversarially generated keys for the corresponding user; whereas in case of an interactive setup, the adversary could simulate the behavior of the queried user index in the setup protocol. (Let $\mathcal{S}_x \subseteq [n_x]$ and $\mathcal{S}_y \subseteq [n_y]$ denote the set of user indices for which the corresponding encryption and master keys have been corrupted.)

$\mathsf{QEnc}^{\mathsf{K},\beta}(\cdot,\cdot)$, upon a call to this oracle for an honest user index $i \in [n_x]$, message inputs $(x_i^{\ell,0}, x_i^{\ell,1})$ (where $x_i^{\ell,b} = \left(x_{i,\mathsf{pub}}^{\ell,b}, x_{i,\mathsf{priv}}^{\ell,b}\right)$ for $b \in \{0,1\}$), the challenger first checks whether the user i was already corrupted or not. That is, if $i \in \mathcal{S}_x$, then it sends nothing, otherwise it samples a ciphertext for input $x_i^{\ell,\beta}$ using key ek_i and sends it to the adversary.

$\mathsf{QKey}^{\mathsf{K},\beta}(\cdot,\cdot)$, upon a call to this oracle for an honest user index $j \in [n_y]$, function inputs $(y_j^{k,0}, y_j^{k,1})$ (where $y_j^{k,b} = \left(y_{j,\mathsf{pub}}^{k,b}, y_{j,\mathsf{priv}}^{k,b}\right)$ for $b \in \{0,1\}$), the challenger first checks whether the user j was already corrupted or not. That is, if $j \in \mathcal{S}_y$, then it sends nothing, otherwise it samples a decryption key for function input $y_j^{k,\beta}$ using key msk_j and sends it to the adversary. (Here β is the challenge bit chosen at the start of the experiment.)

We let Q_x and Q_y be the number of encryption and key generation queries (respectively) that had non-empty responses. Let $\mathcal{Q}_x = \{(i, (x^{\ell,0}, x^{\ell,1}))\}_{\ell \in [Q_x]}$ be the set of ciphertext queries and $\mathcal{Q}_y = \{(j, (y_j^{k,0}, y_j^{k,1}))\}_{k \in [Q_y]}$ be the set of key queries.

We say that an adversary $\mathcal{A}$ is *admissible* if:

1. For each of the encryption and key challenges, the public components of the two challenges are equal, namely $x_{\mathsf{pub}}^{\ell,0} = x_{\mathsf{pub}}^{\ell,1}$ for all $\ell \in [Q_x]$, and $y_{\mathsf{pub}}^{k,0} = y_{\mathsf{pub}}^{k,1}$ for all $k \in [Q_y]$.

2. For each of the encryption and key challenges, the *private* components of the two challenges are also equal, namely $x_{\mathsf{priv}}^{\ell,0} = x_{\mathsf{priv}}^{\ell,1}$ for all $\ell \in [Q_x]$ whenever $(i, (x^{\ell,0}, x^{\ell,1})) \in \mathcal{Q}_x$ and $i \in \mathcal{S}_x$, and $y_{\mathsf{priv}}^{k,0} = y_{\mathsf{priv}}^{k,1}$ for all $k \in [Q_y]$ whenever $(j, (y^{\ell,0}, y^{\ell,1})) \in \mathcal{Q}_y$ and $j \in \mathcal{S}_y$. That is, the private components must be the same as well if the user index i or j, that the query was made for, was corrupted during the execution.[13]

3. There do not exist two sequences $(\overrightarrow{x}^0, \overrightarrow{y}^0)$ and $(\overrightarrow{x}^1, \overrightarrow{y}^1)$ such that:

$$\mathcal{U}\left(\mathsf{Agg}_x(\{x_i^0\}), \mathsf{Agg}_y(\{y_j^0\})\right) \neq \mathcal{U}\left(\mathsf{Agg}_x(\{x_i^1\}), \mathsf{Agg}_y(\{y_j^1\})\right)$$

   and i) for every $i \in [n_x]$, either x_i^b was queried or ek_i was corrupted, and ii) for every $j \in [n_y]$, either y_j^b was queried or msk_j was corrupted, and iii) at least one of inputs $= x_i^b, = y_j^b$ were queried and indices i, j were not corrupted. (Note that if $i \in [n_x]$ or $j \in [n_y]$ were queried to the QCor oracle, the adversary can generate partial keys or ciphertexts for any value of its choice.)

An MPFE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be IND secure if for any *admissible* PPT adversary $\mathcal{A}$, all length parameters $n_x, n_y \in \mathbb{N}$, and aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following

[13] This condition is an option. When we would like to claim that $\mathsf{ek}_i/\mathsf{msk}_i$ does not help to decode $\mathsf{ct}_i/\mathsf{sk}_i$, item 2 should be removed.

holds

$$\Pr\left[b' = \beta : \begin{array}{c} \mathsf{K} \leftarrow \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y), \\ \mathsf{K} = (\mathsf{pk}, \{\mathsf{ek}_i\}_i, \{\mathsf{msk}_j\}_j), \\ \beta \leftarrow \{0,1\}, \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}^{\mathsf{K}}(\cdot), \mathsf{QKey}^{\mathsf{K},\beta}(\cdot), \mathsf{QEnc}^{\mathsf{K},\beta}(\cdot)}(1^\lambda, \mathsf{pk}) \end{array}\right] \leq \frac{1}{2} + \mathrm{negl}\,.$$

Remark 11 (Weaker notions of security)**.** We say the scheme is selective IND secure if the adversary outputs the challenge message and function pairs at the very beginning of the game, before it makes any queries or receives the pk. One may also consider the semi-honest setting, where the QCor oracle is not provided, or the case of static corruptions where the adversary provides all its corruptions once and for all at the start of the game.

5.B.1 Dynamic Multi-Party Functional Encryption

In this section, we define the dynamic notion of multi-party functional encryption (MPFE). We consider the fully dynamic setting where the number of key/ciphertext inputs is unspecified during setup time, and the aggregation functions are also specified only during key generation and encryption times. In the dynamic setting, an interactive or centalized setup is not meaningful since the number of parties is itself not known during setup time, hence we restrict ourselves to the local setup mode for simplicity.

Definition 5.21 (Dynamic Multi-Party Functional Encryption)**.** Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ be the space of ciphertext inputs and $\mathcal{Y} = \mathcal{Y}_{\mathsf{pub}} \times \mathcal{Y}_{\mathsf{priv}}$ be the space of key inputs. Also, let $\mathcal{PK}$ be the space to which each local public key belongs. A dynamic multi-party functional encryption scheme (MPFE) with local setup is defined as a tuple of 5 algorithms/protocols ($\mathsf{Gsetup}, \mathsf{Lsetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) with the following syntax:

$\mathsf{Gsetup}(1^\lambda)$ On input the security parameter, the global setup algorithm samples a globally shared set of public parameters pp.

$\mathsf{Lsetup}(\mathsf{pp})$ Given input the public parameters, the local setup algorithm outputs a tuple

consisting of local public key pk, an encryption key ek, and a master secret key msk. (Here the local public key is just regarded as a public identifier for the user, and not given as explicit input to other algorithms since it could always be added to the encryption and/or master secret key.)

$\mathsf{Enc}\left(\mathsf{ek}, i, x = \left(x_{\mathsf{pub}}, x_{\mathsf{priv}}\right), \mathsf{Agg}_x\right)$ Given input an encryption key ek, user index $i \in [n_x]$, an input $x = \left(x_{\mathsf{pub}}, x_{\mathsf{priv}}\right)$, and an aggregation function $\mathsf{Agg}_x : (\mathcal{PK} \times \mathcal{X})^{n_x} \to \mathcal{X}^*$ (for some $n_x \in \mathbb{N}$), this algorithm outputs a ciphertext ct_i.

$\mathsf{KeyGen}\left(\mathsf{msk}, j, y = \left(y_{\mathsf{pub}}, y_{\mathsf{priv}}\right), \mathsf{Agg}_y\right)$ Given input a master secret key msk, user index $j \in [n_y]$

$\mathsf{Dec}\left((\mathsf{sk}_j)_j, (\mathsf{ct}_i)_i\right)$ Given input a sequence of secret keys $(\mathsf{sk}_j)_j$ and a sequence of ciphertexts $(\mathsf{ct}_i)_i$, this algorithm outputs a value z or $\perp$.

Correctness. We say that an MPFE scheme is *correct* if, $\forall (N, n_x, n_y) \in \mathbb{N}^3$, ciphertext inputs $x_i \in \mathcal{X}$ for $i \in [n_x]$, key inputs $y_j \in \mathcal{Y}$ for $j \in [n_y]$, message and function aggregation circuits Agg_x and Agg_y, and indexing functions $\mathsf{index}_x : [n_x] \to [N]$, $\mathsf{index}_y : [n_y] \to [N]$ it holds that:

$$\Pr\left[z = z' : \begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Gsetup}(1^\lambda) & \\ (\mathsf{pk}_\ell, \mathsf{ek}_\ell, \mathsf{msk}_\ell) \leftarrow \mathsf{Lsetup}(\mathsf{pp}) & \forall \ell \in [N] \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_{\mathsf{index}_x(i)}, i, x_i, \mathsf{Agg}_x) & \forall i \in [n_x] \\ \mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}_{\mathsf{index}_y(j)}, j, y_j, \mathsf{Agg}_y) & \forall j \in [n_y] \\ z \leftarrow \mathsf{Dec}\left((\mathsf{sk}_j)_{j \le n_y}, (\mathsf{ct}_i)_{i \le n_x}\right) & \\ z' = \mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{pk}_{\mathsf{index}_x(i)}, x_i)_i\right), \mathsf{Agg}_y\left((\mathsf{pk}_{\mathsf{index}_y(j)}, y_j)_j\right)\right) & \end{array}\right] = 1.$$

Recall that $\mathcal{U}$ is the universal circuit with appropriate input and output size.

Indistinguishability based security. Here we extend the security experiment for multi-party functional encryption that we provided in Definition 5.20 to the dynamic user setting in the local setup mode. Since we are working in the dynamic setting, we

need to define the following oracles

$\mathsf{HonestGen}()$, upon a call to this oracle, the challenger samples a fresh tuple of local public key, encryption key, and master key $(\mathsf{pk}, \mathsf{ek}, \mathsf{msk})$, and stores them in a list $\mathcal{L}_{\mathsf{setup}}$. It sends pk to the adversary. (Note that if the scheme is a public key scheme, then the challenger sends the encryption key ek to the adversary.)

$\mathsf{QCor}(\cdot, \cdot)$, upon a call to this oracle for an honest user local public key pk and key type $\mathsf{type} \in= \mathsf{enc}, \mathsf{master}$, the challenger first checks whether the list $\mathcal{L}_{\mathsf{setup}}$ contains a key pair associated with pk. If there is such a key pair $(\mathsf{pk}, \mathsf{ek}, \mathsf{msk})$, then it sends either the ek or msk depending on the type queried. Otherwise, it sends nothing.[14]

$\mathsf{QEnc}^{\beta}(\cdot, \cdot, \cdot, \cdot)$, upon a call to this oracle for an honest user local public key pk, inputs $(x_j^{\ell,0}, x_j^{\ell,1})$ (where $x_j^{\ell,b} = \left(x_{j,\mathsf{pub}}^{\ell,b}, x_{j,\mathsf{priv}}^{\ell,b}\right)$ for $b \in \{0,1\}$), index j, aggregation function $\mathsf{Agg}_{x,j}^{\ell}$, the challenger first checks whether the list $\mathcal{L}_{\mathsf{setup}}$ contains a key pair associated with pk. If there is such a key pair $(\mathsf{pk}, \mathsf{ek}, \mathsf{msk})$, then it samples a ciphertext for input $x_j^{\ell,\beta}$ using key ek and sends it to the adversary. Otherwise, it sends nothing. (Here β is the challenge bit chosen at the start of the experiment.)

$\mathsf{QKey}^{\beta}(\cdot, \cdot, \cdot, \cdot)$, upon a call to this oracle for an honest user local public key pk, function inputs $(y_j^{k,0}, y_j^{k,1})$ (where $y_j^{k,b} = \left(y_{j,\mathsf{pub}}^{k,b}, y_{j,\mathsf{priv}}^{k,b}\right)$ for $b \in \{0,1\}$), index j, aggregation function $\mathsf{Agg}_{y,j}^{k}$, the challenger first checks whether the list $\mathcal{L}_{\mathsf{setup}}$ contains a key pair associated with pk. If there is such a key pair $(\mathsf{pk}, \mathsf{ek}, \mathsf{msk})$, then it samples a decryption key for function input $y_j^{k,\beta}$ using key msk and sends it to the adversary. Otherwise, it sends nothing. (Here β is the challenge bit chosen at the start of the experiment.)

We let Q_x and Q_y be the number of encryption and key generation queries (respectively) that had non-empty responses. Let $Q_x = \{(\mathsf{pk}^{\ell}, (x_j^{\ell,0}, x_j^{\ell,1}), j, \mathsf{Agg}_{x,j}^{\ell})\}_{\ell \in [Q_x]}$ be the set

[14] As we point out in the static setting, in case ek_i is completely contained in some msk_i (or vice versa), then making a master secret corruption query for i will also imply that encryption key for i has been corrupted too (and vice versa).

of ciphertext challenge queries and $Q_y = \{(\mathsf{pk}^k, (y_j^{k,0}, y_j^{k,1}), j, \mathsf{Agg}_{y,j}^k)\}_{k\in[Q_y]}$ be the set of key challenge queries.

We say that an adversary $\mathcal{A}$ is *admissible* if:

1. For each of the encryption and key challenges, the public components of the two challenges are equal, namely $x_{j,\mathsf{pub}}^{\ell,0} = x_{j,\mathsf{pub}}^{\ell,1}$ for all $\ell \in [Q_x]$, and $y_{j,\mathsf{pub}}^{k,0} = y_{j,\mathsf{pub}}^{k,1}$ for all $k \in [Q_y]$.

2. For each of the encryption and key challenges, the *private* components of the two challenges are also equal, namely $x_{j,\mathsf{priv}}^{\ell,0} = x_{j,\mathsf{priv}}^{\ell,1}$ for all $\ell \in [Q_x]$, and $y_{j,\mathsf{priv}}^{k,0} = y_{j,\mathsf{priv}}^{k,1}$ for all $k \in [Q_y]$ if the encryption key ek^ℓ or the master secret key msk^k, that the query was made for, was corrupted during the execution (respectively).

3. There do not exist two sequences $((\overrightarrow{\mathsf{pk}}_x, \overrightarrow{x}^0), (\overrightarrow{\mathsf{pk}}_y, \overrightarrow{y}^0)) \neq ((\overrightarrow{\mathsf{pk}}_x, \overrightarrow{x}^1), (\overrightarrow{\mathsf{pk}}_y, \overrightarrow{y}^1))$ and aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$ such that:

$$\mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{pk}_{x,i}, x_i^0)_i\right), \mathsf{Agg}_y\left((\mathsf{pk}_{y,j}, y_j^0)_j\right)\right) \neq \mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{pk}_{x,i}, x_i^1)_i\right), \mathsf{Agg}_y\left((\mathsf{pk}_{y,j}, y_j^1)_j\right)\right)$$

   and i) x_i^b was queried for aggregation function Agg_x, index i and public key $\mathsf{pk}_{x,i}$, and ii) y_j^b was queried for aggregation function Agg_y, index j and public key $\mathsf{pk}_{y,j}$, and iii) at least one of inputs $= x_i^b, = y_j^b$ were queried and public key $\mathsf{pk}_{x,i}, \mathsf{pk}_{y,j}$ was not corrupted. Note that if some x_i^b or y_j^b was not queried by the adversary, then it can generate partial keys or ciphertexts for any value of its choice by performing a fresh key generation since this is a fully dynamic system, however that samples a fresh public as well.

An MPFE scheme $(\mathsf{Gsetup}, \mathsf{Lsetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be IND secure if for any *admissible* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{HonestGen}(), \mathsf{QCor}(), \mathsf{QKey}^\beta(), \mathsf{QEnc}^\beta()}(1^\lambda) = \beta : \beta \leftarrow \{0,1\}\right] \leq \frac{1}{2} + \mathrm{negl}\,.$$

Remark 12 (Potential variations)**.** The above multi-party function encryption system that we define allows the users to dynamically join the system in the permissionless model, where each incoming user only needs to know the public parameters and not interact with

any authority. A slightly weaker setting could be a permissioned model in which users can still dynamically join the system but they need to contact the global authority (which sampled the public parameters) either for some identification tokens or its encryption and master secret key pair in order to prevent totally unrestricted computation which happens in the permissionless model.

Also, we want to point out that in our current framework we let the users select the aggregation functions during individual functional key and ciphertext generation to allow for more flexibility. This could be relaxed even further by letting the aggregation functions be either be described in a uniform computation model, or using an ensemble of non-uniform functions. Also, one could instead restrict the flexibility in aggregation by asking each user to choose their aggregation functions at setup time. Such flexibilities will be important in capturing the notion of DDFE described in Definition 5.17.

5.B.2 Capturing our primitives in the MPFE framework

They also proposed a dynamic variant of MPFE, which is presented in Section 5.B.1. In this work, we use following variants of FE subsumed by MPFE or dynamic MPFE. Formal definitions of these are found in Section 5.4.3 or respective sections.

Attribute-Based Encryption. Attribute-based encryption (ABE) for predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is captured by MPFE as follows: $(n_x, n_y) = (1,1)$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = (x', M)$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (y', \perp)$. $\mathsf{Agg}_x(x) = x$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}(x, f) = M$ if $\mathsf{P}(x', y') = 1$ and $\mathcal{U}(x, f) = \perp$ otherwise.

Secret-Key Functional Encryption. Secret-key functional encryption (SK-FE) for function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (1,1)$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = (x_1, x_2)$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x) = x$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}(x, f) = f(x)$.

Multi-Input Functional Encryption. Multi-input functional encryption (MIFE) for

function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (n, 1)$, $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = (x_{i,1}, x_{i,2})$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}((x_1, \ldots, x_n), f) = f(x_1, \ldots, x_n)$.

Multi-Client Functional Encryption. Multi-client functional encryption (MCFE) for function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (n, 1)$, $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = ((x_{i,1}, L_i), x_{i,2})$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}((x_1, \ldots, x_n), f) = f((x_{1,1}, x_{1,2}), \ldots, (x_{n,1}, x_{n,2}))$ if and only if $L_1 = \cdots = L_n$.

Dynamic Decentralized Functional Encryption. Dynamic decentralized functional encryption (DDFE) for function F is captured by dynamic MPFE (Section 5.B.1) as follows: $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = (m_{i,1}, m_{i,2}) = m_i$, $y_i = (y_{i,\mathsf{pub}}, y_{i,\mathsf{priv}}) = (k_{i,1}, k_{i,2}) = k_i$. Agg_x is an identity function, and $\mathsf{Agg}_y(\{\mathsf{pk}_i, k_i\}_{i \in \mathcal{U}_K})$ outputs f such that $\mathcal{U}(\{\mathsf{pk}_i, m_i\}_{i \in \mathcal{U}_M}, f) = F(\{\mathsf{pk}_i, m_i\}_{i \in \mathcal{U}_M}, \{\mathsf{pk}_i, k_i\}_{i \in \mathcal{U}_K})$. Note that we assume that aggregate functions here can be described in non-uniform computation model such as Turing machines as in Remark 12.

Attribute-Based FE for Attribute-Weighted Sums with Inner Product We can capture Attribute-Based FE for AWS with Inner Product in the context of MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The setup algorithm is run in the central mode, and $n_x = n_y = 1$. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}), (\{\mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1))$ where $\mathbf{y}, \mathbf{x}_j, \mathbf{z}_j, \mathbf{p}$ are all vectors in $\mathbb{Z}_p$ while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = ((g, h), [\mathbf{q}]_2)$ where g, h are ABPs, and $\mathbf{q}$ is a vector in $\mathbb{Z}_p$. Agg_x is an identity function, and Agg_y outputs a function $f_{g,h,[\mathbf{q}]_2}$ that outputs $[\sum_{j \in [N]} \langle h(\mathbf{x}_i), \mathbf{z}_i \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T$ if and only if $g(\mathbf{y}) = 0$ on input $x = ((\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}), (\{\mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1))$.

Attribute-Based MIFE for Attribute-Weighted Sums We can capture Attribute-Based MIFE for AWS in the context of MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$

be bilinear groups. The setup algorithm is run in the central mode, and $n_x = n, n_y = 1$ for some $n \in \mathbb{N}$. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\mathbf{y}, \{\mathbf{x}_j\}_{j\in[N]}), \{\mathbf{z}_j\}_{j\in[N]})$ where $\mathbf{y}, \mathbf{x}_j, \mathbf{z}_j$ are all vectors in $\mathbb{Z}_p$ while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (\{g_i, h_i\}_{i\in[n]}, \perp)$ where g_i, h_i are ABPs. Agg_x is an identity function, and Agg_y outputs a function $f_{\{g_i,h_i\}_{i\in[n]}}$ that outputs $[\sum_{i\in[n]} \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle]_T$ if and only if $g_i(\mathbf{y}_i) = 0$ for all $i \in [n]$ on input $\{x_i\}_{i\in[n]} = \{(\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j\in[N_i]}), \{\mathbf{z}_{i,j}\}_{j\in[N_i]}\}_{i\in[n]}$.

Dynamic Decentralized Functional Encryption for AWS We can capture DDFE for AWS in the context of dynamic MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The setup algorithm is run in the local mode, since it works in a dynamic manner. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\{\mathbf{x}_j\}_{j\in[N]}, \mathcal{U}_M, L_M), \{\mathbf{z}_j\}_{j\in[N]})$ where $\mathbf{x}_j, \mathbf{z}_j$ are vectors in $\mathbb{Z}_p$, $\mathcal{U}_M$ is a set of IDs, and L_M is a label while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = ((\{f_i\}_{i\in\mathcal{U}_K}, \mathcal{U}_K), \perp)$ where f is an ABP. Agg_x checks if the public inputs $(\mathcal{U}_M, L_M)$ match for all parties and that all the ciphertexts are provided for the set $\mathcal{U}_M$. If so, it outputs $(\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{i\in\mathcal{U}_M, j\in[N_i]}, \mathcal{U}_M)$. Agg_y checks $(\{f_i\}_{i\in\mathcal{U}_K}, \mathcal{U}_K)$ match for all parties and that all the ciphertexts are provided for the set $\mathcal{U}_K$. If so, it outputs a function f'_y that outputs $[\sum_{i\in\mathcal{U}_K} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle]_T$ if and only if $\mathcal{U}_M = \mathcal{U}_K$ on input $(\mathcal{U}_M, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{i\in\mathcal{U}_M, j\in[N_i]})$.

CHAPTER 6

ROUND-OPTIMAL LATTICE-BASED THRESHOLD SIGNATURES

6.1 INTRODUCTION

In this chapter we describe our constructions of round-optimal threshold signatures from lattice based assumptions. Threshold signatures [Des94] is a generalization of digital signatures where the signature issuing capacity is distributed among several users, so that a signature can be generated only if a sufficient number of users collaborate to sign a message. In more detail, each of N parties holds a partial signing key, and any set of parties at least as large as a given threshold $t \leq N$ can participate in a protocol to generate a signature. Security requires that a valid signature cannot be generated if fewer than t parties cooperate.

While threshold signatures have been studied for a long time [Lin17b; DKLS18; CCL+19; GGN16; GG18; LN18; DKLS19; DOK+20; CCL+20; CGG+20; GKSŚ20; DJN+20; GG20; BKP13], they have received renewed attention in recent years due to numerous applications in modern topics such as cryptocurrencies and blockchains. Most prior work has focused on creating distributed versions of classical digital signature schemes, ECDSA or Schnorr signatures [LN18; GG18; DKLS19; CCL+19; CCL+20] which are not quantum secure. From conjectured post-quantum assumptions such as those related to Euclidean lattices, much less is known, especially with optimal round complexity.

Prior Work. The thresholdisation of lattice-based signatures from the NIST post-quantum cryptography project has been investigated in [CS19] but the resulting candidates incur several rounds of communication. A threshold signature restricted to $t = N$ was proposed in [DOTT21] but it also involves possibly many rounds, because of aborts.

To the best of our knowledge, the only lattice-based, round-optimal threshold signature construction is by Boneh *et al* [BGG+18] (henceforth BGGJKRS), relying on the Learning With Errors problem (LWE). However, while this construction provided the first feasibility result for a long-standing open problem, it suffers from the following drawbacks:

1. *Noise Flooding and Impact on Parameters.* It makes use of the so-called "noise flooding" technique [Gen09; BD10; GKPV10], which aims to hide a noise term $e \in \mathbb{Z}$ that possibly contains sensitive information, by adding to it a fresh noise term e' whose distribution has a standard deviation that is much larger than an a priori upper bound on $|e|$. To get security against attackers with success probability $2^{-o(\lambda)}$ where λ is the security parameter, the standard deviation of e' must be a factor $2^{\Omega(\lambda)}$ larger than the upper bound on $|e|$.

   Unfortunately, this precludes the use of an efficient LWE parametrisation. Concretely, one has to set the LWE noise rate α as $2^{-\Omega(\lambda)}$ so that $|e'|$ remains small compared to the working modulus q. As the best known algorithms for attacking LWE with (typical) parameters n, q, α have run-times that grow as $\exp(\widetilde{O}(n \log q / \log^2 \alpha))$ (see, e.g., [HKM18]) this leads to setting $n \log q = \widetilde{\Omega}(\lambda^3)$. As the signature shares have bit-sizes that grow as $\Omega(n \log q)$, this leads to $\widetilde{\Omega}(\lambda^3)$-bit signature sizes – prohibitively expensive in practice.

2. *Instantiating Underlying Signature.* It requires a standard signature scheme to be evaluated homomorphically. BGGJKRS do not suggest a candidate and existing lattice based signatures are not suitable – the GPV signature scheme [GPV08] and its practical versions [DP16; PP19; FHK+] seem ill-suited, as the signing algorithm is very sequential, and the required 1-dimensional Gaussian samples are obtained via algorithms based on rejection sampling (see, e.g., [HPRR20; ZSS20]) that are costly to transform into circuits. The other candidate is Lyubashevsky's signature scheme [Lyu09; Lyu12]. It has the advantage of being far less sequential, but it also relies on rejection sampling: when some rejection test does not pass, then one needs to restart the signing process.

3. *Selective Security.* It only achieves a very restricted notion of *selective* security, where all the corrupted parties must be announced before any partial signing query is made. To obtain security in the more realistic *adaptive* setting, one option is to invoke complexity leveraging, which consists in guessing at the outset which parties will be corrupted. This is not only dissatisfying as a solution but also leads to a further degradation of the parameters.

6.2 OUR RESULTS

We improve the construction from [BGG+18] in different ways. We list them below:

- *Efficiency.* We decrease the noise flooding ratio from $2^{\Omega(\lambda)}$ down to $\sqrt{Q}$, where Q is the bound on the number of generated signatures. This gives a one-round threshold signature of bit-length growing as $\widetilde{O}(\lambda \log^2 Q)$, which is $\widetilde{O}(\lambda)$ for any polynomially bounded Q,[1] in contrast with $\widetilde{O}(\lambda^3)$ for the construction from [BGG+18]. These bit-lengths are obtained when relying on the ring variants of SIS and LWE [LM06; PR06; SSTX09; LPR10]. Additionally, we show that the amount of noise flooding used in this construction is *optimal*, by exhibiting an attack when a smaller noise flooding ratio is used.

- *Instantiation.* To instantiate the signature underlying BGGJKRS, we provide a homomorphism friendly variant of Lyubashevsky's signature [EUROCRYPT '12] which achieves low circuit depth. We remove the rejection sampling at the expense of adding moderate noise of size $\sqrt{Q}$, matching the above. Again, we show that this amount of flooding is optimal by demonstrating an attack when smaller flooding is used.

- *Selective versus Adaptive.* As discussed above, the construction BGGJKRS satisfies only selective security. We improve this in two ways: in the Random Oracle Model (ROM), in which a hash function is being modeled as a uniformly sampled function with the same domain and range, we obtain a notion of *partial adaptivity* where signing queries can be made before the corrupted parties are announced. However, the set of corrupted parties must be announced *all at once*. In the standard model, we obtain a construction with full adaptivity, where parties can be corrupted at any stage in the protocol. However, this construction is in a weaker *pre-processing* model where signers must be provided correlated randomness of length proportional to the number of signing queries. The informed reader may notice similarities with the "MPC with Preprocessing" model, please see [FKOS15] and references therein [2].

[1] For many applications, the bound Q is quite limited and can be considered to be a small polynomial in λ. For example, for applications pertaining to cryptocurrencies, the bound Q may capture the total number of transactions made with a user's wallet during the lifetime of a signing key. According to statistics available at the URLs below, one transaction per day and per user is a generous upper bound. This suggests that number of signing queries in the lifecycle of the key will be quite limited. https://www.blockchain.com/charts/n-transactions, https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/

[2] Note that we can trade the offline sharing of correlated randomness with an additional communication round in the signing protocol – however, this would destroy round optimality.

6.3 TECHNICAL OVERVIEW

Recap of BGGJKRS Threshold Signatures. The round-optimal threshold signatures provided by [BGG+18] are designed using a "universal thresholdizer" which enables the thresholdizing of a number of primitives. This thresholdizer is itself instantiated using a threshold version of "special" fully homomorphic encryption (FHE), which in turn can be constructed using the LWE assumption. In threshold fully homomorphic encryption (TFHE), the setup algorithm takes as input a threshold t and produces a set of decryption key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ for the parties such that every party can perform a partial decryption using its own decryption key and any t out of N partial decryptions can be combined into a complete decryption of the ciphertext in a single round.

In more detail, the TFHE construction of BGGJKRS leverages the fact that the decryption in LWE based FHE schemes [BV11; BGV12; GSW13] requires to compute an inner product of the ciphertext ct with the secret key sk, followed by a rounding operation. Since inner product is a linear operation, a natural approach to thresholdize FHE decryption is by applying a Shamir t-out-of-N secret sharing to sk. This will yield N keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, which can be distributed to the N users. Now, to decrypt a ciphertext ct, each user can compute the inner product with its individual secret key sk_i as its partial decryption m_i. To combine any t partial decryptions into the final decryption, the combiner chooses Lagrange coefficients $\gamma_1, \ldots, \gamma_t$ so that $\sum_i \gamma_i \mathsf{sk}_i = \mathsf{sk}$. Then, she computes

$$\sum_i \gamma_i m_i = \sum_i \gamma_i \langle \mathsf{ct}, \mathsf{sk}_i \rangle = \langle \mathsf{ct}, \sum_i \gamma_i \mathsf{sk}_i \rangle = \langle \mathsf{ct}, \mathsf{sk} \rangle,$$

followed by rounding, as desired. However, this appealingly simple construction turns out to be insecure. This is because each time a party computes a partial decryption, it leaks information about its secret share sk_i via the inner product with (the public) value ct.

To get around this insecurity, a natural approach is to add noise to the partial decryption which quickly transforms a simple computation to intractable. However, care must

be taken to ensure that this added noise does not affect correctness, since it is later multiplied by the Lagrange coefficients during reconstruction: the previous $\sum_i \gamma_i m_i$ will now become $\sum_i \gamma_i(m_i + e_i)$ for some noise terms e_i. BGGJKRS propose two solutions – one to use a secret sharing scheme whose reconstruction coefficients are binary, and another, to "clear the denominators" by observing that since the Lagrange coefficients are rational numbers, it is possible to scale them to be integers. The exact details are not relevant for the current discussion and hence omitted (please refer to [BGG+18] for more details).

To use this technique to construct threshold signatures, the authors propose the following. Choose a signature scheme Sig, compute an FHE encryption $\mathsf{ct}_{\mathsf{sk}}$ of its signing key Sig.sk and let each signer homomorphically evaluate the signing algorithm for a message μ on this ciphertext. In more detail, given $\mathsf{ct}_{\mathsf{sk}} = \mathsf{FHE.Enc}(\mathsf{Sig.sk})$, each party first computes $\mathsf{FHE.Eval}(C, \mathsf{ct}_{\mathsf{sk}})$ where C is the circuit $\mathsf{Sig.Sign}(\mu, \cdot)$. By correctness of FHE, this yields an FHE encryption of the signature $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, \mu)$. To this ciphertext, the thresholdization trick described above may now be applied.

Modeling the Adversary and Effect on Parameters. In their analysis, BGGJKRS consider the complexity-theory security requirement of "no polynomial time attacks", corresponding to assuming attacks with advantage $\epsilon = \lambda^{-O(1)}$ and run-time $\lambda^{O(1)}$. However, for practically motivated primitives like threshold signatures, it is more meaningful to consider attackers with advantage $2^{-o(\lambda)}$ and run-time $2^{o(\lambda)}$. We choose our adversarial model so that all attacks should be exponential while all honest algorithms run in polynomial time. Compared to the complexity-theory definition of security, this provides a much more significant (and practically meaningful) hardness gap between honest and malicious parties.

For subexponentially strong attackers as described above, the noise flooding used in BGGJKRS is exponential, severely damaging the practicality of the scheme, despite the

exciting developments in practical FHE [CGGI20; ZDH20; KS21; CKKS17; CHK+18; DM15]. In more detail, the proof requires to make the statistical distance between some noise terms e' and $e + e'$ small, so that knowing $e + e'$ is essentially the same as knowing e', which does not carry sensitive information. To get security against attackers with advantage $2^{-o(\lambda)}$, the statistical distance must be set to $2^{-\Omega(\lambda)}$ and, as a result, the standard deviation of e' must be a factor $2^{\Omega(\lambda)}$ larger than the upper bound on $|e|$.

Tightening Analysis via Rényi divergence. In this work, we examine whether this flooding noise can be improved so that the impact of flooding e by e' on efficiency is minimised. To this end, we explore using *Rényi divergence* rather than statistical distance to bound the distance between distributions in the security proof. Rényi divergence has been used in prior work as a replacement to the statistical distance in lattice based cryptography [LSS14; LPSS14; BGM+16; Pre17; HLS18; ADPS16; BCD+16; AD17; BLRL+18]. To understand why this may be beneficial, let us first see how statistical distance is used in typical security proofs of cryptography. Let $\mathcal{P}$ and $\mathcal{Q}$ be two non-vanishing probability distributions over a common measurable support X. Typical security proofs consider a hard problem relying on some ideal distribution $\mathcal{Q}$, and then replace this ideal distribution by a real world distribution $\mathcal{P}$. When the statistical distance $\Delta(\mathcal{Q}, \mathcal{P})$ between the two distributions is small, the problem remains hard, implying security. This is made rigorous by the so-called "probability preservation" property which says that for any measurable event $E \subseteq X$, we have $\mathcal{Q}(E) \geq \mathcal{P}(E) - \Delta(\mathcal{Q}, \mathcal{P})$.

Let us now define Rényi Divergence (RD). For $a \in (1, \infty)$, the RD of order a is defined by $R_a(\mathcal{P}||\mathcal{Q}) = \left(\sum_{x \in X} \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}}\right)^{\frac{1}{a-1}}$. It enjoys an analogous probability preservation property, though multiplicative as against additive. For $E \subseteq X$, we have $\mathcal{Q}(E) \geq \mathcal{P}(E)^{\frac{a}{a-1}} / R_a(\mathcal{P}||\mathcal{Q})$. Thus, if an event E occurs with significant probability under $\mathcal{P}$, and if the SD or the RD is small, then the event E also occurs with significant probability under $\mathcal{Q}$. As discussed in [BLRL+18], probability preservation in SD is meaningful when the distance is smaller than any $\mathcal{P}(E)$ that the security proof is required to deal with

– if $\mathcal{P}(E) \geq \epsilon$ for some ϵ, then we require that $\Delta(\mathcal{Q}, \mathcal{P}) < \epsilon$. The analogous requirement for RD is $R_a(\mathcal{P}||\mathcal{Q}) \leq \mathsf{poly}(1/\epsilon)$. Bai et al. [BLRL$^+$18] observed that RD is often less demanding than SD in proofs. This is because RD between distributions may be small enough to suffice for RD probability preservation while SD may be too large for the SD probability preservation to be applicable. Thus, RD can often serve as a better tool for security analysis, especially in applications with search-type security definitions, like signatures.

In this work, we study the applicability of RD analysis in the construction of threshold signatures. Building upon the above approach, we show that a limited flooding growing as $\sqrt{Q}$ suffices in BGGJKRS, where Q is the number of signing queries made by the attacker. We note that this is a substantial improvement in practice, since the number of sign queries is typically very different, and much smaller, than the run time of the adversary. Note that signature queries require active participation by an honest user and there is no reason for an honest user to keep replying after an overly high number of queries that clearly shows adversarial behavior. As a concrete example, in the NIST post quantum project [NIS17], adversarial runtimes can go up to 2^{256} in some security levels, but the number of signature queries is always bounded by 2^{64} (which is itself an overly conservative bound in many scenarios). Thus, dependence on the number of queries is significantly better than exponential dependence on the security parameter, and this leads to a significant improvement in the signature bit size.

Optimality of our Moderate Flooding. We also show that this magnitude of flooding is necessary for this construction, by exhibiting a statistical attack when smaller noise is used. At a high level, our attack proceeds as follows. First we show that using legitimate information available to her, the adversary can compute $\mathsf{err}_M + e_{1,M}$ where err_M is the error that results from homomorphically evaluating the signing algorithm for message M and $e_{1,M}$ is the flooding noise that is used in the partial signature of the first party. As a warmup, consider the setting where the flooding noise is randomized. Now, since the

signature scheme is deterministic, the term err_M depends only on M and remains fixed across multiple queries for the same message. On the other hand, the term $e_{1,M}$ keeps changing. Using Hoeffding's bound, it is possible to estimate the average of $e_{1,M}$ across multiple queries and use this to recover err_M, leading to an attack.

This attack may be avoided by making the flooding noise a deterministic function of the message, e.g., by using a pseudo-random function evaluated on the message to generate the noise. We show that this modification is not sufficient to make the threshold signature construction secure. For this purpose, we design a signature scheme which includes "useless" information in the signature: this information does not affect correctness nor security of the signature itself, but allows us to recreate the attack described above on the resulting threshold signature. We start with a secure signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ whose signing key is a uniform bit-string among those with the same number of 0's and 1's. Now, let us consider a special signature scheme $\mathsf{Sig}' = (\mathsf{Sig}'.\mathsf{KeyGen}, \mathsf{Sig}'.\mathsf{Sign}, \mathsf{Sig}'.\mathsf{Verify})$ derived from Sig by modifying the signing algorithm as follows: for $i \in [|\mathsf{Sig.sk}|]$, if $\mathsf{Sig.sk}_i = 0$, then append a 0 to the signature. Since our signing key has exactly half as many 0's as 1's, this leads to a string of $|\mathsf{Sig.sk}|/2$ zeroes being appended to every signature: this does not leak any information and does not affect correctness (it is simply ignored during verification). Now, consider using Sig' to instantiate our threshold signature scheme. Then, for any message M, the FHE ciphertext CT_{σ_M} now additionally includes homomorphically evaluated encryptions of $\{\mathsf{Sig.sk}_i\}_{i \in [|\mathsf{Sig.sk}|]:\mathsf{Sig.sk}_i=0}$. Note that these extra encryptions are designed to be a deterministic function of the secret key so that across multiple messages, the corresponding error term (obtained via homomorphic evaluation) will not change. On the other hand, the message-dependent error terms can be assumed to change across messages. Due to this, the error term recovered by the adversary will be a sum of a fixed term (dependent only on the secret key) plus a fresh term per signature, which allows it to recreate the first attack. Please see Section 6.5 for more details.

Homomorphism-Friendly Signature. Next, we provide a variant of Lyubashevsky's signature scheme [Lyu12] which enjoys low circuit depth and is homomorphism friendly. As discussed above, Lyubashevsky's signature contains a rejection sampling step, whose purpose is to make the distribution of the resultant signature canonical, but this step is cumbersome to implement homomorphically. We show that by using RD analysis in place of statistical distance, analogously to the case of threshold signatures discussed above, the rejection sampling step can be replaced by noise flooding of moderate magnitude $\sqrt{Q}$. Additionally, we show that this flooding is optimal – please see Section 6.6 for details.

Towards Adaptive Security. Another limitation of the construction of BGGJKRS is that security is proved in the weak "selective" model where the adversary must announce all corrupted users before receiving the public parameters and verification key. In contrast, the more reasonable adaptive model allows the adversary to corrupt users based on the public parameters, the verification key and previous user corruptions it may have made. We briefly describe the difficulty in achieving adaptive security. At a high level, in the selective game, the challenger proceeds by simulating the partial keys corresponding to the honest parties in a "special way". The challenge in the adaptive setting is that without knowing who are the honest/corrupted parties, the challenger does not know which partial keys to program.

For more details, let us consider the case of an N-out-of-N threshold signature. In the simulation, the challenger knows which party is honest at the start of the game, e.g., player N. Now, the challenger can sample FHE secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$ randomly, implicitly setting the last share as $\mathsf{sk} - \sum_{i \in [N-1]} \mathsf{sk}_i$. To invoke the unforgeability of the underlying signature scheme Sig, the challenger must "forget" the signing key Sig.sk at some point in the proof, and rely on the Sig challenger to return signatures, which it then encrypts using the (public key) FHE scheme. By correctness of FHE, this is the same as computing the signing circuit for a given message on the ciphertext containing the secret key, which is what happens in the real world. However, the FHE encryption of signing

key $\mathsf{Sig.sk}$ is provided as part of the public parameters in the real world, which in turn means that the FHE secret key must be "forgotten" so that the FHE ciphertext of $\mathsf{Sig.sk}$ is indistinguishable from a dummy value. Yet the challenger must return legitimate partial signatures of queried messages m_j in the security game, which in turn are (noisy) partial decryptions of the FHE ciphertexts $\widehat{\sigma_j}$ of signatures σ_j. Knowing all the corrupt secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$ from the outset enables the challenger to walk this tightrope successfully – it obtains σ_j from the Sig challenger, computes an FHE encryption $\widehat{\sigma_j}$ of this, computes partial decryptions using $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$, floods these with appropriate noise and returns these to the adversary.

In the adaptive game, the honest player is not known at the beginning of the game so the challenger is unable to sample FHE secret key shares as described above. When requested for a partial signatures for message m_j, it can obtain the corresponding signature σ_j and can FHE encrypt it, but cannot decrypt it using secret key shares which are unavailable. To preserve correctness and indistinguishability from the real world, it is forced to return (noisy) random secret shares $\{\sigma_{i,j}\}_{i\in[N],j\in\mathsf{poly}}$ of σ_j as partial signatures, for unbounded j. Later if user 1 is corrupted (say), the adversary receives the secret key share sk_1. Now, to preserve indistinguishability, the challenger must explain the partial signatures $\{\sigma_{1,j}\}_{j\in\mathsf{poly}}$ corresponding to user 1 as $\langle\widehat{\sigma}_j, \mathsf{sk}_1\rangle \approx \sigma_{1,j}$, which seems impossible for unbounded j.

We overcome this hurdle in the ROM by having the challenger simulate all partial keys as though corresponding to a corrupt user and when the list of corrupted parties becomes available, "program" the ROM to "explain" the returned keys in a consistent way. This yields an intermediate notion of "partial adaptivity", in which the attacker can make signing queries before corruption, but must announce its corrupted users all at once. In more detail, we modify the signing key to additionally contain a random secret share of $\mathbf{0}$, i.e., each party is provided a vector $\mathbf{v}_i$ of length N, such that $\sum_{i\in[N]} \mathbf{v}_i = \mathbf{0}$. In the scheme, to compute a partial signature for a message m_j, the partial signing algorithm

first computes $r_{i,j} = H(j,K)^{\mathsf{T}}\ \mathbf{v}_i$ where $H(j,K)$ is a random vector of length N, and K is a secret value required for a technical reason that we will not discuss here. It then returns $\langle \widehat{\sigma}_j, \mathsf{sk}_i \rangle + \mathsf{noise}_{i,j} + r_{i,j}$. By linearity, it holds that $\sum_{i\in[N]} H(j,K)^{\mathsf{T}}\ \mathbf{v}_i = \mathbf{0}$, so correctness is not affected. But the unbounded programmability of the ROM helps us overcome the aforementioned impasse in the proof. Now, the challenger answers partial signature queries by returning (noisy) random secret shares $\{\sigma_{i,j}\}_{i\in[N],j\in\mathsf{poly}}$ of σ_j. When later, user 1 is corrupted, it can correctly explain the returned signatures as follows: it samples sk_1, computes $d_{1,j} = \langle \widehat{\sigma}_j, \mathsf{sk}_1 \rangle + \mathsf{noise}$ and sets $r_{1,j} = \sigma_{1,j} - d_{1,j}$. Now we may program $H(j,K)$ so that $r_{i,j} = H(j,K)^{\mathsf{T}}\ \mathbf{v}_i$ for all j – it can be checked that there are enough degrees of freedom to satisfy these equations. However, since all secrets of a user are revealed when it is corrupted, the value $H(j,K)$ is fixed when even a single user is corrupted. This is why we require that all corruptions be made simultaneously and only achieve the restricted notion of "partial" adaptivity.

We also provide a construction in the standard model which achieves full adaptivity where users can be corrupted at arbitrary points in the security game. But this construction is only secure in a weaker *pre-processing* model where the signers must be provided correlated randomness of length proportional to the number of signing queries, in an offline pre-processing phase. We emphasize that the correlated randomness is independent of the messages to be signed later. This model is reminiscent of the "MPC with Preprocessing" model (please see [FKOS15] and references therein). We refer the reader to Section 6.7 and 6.8 for more details.

Oraganisation of the chapter. We organize the rest of the chapter as follows. In 6.4, we give a formal definition of the threshold signatures and define other preliminaries and notations used in this chapter. In section 6.5, we show how to reduce noise flooding in BGGJKRS construction. In section 6.6, we describe our rejection free version of Lyubashevsky's signature scheme. We give our constructions of partially adaptive and

fully adaptive schemes in sections 6.7 and 6.8, respectively. In section 6.9, we give the construction for t-out-of-N access structure.

6.4 PRELIMINARIES

We use the prliminaries defined in Chapter 2. In addition, we define the notations and other preliminaries used in this chapter.

Notations used in this chapter. In this chapter, a vector $\mathbf{v}$ is by default, a column vector. Let S be any set, then $|S|$ represents the cardinality of S, while in case of any $x \in \mathbb{R}$, $|x|$ represents absolute value of x. For lattices, we use the definitions and lemmas from Chapter 2. $\mathcal{D}_{\Lambda,s,\mathbf{c}}$ represents discrete Gaussian distribution over lattice Λ, with center $\mathbf{c}$ and standard deviation parameter s. When $\mathbf{c} = 0$, we omit it. Similarly, we omit Λ, if $\Lambda = \mathbb{Z}$.

6.4.1 Threshold Signatures

Definition 6.1 (Threshold Signatures)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of N parties. A threshold signature scheme for a class of efficient access structures $\mathbb{S}$ on P (see Def. 6.23) is a tuple of PPT algorithms denoted by $\mathsf{TS} = (\mathsf{TS.KeyGen}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ defined as follows:

- $\mathsf{TS.KeyGen}(1^\lambda, \mathsf{A}) \to (\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N)$: On input the security parameter λ and an access structure A, the KeyGen algorithm outputs public parameters pp, verification key vk and a set of key shares $\{\mathsf{sk}_i\}_{i=1}^N$.

- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m) \to \sigma_i$: On input the public parameters pp, a partial signing key sk_i and a message $m \in \{0,1\}^*$ to be signed, the partial signing algorithm outputs a partial signature σ_i.

- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i) \to \mathsf{accept/reject}$: On input the public parameters pp, a message $m \in \{0,1\}^*$ and a partial signature σ_i, the partial signature verification algorithm outputs accept or reject.

- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i \in S}) \to \sigma_m$: On input the public parameters pp and the

partial signatures $\{\sigma_i\}_{i\in S}$ for $S \in \mathsf{A}$, the combining algorithm outputs a full signature σ_m.

- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) \rightarrow \mathsf{accept/reject}$: On input a verification key vk, a message m and a signature σ_m, the verification algorithm outputs accept or reject.

A TS scheme should satisfy the following requirements.

Definition 6.2 (Compactness)**.** A TS scheme for $\mathbb{S}$ satisfies compactness if there exist polynomials $\text{poly}_1(\cdot), \text{poly}_2(\cdot)$ such that for all λ, $\mathsf{A} \in \mathbb{S}$ and $S \in \mathsf{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in S$, and $\sigma_m \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$, we have that $|\sigma_m| \leq \text{poly}_1(\lambda)$ and $|\mathsf{vk}| \leq \text{poly}_2(\lambda)$.

Definition 6.3 (Evaluation Correctness)**.** A signature scheme TS for $\mathbb{S}$ satisfies evaluation correctness if for all $\lambda, \mathsf{A} \in \mathbb{S}$ and $S \in \mathsf{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in [N]$ and $\sigma_m \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$, we have that $\Pr[\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) = accept] \geq 1 - \lambda^{-\omega(1)}$.

Definition 6.4 (Partial Verification Correctness)**.** A signature scheme TS for $\mathbb{S}$ satisfies partial verification correctness if for all λ and $\mathsf{A} \in \mathbb{S}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$,

$$\Pr[\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)) = 1] = 1 - \lambda^{-\omega(1)}.$$

Definition 6.5 (Unforgeability)**.** A TS scheme is unforgeable if for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the output of the following experiment $\mathsf{Expt}^{\mathsf{uf}}_{\mathsf{TS},\mathcal{A}}(1^\lambda)$ is 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ, the adversary outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. Challenger runs the $\mathsf{TS.KeyGen}(1^\lambda)$ algorithm and generates public parameters pp, verification key vk and set of N key shares $\{\mathsf{sk}_i\}_{i=1}^N$. It sends pp and vk to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ then issues polynomial number of following two types of queries in any order
   - Corruption queries: $\mathcal{A}$ outputs a party $i \in [N]$ which it wants to corrupt. In response, the challenger returns the key share sk_i.

- Signature queries: $\mathcal{A}$ outputs a query of the form (m, i), where m is a message and $i \in [N]$, to get partial signature σ_i for m. The challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.

4. At the end of the experiment, adversary $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).

5. The experiment outputs 1 if both of the following conditions are met: (i) Let $S \subseteq [N]$ be the set of corrupted parties, then S is a an invalid party set, i.e. $S \notin \mathsf{A}$ (ii) m^* was not queried previously as a signing query and $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = accept$.

We also consider following weaker notions of unforgeability.

Definition 6.6 (Partially Adaptive Unforgeability)**.** Here, all the corruptions are done all at once. That is, Step 3, is now changed as follows:

- $\mathcal{A}$ issues polynomial number of signing queries of the form (m, i) adaptively and gets corresponding σ_i's.

- $\mathcal{A}$ outputs a set $S \subseteq [N]$ such that $S \notin \mathsf{A}$. The challenger returns $\{\mathsf{sk}_i\}_{i \in S}$.

- $\mathcal{A}$ continues to issue polynomial number of more signing queries of the form (m, i) adaptively, and gets corresponding σ_i.

Rest of the steps remain the same.

Definition 6.7 (Selective Unforgeability)**.** In this case, all the corruptions happen before any signing query. That is, Step 3, is now further changed as follows:

- $\mathcal{A}$ outputs a set $S \subseteq [N]$ such that $S \notin \mathsf{A}$. The challenger returns $\{\mathsf{sk}_i\}_{i \in S}$.

- $\mathcal{A}$ issues polynomial number of signing queries of the form (m, i) adaptively, and gets corresponding σ_i.

Rest of the steps remain the same.

Definition 6.8 (Robustness)**.** A TS scheme for $\mathbb{S}$ satisfies robustness if for all λ, the following holds. For any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment $\mathsf{Expt}^{\mathsf{rb}}_{\mathsf{TS},\mathcal{A}}(1^\lambda)$ outputs 1 with probability $2^{-\Omega(\lambda)}$:

- On input the security parameter 1^λ, the adversary outputs an access structure $\mathsf{A} \in \mathbb{S}$.

- The challenger samples $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$ and provides $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ to $\mathcal{A}$.

- Adversary $\mathcal{A}$ outputs a partial signature forgery (m^*, σ_i^*, i).
- The experiment outputs 1 if $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m^*, \sigma_i^*) = 1$ and $\sigma_i^* \neq \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m^*)$.

6.4.2 Fully Homomorphic Encryption (FHE)

A fully homomorphic encryption scheme is an encryption scheme that allows computations on encrypted data.

Definition 6.9 (Fully Homomorphic Encryption)**.** A fully homomorphic encryption scheme FHE is a tuple of PPT algorithms $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ defined as follows:

- $\mathsf{FHE.KeyGen}(1^\lambda, 1^d) \rightarrow (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ and a depth bound d, the KeyGen algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{FHE.Enc}(\mathsf{pk}, \mu) \rightarrow \mathsf{ct}$: On input a public key pk and a message $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) \rightarrow \hat{\mathsf{ct}}$: On input a public key pk, a circuit $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, and a tuple of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{FHE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) \rightarrow \hat{\mu}$: On input a public key pk, a secret key sk and a ciphertext $\hat{\mathsf{ct}}$, the decryption algorithm outputs a message $\hat{\mu} \in \{0, 1, \perp\}$.

The definition above can be adapted to handle plaintexts over larger sets than $\{0,1\}$. Note that the evaluation algorithm takes as input a (deterministic) circuit rather than a possibly randomized algorithm. An FHE should satisfy compactness, correctness and security properties defined below.

Definition 6.10 (Compactness)**.** We say that an FHE scheme is compact if there exists a polynomial function $f(\cdot, \cdot)$ such that for all λ, depth bound d, circuit $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the bit-length of $\hat{ct}$ is at most $f(\lambda, d)$.

Definition 6.11 (Correctness)**.** We say that an FHE scheme is correct if for all λ, depth bound d, circuit $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the

following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we have

$$\Pr[\mathsf{FHE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) = C(\mu_1, \ldots, \mu_k)] = 1 - \lambda^{-\omega(1)}.$$

Definition 6.12 (Security)**.** We say that an FHE scheme is secure if for all λ and depth bound d, the following holds: for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment outputs 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ and a depth bound d, the challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$ and $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, b)$ for $b \leftarrow \{0, 1\}$. It provides $(\mathsf{pk}, \mathsf{ct})$ to $\mathcal{A}$.

2. $\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b = b'$.

In this work, our constructions use a *special* FHE having some additional properties as described in [BGG+18]. These properties are satisfied by direct adaptations of typical FHE schemes such as [BV11; GSW13] (see, e.g., [BGG+18, Appendix B]).

Definition 6.13 (Special FHE)**.** An FHE scheme is a special FHE scheme if it satisfies the following properties:

1. On input $(1^\lambda, 1^d)$, the key generation algorithm $\mathsf{FHE.KeyGen}$ outputs $(\mathsf{pk}, \mathsf{sk})$, where the public key contains a prime q and the secret key is a vector $\mathsf{sk} \in \mathbb{Z}_q^m$ for some $m = \mathrm{poly}(\lambda, d)$.

2. The decryption algorithm $\mathsf{FHE.Dec}$ consists of two functions $(\mathsf{FHE.decode}_0, \mathsf{FHE.decode}_1)$ defined as follows:

   - $\mathsf{FHE.decode}_0(\mathsf{sk}, \mathsf{ct})$: On input an encryption of a message $\mu \in \{0, 1\}$ and a secret key vector sk, it outputs $p = \mu \lfloor q/2 \rceil + e \in \mathbb{Z}_q$ for $e \in [-cB, cB]$ with $B = B(\lambda, d, q)$ and e is an integer multiple of c. This algorithm must be a linear operation over $\mathbb{Z}_q$ in the secret key sk.

   - $\mathsf{FHE.decode}_1(p)$: On input $p \in \mathbb{Z}_q$, it outputs 1 if $p \in [-\lfloor q/4 \rceil, \lfloor q/4 \rceil]$, and 0 otherwise.

   The bound $B = B(\lambda, d, q)$ is referred to as the associated noise bound parameter of the construction and c as the associated multiplicative constant.

6.4.3 Threshold Fully Homomorphic Encryption

Definition 6.14 (Threshold Fully Homomorphic Encryption)**.** A threshold fully homomorphic encryption for a class of efficient access structures $\mathbb{S}$, defined on a set $P = \{P_1, P_2, \ldots, P_N\}$ of parties is defined by a tuple of five algorithms $\mathsf{TFHE} = (\mathsf{TFHE.KeyGen}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ with the following specifications:

- $\mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A}) \rightarrow (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$: On input the security parameter λ, a depth bound d and an access structure $\mathsf{A} \in \mathbb{S}$, the KeyGen algorithm outputs a public key pk and a set of secret key shares $\{\mathsf{sk}_i\}_{i=1}^N$.

- $\mathsf{TFHE.Enc}(\mathsf{pk}, \mu) \rightarrow \mathsf{ct}$: On input a public key pk and a single bit message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_k) \rightarrow \hat{\mathsf{ct}}$: On input a public key pk, a circuit $C : \{0, 1\}^k \rightarrow \{0, 1\}$ of depth at most d and a set of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.

- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}) \rightarrow p_i$: On input a public key pk, a secret key share sk_i and a ciphertext ct, the partial decryption algorithm outputs a partial decryption p_i corresponding to the party P_i.

- $\mathsf{TFHE.FinDec}(\mathsf{pk}, \{p_i\}_{i \in S}) \rightarrow \hat{\mu}$: On input a public key pk and a set of partial decryptions corresponding to parties in some set $S \subseteq [N]$, the final decryption algorithm outputs a message $\hat{\mu} \in \{0, 1, \perp\}$.

Definition 6.15 (Correctness)**.** A TFHE scheme for $\mathbb{S}$ is said to satisfy evaluation correctness if for all λ, depth bound d, access structure $\mathsf{A} \in \mathbb{S}$, circuit $C : \{0, 1\}^k \rightarrow \{0, 1\}$ of depth at most d, $S \in \mathsf{A}$, and $\mu_i \in \{0, 1\}$ for $i \in [k]$, the following condition holds. For $(\mathsf{pp}, sk_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$:

$$\Pr[\mathsf{TFHE.FinDec}(\mathsf{pk}, \{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i \in S}) = C(\{\mu_i\}_{i \in [k]})] = 1 - \lambda^{-\omega(1)}.$$

Definition 6.16 (Semantic security)**.** A TFHE scheme is said to satisfy semantic security if for all λ and depth bound d, the following holds. For any adversary $\mathcal{A}$ with run-time bounded as $2^{o(\lambda)}$, the experiment below outputs 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ, and a circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.

2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a set S of participants, such that $S \notin \mathsf{A}$.

4. The challenger provides $\{\mathsf{sk}_i\}_{i \in S}$ and $\mathsf{TFHE.Enc}(\mathsf{pk}, b)$, where $b \leftarrow \{0,1\}$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs a guess bit b'. The experiment outputs 1 if $b = b'$.

Definition 6.17 (Simulation security)**.** A TFHE scheme for $\mathbb{S}$ is said to satisfy simulation security if for all λ, depth bound d and access structure A, the following holds: there exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any adversary $\mathcal{A}$ with run-time bounded as $2^{o(\lambda)}$, the following two experiments are indistinguishable.

$\mathsf{Expt}^{\mathsf{Real}}_{\mathsf{TFHE},\mathcal{A}}(1^\lambda, 1^d)$:

1. On input the security parameter λ and a circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.

2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.

3. Adversary $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq [N]$ and a set of message bits, $\mu_1, \mu_2, \ldots, \mu_k \in \{0,1\}$.

4. The challenger provides $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i = \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.

5. Adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_N\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d.

6. For each query, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and sends $\{\mathsf{TFHE.PartDec}\ (\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i \in S}$ to $\mathcal{A}$.

7. At the end of the experiment, adversary $\mathcal{A}$ outputs a bit b.

$\mathsf{Expt}^{\mathsf{Ideal}}_{\mathsf{TFHE},\mathcal{A}}(1^\lambda, 1^d)$:

1. On input the security parameter λ and circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.

2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.

3. Adversary $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq P$ and a set of message bits, $\mu_1, \mu_2, \ldots, \mu_k \in \{0,1\}$.

4. The challenger provides $\{\mathsf{sk}_i\}_{i\in S^*}$ and $\{\mathsf{ct}_i = \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)\}_{i\in[k]}$ to $\mathcal{A}$.

5. Adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq [N], C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d.

6. For each query, the challenger runs the simulator $\mathcal{S}_2$ to compute partial decryptions as
$\{\mathsf{p}_i\}_{i\in S} \leftarrow \mathcal{S}_2(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k, C(\mu_1, \ldots, \mu_k), S, \mathsf{st})$ and sends $\{\mathsf{p}_i\}_{i\in S}$ to $\mathcal{A}$.

7. At the end of the experiment, adversary $\mathcal{A}$ outputs a bit b.

6.4.4 Multi-data Homomorphic Signature

A homomorphic signature scheme is a signature scheme that allows computations on authenticated data. In a multi-data homomorphic signature scheme, the signer can sign many different datasets of arbitrary size. Each dataset is tied to some label τ (e.g., the name of the dataset) and the verifier is assumed to know the label of the dataset over which it wishes to verify computations.

Definition 6.18 (Multi-data Homomorphic Signature)**.** A *multi-data homomorphic signature* for messages over a set $\mathcal{X}$ is a tuple of PPT algorithms $\mathsf{HS} = (\mathsf{HS.PrmsGen}, \mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Process}, \mathsf{HS.Verify})$ with the following syntax.

- $\mathsf{HS.PrmsGen}(1^\lambda, 1^N) \to \mathsf{prms}$: Gets the security parameter λ and a data-size bound N and generates public parameters prms.

- $\mathsf{HS.KeyGen}(1^\lambda, \mathsf{prms}) \to (\mathsf{pk}, \mathsf{sk})$: Produces a public verification key pk and a secret signing key sk.

- $\mathsf{HS.Sign}(\mathsf{sk}, (x_1, \ldots, x_N), \tau) \to (\sigma_\tau, \sigma_1, \ldots, \sigma_N)$: Signs some data $(x_1, \ldots, x_N) \in \mathcal{X}^*$ under a label $\tau \in \{0,1\}^*$.

- $\mathsf{HS.SignEval}(\mathsf{prms}, g, \sigma_\tau, (x_1, \sigma_1), (x_\ell, \sigma_\ell)) \to \sigma^*$: Homomorphically computes a signature σ^* for $g(x_1, \ldots, x_N)$.

- $\mathsf{HS.Process}(\mathsf{prms}, g) \to \alpha_g$: Produces a "public-key" α_g for the function g.

- $\mathsf{HS.Verify}(\mathsf{pk}, \alpha_g, y, \tau, (\sigma_\tau, \sigma^*)) \to \mathsf{accept/reject}$: Verifies that $y \in \mathcal{X}$ is indeed

the output of the function g over the data signed with label τ. We can define the "combined verification procedure" $\mathsf{HS.Verify}^*(\mathsf{pk}, g, y, \tau, \sigma_\tau, \sigma^*)$ as: Compute $\alpha_g \leftarrow \mathsf{HS.Process}(\mathsf{prms}, g)$ and output $\mathsf{HS.Verify}(\mathsf{pk}, \alpha_g, y, \tau, (\sigma_\tau, \sigma^*))$.

A homomorphic signature should satisfy the correctness and security properties defined below.

Definition 6.19 (Correctness)**.** *Correctness of Signing.* Let $\mathsf{id}_i : \mathcal{X}^N \to \mathcal{X}$ be a canonical description of the function $\mathsf{id}_i(x_1, ..., x_N) = x_i$ (i.e., a circuit consisting of a single wire taking the i'th input to the output). We require that for any $\mathsf{prms} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^N)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{prms})$, $(x_1, \ldots, x_N) \in \mathcal{X}^N$, any $\tau \in \{0,1\}^*$, and any $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, (x_1, \ldots, x_N), \tau)$, the following must satisfy:

$\mathsf{HS.Verify}^*(\mathsf{pk}, \mathsf{id}_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \mathsf{accept}$. In other words, the pair (σ_τ, σ_i) certifies x_i as the ith data item of the data with label τ.

Correctness of Evaluation. For any functions $h_1, \ldots, h_\ell$ with $h_i : \mathcal{X}^N \to \mathcal{X}$ for $i \in [\ell]$, any function $g : \mathcal{X}^\ell \to \mathcal{X}$, any $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$, any $\tau \in \{0,1\}^*$ and any $(\sigma_\tau, \sigma_1, \ldots, \sigma_\ell)$:

$$\begin{aligned} &\{\{\mathsf{HS.Verify}(\mathsf{pk}, h_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \mathsf{accept}\}_{i \in [\ell]}, \\ &\sigma^* \leftarrow \mathsf{HS.SignEval}(\mathsf{prms}, g, \sigma_\tau, (x_1, \sigma_1), (x_\ell, \sigma_\ell))\} \\ \Rightarrow\ &\mathsf{HS.Verify}^*(\mathsf{pk}, (g \circ \bar{h}), g(x_1, \ldots, x_\ell), \tau, (\sigma_\tau, \sigma^*)) = \mathsf{accept}. \end{aligned}$$

In other words, if the signatures (σ_τ, σ_i) certify x_i as the outputs of function h_i over the data labeled with τ for all $i \in [\ell]$, then (σ_τ, σ^*) certifies $g(x_1, \ldots, x_\ell)$ as the output of $g \circ \bar{h}$ over the data labeled with τ.

Definition 6.20 (Security)**.** The security is defined via the following game between an attacker $\mathcal{A}$ and a challenger:

- The challenger runs $\mathsf{prms} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^N)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HS.KeyGen}(\mathsf{prms}, 1^\lambda)$, and gives $\mathsf{prms}, \mathsf{pk}$ to the attacker $\mathcal{A}$.
- Signing queries: The attacker $\mathcal{A}$ can ask an arbitrary number of signing queries. In each query j, the attacker chooses a fresh tag τ_j which was never queried

previously and a message $(x_{j1}, \ldots, x_{jN_j}) \in \mathcal{X}^*$. The challenger responds with

$$(\sigma_{\tau_j}, \sigma_{j1}, \ldots, \sigma_{jN_j}) \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, (x_{j1}, \ldots, x_{jN_j}), \tau_j).$$

- The attacker $\mathcal{A}$ outputs a function $g : \mathcal{X}^{N'} \rightarrow \mathcal{X}$ and values $\tau, y', (\sigma'_\tau, \sigma')$. The attacker wins if $.\mathsf{HS.Verify}^*(\mathsf{pk}, g, y', \tau, (\sigma'_\tau, \sigma')) = \mathsf{accept}$ and either:

  - *Type 1 forgery*: $\tau \notin \{\tau_j\}_j$ or $\tau = \tau_j$ for some j but $N' \neq N_j$, i.e., the signing query with label τ was never made or there is a mismatch between the size of the data signed under label τ and the arity of the function g.

  - *Type 2 forgery*: $\tau = \tau_j$ for some j with corresponding message $x_{j,1}, \ldots, x_{j,N'}$ such that (a) g is admissible on $x_{j,1}, \ldots, x_{j,N'}$ and (b) $y' \neq g(x_{j,1}, \ldots, x_{j,N'})$.

We require that for all $\mathcal{A}$ with run-time $2^{o(\lambda)}$, we have $\Pr[\mathcal{A} \text{ wins}] \leq 2^{-\Omega(\lambda)}$ in the above game.

We now give a simulation-based notion of context-hiding security, requiring that a context hiding signature $\tilde{\sigma}$ can be simulated given the knowledge of only the computation g and output y, but without any other knowledge of underlying data. The simulation remains indistinguishable even given the underlying data, the underlying signatures, and even the public/secret key of the scheme. In other words, the derived signature does not reveal anything beyond the output of the computation even to an attacker that may have some partial information on the underlying values.

Definition 6.21 (Context Hiding)**.** A multi-data homomorphic signature supports context hiding if there exist additional PPT procedures $\tilde{\sigma} \leftarrow \mathsf{HS.Hide}(\mathsf{pk}, y, \sigma)$, $\mathsf{HS.HVerify}(\mathsf{pk}, g, \mathsf{HS.Process}(g), y, \tau, (\sigma_\tau, \tilde{\sigma}))$ such that:

- *Correctness:* For any $\mathsf{prms} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^N)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{prms})$ and any $\alpha, y, \tau, \sigma_\tau, \sigma$ such that $\mathsf{HS.Verify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathsf{accept}$, for any $\tilde{\sigma} \leftarrow \mathsf{HS.Hide}(\mathsf{pk}, y, \sigma)$ we have

$$\mathsf{HS.HVerify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \tilde{\sigma})) = \mathsf{accept}.$$

- *Unforgeability:* Multi-data signature security holds when we replace the $\mathsf{HS.Verify}$ procedure by $\mathsf{HS.HVerify}$ in the security game.

- *Context hiding security:* Firstly, in the procedure $(\sigma_\tau, \{\sigma_i\}_{i \in [N]}) \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, \{x_i\}_{i \in [N]}, \tau)$, we require that σ_τ can only depend on (sk, N, τ) but not on the

data $\{x_i\}$. Secondly, we require that there is a simulator $\mathsf{HS.Sim}$ such that for any fixed (worst-case) choice of prms, $(\mathsf{pk}, \mathsf{sk})$ and any $\alpha, y, \tau, \sigma_\tau, \sigma$ such that $\mathsf{HS.Verify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathsf{accept}$, we have that the distributions $\mathsf{HS.Hide}(\mathsf{pk}, y, \sigma)$ and $\mathsf{HS.Sim}(\mathsf{sk}, \alpha, y, \tau, \sigma_\tau)$ are indistinguishable, where the randomness is only over the random coins of the simulator and the $\mathsf{HS.Hide}$ procedure. We say that such schemes are statistically context hiding if the above indistinguishability holds statistically.

6.4.5 Rényi Divergence

The Rényi Divergence (RD) is a measure of closeness of any two probability distributions. In certain cases, especially in proving the security of cryptographic primitives where the adversary is required to solve a search-based problem, the RD can be used as an alternative to the statistical distance [BLRL+18], which may help obtain security proofs for smaller scheme parameters and may sometimes lead to simpler proofs.

Definition 6.22 (Rényi Divergence)**.** Let P and Q be any two discrete probability distributions such that $Supp(P) \subseteq Supp(Q)$. Then for $a \in (1, \infty)$, the Rényi Divergence of order a is defined by

$$R_a(P||Q) = \left(\sum_{x \in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

For $a = 1$ and $a = \infty$, the RD is defined as

$$R_1(P||Q) = \exp\left(\sum_{x \in Supp(P)} P(x) \log \frac{P(x)}{Q(x)} \right) \text{ and } R_\infty(P||Q) = \max_{x \in Supp(P)} \frac{P(x)}{Q(x)}.$$

For any fixed distributions P and Q, the function $f(a) = R_a(P\|Q)$ is non decreasing, continuous over $(1, \infty)$ and tends to $R_\infty(P\|Q)$ as a goes to infinity. Further, if $R_a(P\|Q)$ is finite for some a, then it tends to $R_1(P\|Q)$ as a tends to 1.

The following lemma is borrowed from [BLRL+18, Lemma 2.9], with the exception of the multiplicativity property for non-independent variables, which is borrowed from [Ros20, Proposition 2].

Lemma 6.1. *Let $a \in [1, \infty]$. Let P and Q denote distributions with $Supp(P) \subseteq$*

$Supp(Q)$. Then the following properties hold

- ***Log Positivity:*** $R_a(P||Q) \geq R_a(P||P) = 1$.
- ***Data Processing Inequality:*** $R_a(P^f||Q^f) \leq R_a(P||Q)$ *for any function f, where P^f (resp. Q^f) denotes the distribution of $f(y)$ induced by sampling $y \leftarrow P$ (resp. $y \leftarrow Q$).*
- ***Probability preservation:*** *Let $E \subseteq Supp(Q)$ be an arbitrary event. If $a \in (1, \infty)$, then*

$$Q(E) \geq P(E)^{\frac{a}{a-1}} / R_a(P\|Q).$$

For $a = \infty$,

$$Q(E) \geq P(E) / R_\infty(P\|Q).$$

For $a = 1$, Pinsker's inequality gives the following analogue property:

$$Q(E) \geq P(E) - \sqrt{\ln R_1(P||Q)/2}.$$

- ***Multiplicativity:*** *Assume that P and Q are two distributions of a pair of random variables (Y_1, Y_2). For $i \in \{1, 2\}$, let P_i (resp. Q_i) denote the marginal distribution of Y_i under P (resp. Q), and let $P_{2|1}(\cdot|y_1)$ (resp. $Q_{2|1}(\cdot|y_1)$) denote the conditional distribution of Y_2 given that $Y_1 = y_1$. Then we have:*
  - $R_a(P||Q) = R_a(P_1||Q_1) \cdot R_a(P_2||Q_2)$ *if Y_1 and Y_2 are independent for $a \in [1, \infty]$.*
  - $R_a(P||Q) \leq R_a(P_1||Q_1) \cdot \max_{y_1 \in Y_1} R_a(P_{2|1}(\cdot|y_1)||Q_{2|1}(\cdot|y_1))$.
- ***Weak Triangle Inequality:*** *Let P_1, P_2, P_3 be three distributions with $Supp(P_1) \subseteq Supp(P_2) \subseteq Supp(P_3)$. Then we have*

$$R_a(P_1||P_3) \leq \begin{cases} R_a(P_1||P_2) \cdot R_\infty(P_2||P_3), \\ R_\infty(P_1||P_2)^{\frac{a}{a-1}} \cdot R_a(P_2||P_3) \quad \text{if } a \in (1, +\infty). \end{cases} \tag{6.1}$$

We will use the following RD bounds. Note that proof tightness can often be improved by optimizing over a, as suggested in [TT15].

Lemma 6.2 ([BLRL$^+$18])**.** *For any n-dimensional lattice, $\Lambda \subseteq \mathbb{R}^n$ and $s > 0$, let P be the distribution $\mathcal{D}_{\Lambda,s,\mathbf{c}}$ and Q be the distribution $\mathcal{D}_{\Lambda,s,\mathbf{c}'}$ for some fixed $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^n$. If $\mathbf{c}, \mathbf{c}' \in \Lambda$, let $\epsilon = 0$. Otherwise fix $\epsilon \in (0, 1)$ and assume that $s > \eta_\epsilon(\Lambda)$. Then for any*

$a \in (1, +\infty)$

$$R_a(P||Q) \in \left[\left(\frac{1-\epsilon}{1+\epsilon}\right)^{\frac{2}{a-1}}, \left(\frac{1+\epsilon}{1-\epsilon}\right)^{\frac{2}{a-1}}\right] \cdot \exp\left(a\pi\frac{||\mathbf{c}-\mathbf{c}'||^2}{s^2}\right).$$

6.4.6 Secret Sharing

We now recall some standard definitions related to secret sharing.

Definition 6.23 (Monotone Access Structure)**.** Let $P = \{P_i\}_{i\in[N]}$ be a set of parties. A collection $\mathsf{A} \subseteq \mathcal{P}(P)$ is monotone if for any two sets $B, C \subseteq P$, if $B \in \mathsf{A}$ and $B \subseteq C$, then $C \in \mathsf{A}$. A monotone access structure on P is a monotone collection $\mathsf{A} \subseteq \mathcal{P}(P) \setminus \emptyset$. The sets in A are called *valid sets* and the sets in $\mathcal{P}(P) \setminus \mathsf{A}$ are called *invalid sets*.

Let $S \subseteq P$ be a subset of parties in P. S is called maximal invalid party set if $S \notin \mathsf{A}$, but for any $P_i \in P \setminus S$, we have $S \cup \{P_i\} \in \mathsf{A}$. S is called minimal valid party set if $S \in \mathsf{A}$, but for any $S' \subsetneq S$, we have $S' \notin \mathsf{A}$.

In this work, since we only use monotone access structures, we sometimes drop the word monotone. When it is clear from the context, we use either i or P_i to represent party P_i.

Definition 6.24 (Threshold Access Structure)**.** Let $P = \{P_i\}_{i\in[N]}$ be a set of N parties. An access structure A_t is called a threshold access structure, if for all $S \subseteq P$, we have $S \in \mathsf{A}$ iff $|S| \geq t$. We let TAS denote the class of all access structures A_t for all $t \in \mathbb{N}$.

For any set of parties $S \subseteq P$, we define $\mathbf{x}_S = (x_1, \ldots, x_N) \in \{0,1\}^N$ with $x_i = 1$ iff $P_i \in S$.

Definition 6.25 (Efficient Access Structure)**.** An access structure A on set P as defined above is called an efficient access structure if there exists a polynomial size circuit $f_{\mathsf{A}} : \{0,1\}^N \to \{0,1\}$, such that for all $S \subseteq P$, $f_{\mathsf{A}}(\mathbf{x}_S) = 1$ iff $S \in \mathsf{A}$.

Definition 6.26 (Secret sharing)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures on P. A secret sharing scheme SS for a secret space $\mathcal{K}$ is a tuple of PPT algorithms $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ defined as follows:

- $\mathsf{SS.Share}(\mathsf{k}, \mathsf{A}) \to (s_1, \ldots, s_N)$: On input a secret $\mathsf{k} \in \mathcal{K}$ and an access structure A, the sharing algorithm returns shares $s_1, \ldots, s_N$ for all parties.
- $\mathsf{SS.Combine}(B) \to \mathsf{k}$: On input a set of shares $B = \{s_i\}_{i\in S}$, where $S \subseteq [N]$, the combining algorithm outputs a secret $\mathsf{k} \in \mathcal{K}$.

A secret sharing algorithm must satisfy the following correctness and privacy properties.

Definition 6.27 (Correctness)**.** For all $S \in \mathsf{A}$ and $\mathsf{k} \in \mathcal{K}$, if $(s_1, \ldots, s_N) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathsf{A})$, then

$$\mathsf{SS.Combine}(\{s_i\}_{i \in S}) = \mathsf{k}.$$

Definition 6.28 (Privacy)**.** For all $S \notin \mathsf{A}$ and $\mathsf{k}_0, \mathsf{k}_1 \in \mathcal{K}$, if $(s_{b,1}, \ldots, s_{b,N}) \leftarrow \mathsf{SS.Share}(\mathsf{k}_b, \mathsf{A})$ for $b \in \{0, 1\}$, then the distributions $\{s_{0,i}\}_{i \in S}$ and $\{s_{1,i}\}_{i \in S}$ are identical.

Definition 6.29 (Linear Secret Sharing (LSSS))**.** Let $P = \{P_i\}_{i \in [N]}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures. A secret sharing scheme SS with secret space $\mathcal{K} = \mathbb{Z}_p$ for some prime p is called a linear secret sharing scheme if it satisfies the following properties:

- $\mathsf{SS.Share}(\mathsf{k}, \mathsf{A})$: There exists a matrix $\mathbf{M} \in \mathbb{Z}_p^{\ell \times N}$ called the *share matrix*, and each party P_i is associated with a partition $T_i \subseteq [\ell]$. To create the shares on a secret k, the sharing algorithm first samples uniform values $r_2, \ldots, r_N \leftarrow \mathbb{Z}_p$ and defines a vector $\mathbf{w} = \mathbf{M} \cdot (\mathsf{k}, r_2, \ldots, r_N)^T$. The share for P_i consists of the entries $\{w_j\}_{j \in T_i}$.

- $\mathsf{SS.Combine}(B)$: For any valid set $S \in \mathsf{A}$, we have

  $$(1, 0, \ldots, 0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in \bigcup_{i \in S} T_i}).$$

  over $\mathbb{Z}_p$ where $M[j]$ denotes the jth row of M. Any valid set of parties $S \in \mathsf{A}$ can efficiently find the coefficients $\{c_j\}_{j \in \bigcup_{i \in S} T_i}$ satisfying

  $$\sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot \mathbf{M}[j] = (1, 0, \ldots, 0)$$

  and recover the secret by computing $\mathsf{k} = \sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot w_j$. The coefficients $\{c_j\}$ are called *recovery coefficients*.

Definition 6.30. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, $\mathbb{S}$ a class of efficient structures on P, and SS a linear secret sharing scheme with share matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times N}$. For a set of indices $T \subseteq [\ell]$, T is said to be a valid share set if $(1, 0, \ldots, 0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T})$, and an invalid share set otherwise. We also use following definitions:

- A set of indices $T \subseteq [\ell]$ is a maximal invalid share set if T is an invalid share set,

but for any $i \in [\ell] \setminus T$, the set $T \cup \{i\}$ is a valid share set.

- A set of indices $T \subseteq [\ell]$ is a minimal valid share set if T is a valid share set, but for any $T' \subsetneq T$, T' is an invalid share set.

The class of access structures that can be supported by a linear secret sharing scheme on N parties is represented by LSSS_N. When the context is clear LSSS_N is simply written as LSSS. We let $\{\mathbf{0}, \mathbf{1}\}$-$\mathsf{LSSS}$ denote the class of access structures that can be supported by a LSSS where the recovery coefficients are binary: for a set P of N parties, let k be the shared secret and $\{w_j\}_{j \in T_i}$ be the share of party P_i for $i \in [N]$; then for every set $S \in \mathsf{A}$, there exists a subset $T \subseteq \bigcup_{i \in S} T_i$ such that $\mathsf{k} = \sum_{j \in T} w_j$. It was shown in [BGG+18] that such a set $T \subseteq \bigcup_{i \in S} T_i$ can be computed efficiently, and that TAS belongs to $\{0, 1\}$-LSSS. Hence, we can use $\{0, 1\}$-LSSS for secret sharing in TAS.

To secret-share a vector $\mathbf{s} = \{s_1, \ldots, s_n\} \in \mathbb{Z}_p^n$, we can simply secret-share each entry s_i using fresh randomness. This gives secret share vectors $\mathbf{s}_1, \ldots, \mathbf{s}_\ell \in \mathbb{Z}_p^n$. Using these secret shares, the secret vector $\mathbf{s}$ can be recovered using the same coefficients as that for a single field element.

6.4.7 Lattice preliminaries

For definitions and related hard problems on lattices, we refer to Chapter 2. We will use the following adaptation of Lemma 2.1.

Lemma 6.3 (Adapted from [Lyu12, Lemma 4.4])**.**

1. *For any* $k > 0$, $\Pr[|z| > k\sigma; z \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}] \leq 2\exp(-\pi k^2)$.
2. *For any* $\sigma \geq 3$, $H_\infty(\mathcal{D}_{\mathbb{Z}^m,\sigma}) \geq m$.
3. *For any* $k > 1/\sqrt{2\pi}$, $\Pr[\|\mathbf{z}\| > k\sigma\sqrt{2\pi m};\ \mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}] < (k\sqrt{2\pi})^m \exp(\frac{m}{2}(1 - 2\pi k^2))$.

6.5 MORE EFFICIENT THRESHOLD SIGNATURES FROM LATTICES

In this section, we show how to drastically decrease the exponential flooding used in the scheme by Boneh *et al* [BGG+18]. We also show that the limited flooding that we

use is in fact optimal, and smaller noise would lead to an attack. For ease of exposition, the construction below is for the special case of N out of N threshold and restricted to selective security. We extend it to adaptive security in Section 6.7 and Section 6.8 and t out of N threshold in Section 6.9. In Section 6.6, we show how to instantiate the underlying signature scheme using a variant of Lyubashevsky's signature [Lyu12] with matching moderate flooding.

6.5.1 Optimizing the Boneh *et al* scheme using the Rényi Divergence

Our scheme is similar to the one in [BGG+18]. The construction uses the following building blocks:

- A PRF $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^r$, where $\mathcal{K}$ is the PRF key space and r is the bit-length of randomness used in sampling from discrete Gaussian $\mathcal{D}_s$.
- A fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$. As in [BGG+18], we also assume that the $\mathsf{FHE.Dec}$ can be divided into two sub-algorithms: $\mathsf{FHE.decode}_0$ and $\mathsf{FHE.decode}_1$ as defined in Section 6.4.2.
- A deterministic UF-CMA signature scheme[3] $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$.
- A context hiding homomorphic signature scheme $\mathsf{HS} = (\mathsf{HS.PrmsGen}, \mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Process}, \mathsf{HS.Verify}, \mathsf{HS.Hide}, \mathsf{HS.HVerify})$ to provide robustness.
- An N out of N secret sharing scheme Share.

Construction.

$\mathsf{TS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following.

1. For each party P_i, sample a PRF key $\mathsf{sprf}_i \leftarrow \mathcal{K}$.
2. Generate the signature scheme's keys $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
3. Generate the FHE keys $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and compute an FHE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.

[3] Any randomized signature scheme can be made deterministic by using PRF to generate the randomness used by the signing algorithm.

4. Generate the HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$ and the public and the signing keys $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here n is the bit-length of $(\mathsf{FHE.sk}, \mathsf{sprf}_i)$.

5. Share $\mathsf{FHE.sk}$ as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{FHE.sk}$.

6. For each party P_i, randomly choose a tag $\tau_i \in \{0,1\}^*$ and compute $(\pi_{\tau_i}, \pi_i) \leftarrow \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, \mathsf{sprf}_i), \tau_i)$.

7. Output $\mathsf{TSig.pp} = \{\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \mathsf{sprf}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, a partial signing key $\mathsf{TSig.sk}_i$ and a message M, parse $\mathsf{TSig.pp}$ as $(\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\})$ and $\mathsf{TSig.sk}_i$ as $(\mathsf{sk}_i, \mathsf{sprf}_i, \pi_i)$ and do the following.

1. Compute $u = F(\mathsf{sprf}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u)$, where $\mathcal{D}_s(u)$ represents sampling from $\mathcal{D}_s$ using u as the randomness.

2. Let C_M be the signing circuit, with message M being hardwired. Compute $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.

3. Compute $\sigma_i = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$.

4. This step computes a homomorphic signature $\tilde{\pi}_i$ on partial signature σ_i to provide robustness (Definition 6.8).
   Let C_{PS} be the circuit to compute $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$ in which CT_σ is hardcoded and the FHE key share sk_i and the PRF key sprf_i are given as inputs.

   - Compute $\pi_i^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, C_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, \mathsf{sprf}_i), \pi_i)$.
   - Compute $\tilde{\pi}_i = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_i, \pi_i^*)$.

5. Output $y_i = (\sigma_i, \tilde{\pi}_i)$.

$\mathsf{TS.Combine}(\mathsf{TSig.pp}, \{y_i\}_{i\in[N]})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{y_i\}_{i\in[N]}$, parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and output $\sigma_M = \mathsf{FHE.decode}_1(\sum_{i=1}^N \sigma_i)$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M, y_i)$: Upon input the public parameters $\mathsf{TSig.pp}$, message M, and a partial signature y_i, parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and do the following.

1. Compute $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Compute $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$, where C_{PS} is as described above.
3. Parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and output $\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i))$.

$\mathsf{TS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input the verification key $\mathsf{TSig.vk}$, a message M and a signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

In the above, we set $s = B_{eval} \cdot \sqrt{Q\lambda}$, where $B_{eval} \leq \mathrm{poly}(\lambda)$ is a bound on the FHE decryption noise after homomorphic evaluation of the signing circuit C_M, and Q is the bound on the number of signatures.

Correctness

From the correctness of $\mathsf{FHE.Eval}$ algorithm, CT_σ is an encryption of $C_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the FHE secret key $\mathsf{FHE.sk}$.

So, $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the $\mathsf{TS.Combine}$ algorithm is

$$\begin{aligned}
\mathsf{FHE.decode}_1\Big(\sum_{i=1}^{N} \sigma_i\Big) &= \mathsf{FHE.decode}_1\Big(\textstyle\sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + \sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{FHE.decode}_1\Big(\mathsf{FHE.decode}_0\big(\textstyle\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_\sigma\big) + \sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{FHE.decode}_1\Big(\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) + \textstyle\sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{FHE.decode}_1\Big(\sigma_M \lfloor q/2 \rceil + e + \textstyle\sum_{i=1}^{N} e'_i\Big) = \sigma_M.
\end{aligned}$$

6.5.2 Unforgeability

For security, we prove the following theorem.

Theorem 6.4. *Assume F is a secure PRF,* Sig *is a UF-CMA secure signature scheme,* FHE *is a secure fully homomorphic encryption scheme (Definition 6.12),* Share *is a secret sharing scheme that satisfies privacy (Definition 6.28) and* HS *is a context hiding secure homomorphic signature scheme (Definitions 6.21). Then our construction of*

threshold signatures satisfies selective unforgeablity (Definition 6.7) if the flooding noise is of the size $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$*, where* Q *is the number of the signing queries.*

Proof. The security of the construction can be argued using a sequence of hybrids. We assume w.l.o.g. that the adversary $\mathcal{A}$ queries for all but the first key share, i.e., $S = [N] \setminus \{1\}$.

Hybrid_0: This is the real world.

Hybrid_1: Same as Hybrid_0, except that $\tilde{\pi}_1$ in $\mathsf{PartSign}$ is now generated using HS simulator as $\tilde{\pi}_1 = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$.

Hybrid_2: Same as Hybrid_1 except that to compute $\sigma_1 = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1$, the randomness u used to sample $e'_1 \leftarrow \mathcal{D}_s(u)$ is chosen uniformly randomly instead of computing it using the PRF.

Hybrid_3: Same as Hybrid_2, except that now, for signing query for $(M, 1)$, the challenger simulates σ_1 as follows:

1. Computes $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\{\sigma'_i = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma)\}_{i \in [2,N]}$.
2. Computes $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$ and set $\sigma_1 = \sigma_M \left\lfloor \frac{q}{2} \right\rceil - \sum_{i=2}^{N} \sigma'_i + e'_1$, where $e'_1 \leftarrow \mathcal{D}_s$.

Hybrid_4: Same as Hybrid_3 except that instead of sharing $\mathsf{FHE.sk}$, now the challenger generates the FHE key shares as $\{\mathsf{sk}_i\}_{i=1}^{N} \leftarrow \mathsf{Share}(\mathbf{0})$.

Hybrid_5: Same as Hybrid_4, except that $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

Indistinguishability of Hybrids. Now, we show that consecutive hybrids are indistinguishable.

Claim 6.5. Assume HS is a context hiding homomorphic signature scheme. Then, Hybrid_0 and Hybrid_1 are indistinguishable.

Proof. The two hybrids differ only in the way $\tilde{\pi}_1$ is computed. In Hybrid_0, $\tilde{\pi}_1 = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_1, \pi_1^*)$, where $\pi_1^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, C_{\mathsf{PS}}, \pi_{\tau_1}, (\mathsf{sk}_1, \mathsf{sprf}_1), \pi_1)$. In Hybrid_1, $\tilde{\pi}_1 = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$. Hence, the two hybrids are indistinguishable because of the context hiding property of HS which ensures that $\mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_1, \pi_1^*) \approx \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$. ■

Claim 6.6. Assume F is a secure PRF. Then Hybrid_1 and Hybrid_2 are indistinguishable. The proof follows via a standard reduction to PRF security and is omitted.

Claim 6.7. If there is an adversary that can win the unforgeability game in Hybrid_2 with probability ϵ, then its probability of winning the game in Hybrid_3 is at least $\epsilon^2/2$.

Proof. Let the number of signing queries that the adversary makes be Q. The two hybrids differ only in the error term in σ_1, as shown below. In Hybrid_2, we have $\sigma_1 = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1'$, for $e_1' \leftarrow \mathcal{D}_s$. In Hybrid_3, we have:

$$\begin{aligned}
\sigma_1 &= \sigma_M . \lfloor q/2 \rceil - \sum_{i=2}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M . \lfloor q/2 \rceil - \sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M . \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M . \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{sk}, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M . \lfloor q/2 \rceil - \sigma_M . \lfloor q/2 \rceil + e + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + (e_1' + e),
\end{aligned}$$

for some e satisfying $|e| \leq B_{eval}$. Thus, in Hybrid_2, the error term in σ_1 is e_1', while in Hybrid_3, it is $e_1' + e$, where, $e_1' \leftarrow \mathcal{D}_s$, and e is the error in FHE ciphertext CT_σ.

Recall the distribution seen by the adversary – the public parameters TSig.pp, the

verification key $\mathsf{TSig.vk}$, the corrupted secret key shares $\mathsf{TSig.sk}_i$, the messages M_j and corresponding partial signatures $(\sigma_j, \tilde{\pi}_j)$. Note that since messages are chosen adaptively, their distribution depends on previous signature queries and responses, and in particular on the differently generated error terms in both hybrids. On the other hand $\mathsf{TSig.pp}$, $\mathsf{TSig.vk}$, $\{\mathsf{TSig.sk}_i\}$, $\{\tilde{\pi}_j\}$ are constructed identically in both hybrids and independently from the rest (in particular these error terms): we implicitly assume that they are fixed and known, and exclude them from the analysis. We refer to the distribution to be considered in Hybrid_2 as D_2 and in Hybrid_3 as D_3.

Let E_j be the random variables corresponding to the error term in CT_{σ_j} in the j-th response and $\mathcal{E}_j^{(2)}$ and $\mathcal{E}_j^{(3)}$ be their distributions in Hybrids 2 and 3, respectively. Similarly, let M_j be the random variable corresponding to the queried message in j-th query and $\mathcal{M}_j^{(2)}$ and $\mathcal{M}_j^{(3)}$ be their distributions in Hybrids 2 and 3, respectively. Then, from the discussion above, we have $\mathcal{E}_j^{(2)} = \mathcal{D}_s$ and $\mathcal{E}_j^{(3)} = \mathcal{D}_{s,e_j}$ for all $j \in [Q]$, where e_j is the error in CT_{σ_j} and can depend upon previous queries and responses.

Overall, we have $D_k = (\mathcal{E}_Q^{(k)}, \mathcal{M}_Q^{(k)}, \mathcal{E}_{Q-1}^{(k)}, \mathcal{M}_{Q-1}^{(k)}, \ldots, \mathcal{E}_1^{(k)}, \mathcal{M}_1^{(k)})$ for $k \in \{2, 3\}$ and

$$R_a(D_2 \| D_3) = R_a(\mathcal{E}_Q^{(2)}, \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{E}_Q^{(3)}, \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}). \quad (6.2)$$

Applying the multiplicativity property of the Rényi divergence (Lemma 6.1), we obtain that $R_a(D_2 \| D_3)$ is bounded from above by

$$\begin{aligned} &\max_{x \in X} R_a(\mathcal{E}_Q^{(2)} | X = x \parallel \mathcal{E}_Q^{(3)} | X = x) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\ = &\max_{x \in X} R_a(\mathcal{D}_s | X = x \parallel \mathcal{D}_{s,e_Q} | X = x) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}), \end{aligned} \quad (6.3)$$

where $X = (M_Q, E_{Q-1}, \ldots, E_1)$ and e_Q is the error term in CT_{σ_Q}; note that e_Q may depend on the sample from X (which differs in Hybrids 2 and 3) and is bounded by B_{eval}.

Then applying Lemma 6.2 in Equation (6.3), we get

$$\begin{aligned} R_a(D_2\|D_3) &\leq \exp(a\pi\|e_Q\|^2/s^2)\cdot R_a(\mathcal{M}_Q^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}) \\ &\leq \exp(a\pi B_{eval}^2/s^2)\cdot R_a(\mathcal{M}_Q^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}). \end{aligned}$$

Further, since M_Q is a function of $E_{Q-1}, M_{Q-1}, \ldots, E_1, M_1$, the data processing inequality (Lemma 6.1) gives

$$\begin{aligned} &R_a(\mathcal{M}_Q^{(2)},\mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)},\mathcal{E}_{Q-1}^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}) \\ &\qquad\qquad \leq R_a(\mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}), \end{aligned}$$

Hence, we get

$$\begin{aligned} R_a(D_2\|D_3) &\leq \exp(a\pi B_{eval}^2/s^2)\cdot R_a(\mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}) \\ &\leq \exp(a\pi B_{eval}^2 Q/s^2), \end{aligned}$$

where the last inequality follows from induction.

As $s = B_{eval}\cdot\sqrt{Q\lambda}$, we get $R_a(D_2\|D_3) \leq \exp(a\pi/\lambda)$. Therefore, from the probability preservation property of the Rényi divergence (Lemma 6.1), we have $D_3(\mathsf{E}) \geq \frac{D_2(\mathsf{E})^{\frac{a}{a-1}}}{R_a(D_2\|D_3)} \geq D_2(\mathsf{E})^{\frac{a}{a-1}}\exp(-\frac{a\pi}{\lambda})$. The result is obtained by setting $a = 2$. ■

Claim 6.8. Assume that Share is a secret sharing scheme that satisfies privacy (Definition 6.28). Then, Hybrid_3 and Hybrid_4 are indistinguishable.

Proof. The only difference between Hybrid_3 and Hybrid_4 is in the way the key shares $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N$ are generated. In Hybrid_3 $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N)\leftarrow\mathsf{Share}(\mathsf{FHE.sk})$, while in Hybrid_4, $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N)\leftarrow\mathsf{Share}(\mathbf{0})$. Since, the adversary is given secret shares for an invalid set of parties, distribution in the two hybrids are identical. ■

Claim 6.9. Assume FHE is a fully homomorphic encryption that satisfies security (Definition 6.12). Then Hybrid_4 and Hybrid_5 are indistinguishable.

Proof. Let $\mathcal{A}$ be an adversary who can distinguish Hybrid_4 and Hybrid_5. Then we construct an adversary $\mathcal{B}$ against the FHE scheme as follows.

1. After receiving $\mathsf{FHE.pk}$ from the FHE challenger, $\mathcal{B}$ generates $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and the HS keys.

2. It generates secret shares of $\mathbf{0}$ as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathbf{0})$.

3. It sends the challenge messages $m_0 = \mathsf{Sig.sk}$ and $m_1 = \mathbf{0}$ to the FHE challenger.

4. After receiving the challenge ciphertext CT_b from the FHE challenger, $\mathcal{B}$ constructs $\mathsf{TSig.pp}$ using CT_b. It also generates $\mathsf{TSig.vk}$ and $\mathsf{TSig.sk}$ as defined for the hybrid. In particular, note that in both the hybrids, the key shares $\{\mathsf{sk}_i\}_{i=1}^N$ are generated as random secret shares of $\mathbf{0}$ in place of $\mathsf{FHE.sk}$ and hence $\mathcal{B}$ does not need $\mathsf{FHE.sk}$ to answer key queries.

5. To answer a $\mathsf{PartSign}$ query for a message M issued by adversary $\mathcal{A}$, $\mathcal{B}$ computes σ_1 as follows. It computes $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$, samples $e'_1 \leftarrow \mathcal{D}_s$ and returns $\sigma_1 = \sigma - \sum_{i=2}^N \mathsf{TS.decode}_0(\mathsf{TSig.sk}_i, \mathsf{CT}_b) + e'_1$.

6. Finally, $\mathcal{A}$ outputs a guess bit b'. $\mathcal{B}$ returns the same to the FHE challenger.

Clearly, if $b = 0$, then $\mathcal{B}$ simulates Hybrid_4, else Hybrid_5 with $\mathcal{A}$. Hence if $\mathcal{A}$ wins with non-negligible probability in distinguishing the two hybrids then so does $\mathcal{B}$ against the FHE challenger. ■

Finally the proof of Theorem 6.4 completes with the following claim.

Claim 6.10. If the underlying signature scheme Sig is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 6.7 is negligible in Hybrid_5.

Proof. Let $\mathcal{A}$ be an adversary who wins the unforgeability game in Hybrid_5. Then we can construct an adversary $\mathcal{B}$ against the signature scheme Sig as follows:

1. On receiving a verification key $\mathsf{Sig.vk}$ from Sig challenger, $\mathcal{B}$ generates $(\mathsf{FHE.sk}, \mathsf{FHE.pk})$, $\mathsf{HS.pp}$, $(\mathsf{HS.pk}, \mathsf{HS.sk})$ and all the other values required to define $\mathsf{TSig.pp}$, $\mathsf{TSig.vk}$ and $\mathsf{TSig.sk}$ on its own. In particular, since in Hybrid_5, $\mathsf{TSig.pp}$ contains $\mathsf{CT}_{\mathbf{0}}$ instead of $\mathsf{CT}_{\mathsf{Sig.sk}}$, $\mathcal{B}$ does not require $\mathsf{Sig.sk}$ to generate a valid $\mathsf{TSig.pp}$.

2. $\mathcal{B}$ then sends $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.pp}$, $\{\mathsf{TSig.sk}_i\}_{i=2}^N$ to $\mathcal{A}$.

3. To simulate $\mathsf{PartSign}$ query σ_1 for any message M, $\mathcal{B}$ needs σ_M. For this, it issues a signing query on message M to the Sig challenger and receives σ_M.

4. In the end, let (M^*, σ^*) be the forgery returned by $\mathcal{A}$. Then $\mathcal{B}$ returns the same to the Sig challenger.

Since $\mathcal{B}$ issues signing queries on only those messages for which $\mathcal{A}$ also issues signing queries to $\mathcal{B}$, if (M^*, σ^*) is a valid forgery for $\mathcal{A}$, then it is a valid forgery for $\mathcal{B}$ as well. ■

6.5.3 Robustness

Claim 6.11. If HS is multi data secure (Definition 6.20) homomorphic signature, then the above construction of TS satisfies robustness.

Proof. In the robustness security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},rb}(1^\lambda)$, the adversary wins if $\mathcal{A}$ outputs a partial signature forgery (M^*, y_i^*, i) such that

1. $\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M^*, y_i^*) = 1$

2. $y_i^* = (\sigma_i^*, \tilde{\pi}_i^*) \neq \mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M^*)$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M^*, y_i^*)$ first computes $\mathsf{CT}_\sigma \leftarrow \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_{M^*}, \mathsf{CT}_{\mathsf{Sig.sk}})$ and outputs 1 iff $\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i^*, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i^*)) = 1$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$. Thus, $\mathcal{A}$ wins the experiment iff both the following two conditions are true.

1. For $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$

$$\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i^*, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i^*)) = 1,$$

2. $y_i^* = (\sigma_i^*, \tilde{\pi}_i^*) \neq \mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M^*)$, which implies $\sigma_i^* \neq \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$, which in turn is same as

$$\sigma_i^* \neq C_{\mathsf{PS}}(\mathsf{sk}_i, \mathsf{sprf}_i).$$

But this is a case for valid forgery of type 2 (Definition 6.20) against HS scheme, which can happen only with negligible probability. Note that since (τ_i, π_{τ_i}) are part of $\mathsf{HS.pp}$,

case of type 2 in HS security definition is inherently applied. ■

6.5.4 On the Optimality of Our Flooding

We show that the flooding amount that we achieved is optimal for our threshold signature scheme. To argue this, we show how to attack it if the flooding amount is below $\Omega(\sqrt{Q})$. For simplicity, we restrict to the case of $N = 2$. Recall that in our construction, $\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$ outputs $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_{i,M}$, where $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \mathsf{sprf}_i)$.[4] W.l.o.g, assume that the adversary gets the partial signing key $\mathsf{TSig.sk}_2$ and the response for any signing query is a partial signature corresponding to party P_1. For any message M of its choice, the adversary receives $\sigma_{1,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_{1,M}$. From this the adversary can compute:

$$\begin{aligned} \sigma_{1,M} + \mathsf{FHE.decode}_0(\mathsf{sk}_2, \mathsf{CT}_{\sigma_M}) &= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + e'_{1,M} \\ &= \sigma_M + \mathsf{err}_M + e'_{1,M}, \end{aligned}$$

where err_M is the error in CT_{σ_M}. Note that if the adversary succeeds in computing err_M for polynomially many M's, then it can compute $\mathsf{FHE.sk}$.

As a warm-up, we show that if the error $e'_{1,M}$ is randomized, small and of center 0, then the adversary can indeed compute err_M. Later, we will show that even for deterministic flooding $e'_{1,M}$, there exist secure signature schemes for which the attack can be extended. Since the adversary knows the key share sk_2, it can compute $\sigma_{2,M}$ on its own and hence can compute $\sigma_M = \mathsf{TS.Combine}(\mathsf{TSig.pp}, \sigma_{1,M}, \sigma_{2,M})$. Hence, from $\sigma_M + \mathsf{err}_M + e'_{1,M}$, the adversary can compute $\mathsf{err}_M + e'_{1,M}$. Since, the signature scheme is deterministic, err_M depends only on M. Thus, if the same message is queried for signature multiple times, then each time the term err_M remains the same, but since flooding is randomized, the term $e'_{1,M}$ is different.

To compute err_M, the adversary issues all Q signing queries for the same message M and receives $\sigma^{(1)}_{1,M}, \ldots, \sigma^{(Q)}_{1,M}$, where $\sigma^{(i)}_{1,M}$ denotes the partial signature returned for message

[4] We focus only on the $\sigma_{i,M}$ component of PartSign's output since the second component, the HS signature of $\sigma_{i,M}$, is not relevant here.

M in the ith query. From these responses the adversary gets Q different values of the form

$$w^i = \mathsf{err}_M + e'^i_{1,M} \tag{6.4}$$

Since err_M is same, taking average on both sides of Equation (6.4) over all the Q samples, we get $\frac{\sum_{i\in[Q]} w^i}{Q} = \mathsf{err}_M + \frac{\sum_{i\in[Q]} e'^i_{1,M}}{Q}$. If $|\frac{1}{Q}\sum_{i\in[Q]} e'^i_{1,M}| < 1/2$, then the adversary can recover err_M as $\mathsf{err}_M = \left\lfloor \frac{1}{Q}\sum_{i\in[Q]} w^i \right\rceil$. As $e'^1_{1,M}, \ldots, e'^Q_{1,M}$ are independently and identically distributed with mean 0, by Hoeffding's inequality, we have

$$\Pr\left[\left|\frac{\sum_{i\in[Q]} e'^i_{1,M}}{Q}\right| < 1/2\right] \geq 1 - 2\exp\left(-\frac{Q}{2s^2}\right).$$

If $Q \geq \Omega(s^2 \log \lambda)$, then $1 - 2\exp(-Q/(2s^2)) \geq 1 - \lambda^{-\Omega(1)}$, in which case the adversary can recover err_M with probability sufficiently close to 1 to recover sufficiently many err_M's to compute $\mathsf{FHE.SK}$. To prevent this, we do need s to grow at least proportionally to $\sqrt{Q}$.

Attack for Deterministic Error

In the argument for randomized error, the fact that $e'^i_{1,M}$ is randomized is crucial. However, as discussed in Section 6.1, we can extend the attack for the case of deterministic flooding as well, by exhibiting a secure signature scheme (with deterministic flooding) for which a variant of the attack can apply.

Consider a special (contrived) signature scheme $\mathsf{Sig}' = (\mathsf{Sig}'.\mathsf{KeyGen}, \mathsf{Sig}'.\mathsf{Sign}, \mathsf{Sig}'.\mathsf{Verify})$ derived from a secure signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ as follows:

1. $\mathsf{Sig}'.\mathsf{KeyGen}$ is identical to $\mathsf{Sig.KeyGen}$. Let $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$ be the signing and verification keys, respectively, and $\mathsf{Sig.sk}_i$ denote the ith bit of $\mathsf{Sig.sk}$ for $i \in [\ell]$, where ℓ is the bit-length of $\mathsf{Sig.sk}$.

2. $\mathsf{Sig}'.\mathsf{Sign}(\mathsf{Sig.sk}, M)$ is modified as follows:

   - Compute $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$. Set $\sigma'_M := \sigma_M$.
   - For i from 1 to ℓ: if $\mathsf{Sig.sk}_i = 0$, then set $\sigma'_M := \sigma'_M \| \mathsf{Sig.sk}_i$.

- Output σ'_M.

3. $\mathsf{Sig}'.\mathsf{Verify}(\mathsf{Sig.vk}, M, \sigma'_M)$ is defined as $\mathsf{Sig.Verify}(\mathsf{Sig.vk}, M, \sigma_M)$, where σ_M is obtained from σ'_M by removing all the bits after the kth bit, where k is the bit-length of signatures in Sig.

Above, we assume that the signing key of Sig is a uniform bit-string among those with the same number of 0's and 1's. Since $\mathsf{Sig.sk}$ has always $\ell/2$ bits equal to 0, the number of zeroes appended to the signature will be $\ell/2$ and hence does not leak any extra information to the adversary. Hence, it follows easily that if Sig is a secure signature scheme, then so is Sig'. However, as discussed in Section 6.1, our attack can be generalized to work for this setting.

The Attack Now, consider using Sig' to instantiate our threshold signature scheme. Then, for any message M, the FHE ciphertext CT_{σ_M} now additionally includes homomorphically evaluated encryptions of $\{\mathsf{Sig.sk}_i\}_{i\in[\ell]:\mathsf{Sig.sk}_i=0}$. Let $\mathsf{CT}_{\sigma_M}, \mathsf{err}_M, e'_M$ respectively denote the encryption of σ_M, the error in CT_{σ_M} and the flooding noise added to partial decryption of CT_{σ_M}. Let $\mathsf{CT}^*, \mathsf{err}^*$ and e^*_M denote the components corresponding to $\{\mathsf{Sig.sk}_i\}_{i\in[\ell]:\mathsf{Sig.sk}_i=0}$.

For any message M, the adversary can compute $\mathsf{err}_M + e'_M$ as described previously, from which the adversary gets $\mathsf{err}^* + e^*_M$. If the adversary manages to compute err^* (for sufficiently many instances), then it can also recover $\mathsf{FHE.sk}$.

Note that err^* is independent of any message and hence is constant across different messages, while e^*_M does depend on M and is different for different messages. This gives an attack strategy. To compute err^*, the adversary issues Q signing queries on different messages $\{M_j\}_{j\in[Q]}$, and from the received partial signatures, derives the values for

$$w^*_j = \mathsf{err}^* + e^*_{M_j} \quad \text{for } j \in [Q].$$

Observe that the above equation is of the same form as Equation (6.4). Heuristically, one would expect the $e^*_{M_j}$ to behave as independent and identically distributed random

variables with centre 0. Hence, we can argue in similar way that if $Q \geq \Omega(s^2 \log \lambda)$ then the adversary can recover err^* with probability $1 - 1/\mathrm{poly}(\lambda)$. This implies that for hiding err^*, the standard deviation parameter s must grow at least proportionally to $\sqrt{Q}$.

6.6 INSTANTIATING THRESHOLD SIGNATURES: REJECTION-FREE LYUBASHEVSKY

Here, we provide an FHE friendly variant of Lyubashevsky's signature scheme from [Lyu12].

6.6.1 Construction

Our construction uses a hash function $H : \{0,1\}^* \to D_H := \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^k; \|\mathbf{v}\|_1 \leq \alpha\}$, modeled as a random oracle. Here α is a parameter, typically much smaller than k.

$\mathsf{KeyGen}(1^\lambda)$: Upon input the security parameter λ, set $q, n, m, \beta, k, d, \sigma$ such that $n = \Omega(\lambda)$ and the scheme is secure (see Theorem 6.12); then do the following:

1. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{S} \leftarrow \{-d, \ldots, 0, \ldots, d\}^{m \times k}$.
2. Set $\mathbf{T} = \mathbf{AS}$.
3. Output $\mathsf{vk} = (\mathbf{A}, \mathbf{T})$, $\mathsf{sk} = \mathbf{S}$.

$\mathsf{Sign}(\mathsf{sk}, \mu)$: Upon input the signing key sk and a message μ, do the following:

1. Sample $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma}$.
2. Set $\mathbf{c} = H(\mathbf{Ay}, \mu)$.
3. Set $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$.
4. Output $(\mathbf{z}, \mathbf{c})$.

$\mathsf{Verify}(\mathsf{vk}, \mu, (\mathbf{z}, \mathbf{c}))$: Upon input the verification key vk, a message μ, and a signature $(\mathbf{z}, \mathbf{c})$, do the following:

1. Check if $\|\mathbf{z}\| \leq \gamma$, where $\gamma = (2\sigma + \alpha d)\sqrt{m}$.
2. Check if $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
3. If both checks pass, then accept, else reject.

Correctness. Since $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$, where $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, we have $\|\mathbf{z}\| \leq 2\sigma\sqrt{m} + \|\mathbf{Sc}\|$ with probability $1 - 2^{-\Omega(\lambda)}$, using standard Gaussian tail bounds (see, e.g., Lemma 6.3). Since $\|\mathbf{S}\|_\infty \leq d$ and $\|\mathbf{c}\|_1 \leq \alpha$, we have $\|\mathbf{Sc}\| \leq d\alpha\sqrt{m}$. This gives us $\|\mathbf{z}\| \leq (2\sigma + d\alpha)\sqrt{m}$ with overwhelming probability. Finally, note that

$$H(\mathbf{Az} - \mathbf{Tc}, \mu) = H(\mathbf{A}(\mathbf{y} + \mathbf{Sc}) - \mathbf{ASc}, \mu) = H(\mathbf{Ay}, \mu) = \mathbf{c}.$$

6.6.2 Security

We establish security via the following theorem.

Theorem 6.12. *Assume that $m > \lambda + (n \log q)/\log(2d + 1)$, $\sigma \geq \alpha d\sqrt{mQ}$ where Q is the maximum number of signing queries an attacker can make and $|D_H| \geq 2^\lambda$. Assume further that $\mathsf{SIS}_{q,n,m,\beta}$ is hard for $\beta = 2\gamma + 2d\alpha\sqrt{m}$. Then the construction in Section 6.6.1 satisfies* UF-CMA *in the random oracle model.*

Proof. We prove the security via the following hybrids:

Hybrid_0: This is the genuine security game, i.e., with honest executions of the Sign algorithm on signing queries by the adversary.

Hybrid_1: In this hybrid the challenger responds to the signing query for any message μ as follows.

1. Sample $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ as in the previous hybrid.
2. Sample $\mathbf{c} \leftarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \alpha\}$.
3. Set $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$.
4. Set $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.

5. Output $(\mathbf{z}, \mathbf{c})$.

Hybrid_2: In this hybrid the challenger responds to the signing query for any message μ as follows.

1. Sample $\mathbf{c} \leftarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \alpha\}$.
2. Sample $\mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma}$.
3. Set $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
4. Output $(\mathbf{z}, \mathbf{c})$.

The only difference between Hybrid_0 and Hybrid_1 is that in Hybrid_1, the output value for H is chosen at random, and then programmed as the answer to $H(\mathbf{Ay}, \mu)$, without checking whether the value for $(\mathbf{Ay}, \mu)$ is already set, when a signing query for μ is made. By the definition of the random oracle, the two hybrids are identical if the same input $(\mathbf{Ay}, \mu)$ is not programmed twice throughout hash and sign queries, and forgery. The distinguishing advantage is therefore bounded as $Q(Q + Q_H + 1) \cdot 2^{-h}$, where h is the min-entropy of $\mathbf{Ay}$ for $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma}$. Standard arguments show that this is negligible.

The result now follows from the two claims below.

Claim 6.13. If there is an adversary that makes at most Q signing queries and can win the game in Hybrid_1 with probability δ, then its probability of winning in Hybrid_2 is polynomial in δ, if $\sigma \geq \alpha d \sqrt{mQ}$.

Proof. Wlog, we assume that the adversary makes exactly Q queries. The only difference between the two hybrids is in the value of $\mathbf{z}$. Let us refer to the joint distribution of all $(\mathbf{z}, \mathbf{c})$'s in Hybrid_1 as D_1 and that in Hybrid_2 as D_2. Note that the $\mathbf{c}_i$'s are sampled identically in both hybrids, and independently from all the rest. For $i \in [Q]$, the vector $\mathbf{z}_i$ is from distribution $Z_{1i} := \mathcal{D}_{\mathbb{Z}^m, \sigma, \mathbf{Sc}_i}$ in Hybrid_1 and from distribution $Z_{2i} = \mathcal{D}_{\mathbb{Z}^m, \sigma}$

in Hybrid_2. By Lemma 6.2, we have

$$R_a[Z_{1i}\|Z_{2i}] = \exp\left(a\pi\frac{\|\mathbf{S}\mathbf{c}_i\|^2}{\sigma^2}\right) \quad \text{for any } a \in (1,\infty).$$

Recall from the correctness proof that we have $\|\mathbf{S}\mathbf{c}_i\| \leq d\alpha\sqrt{m}$.

Let D_{1i} (resp. D_{2i}) be the distribution of $(\mathbf{z}_i, \mathbf{c}_i)$'s in Hybrid_1 (resp. Hybrid_2). As $\mathbf{c}_i$ is identically distributed in both games, we have, by using the multiplicativity property of Rényi divergence (Lemma 6.1):

$$R_a[D_{1i}\|D_{2i}] \leq 1 \cdot \max_{\mathbf{c}_i} R_a[Z_{1i}\|Z_{2i}] \leq \exp\left(a\pi\frac{(d\alpha\sqrt{m})^2}{\sigma^2}\right).$$

As $D_1 = (D_{1i})_i$ and $D_2 = (D_{2i})_i$, by using the multiplicativity property of the Rényi divergence once more, we get:

$$R_a(D_1\|D_2) \leq \exp\left(a\pi\frac{Q(d\alpha\sqrt{m})^2}{\sigma^2}\right) \leq \exp(a\pi), \quad \text{for any } a \in (1,\infty). \tag{6.5}$$

Now, the view of the adversary in both hybrids includes the verification key vk, the queried messages M_i and the signature replies $(\mathbf{z}_i, \mathbf{c}_i)$ for $i \in [Q]$. As the distribution of vk is identical in both games and vk is sampled independently from all the rest, we may implicitly assume that it is fixed. As they are chosen adaptively, the μ_i's may depend on the previous queries and replies. But the dependence of the responses on the messages is broken by the random oracle (unlike in the proof of Claim 6.7). Hence, the $(\mathbf{c}_i, \mathbf{z}_i)$'s are independent of the μ_i's in both the hybrids. As the μ_i's are functions of the $(\mathbf{c}_i, \mathbf{z}_i)$'s, by the data processing inequality of the Rényi divergence (Lemma 6.1), we have

$$R_a(V_1\|V_2) \leq R_a(D_1\|D_2), \tag{6.6}$$

where V_1 (resp. V_2) is the adversary's view in Hybrid_1 (resp. Hybrid_2).

Let E denote the event that the adversary wins the game. Then by our assumption, we have $D_1(\mathsf{E}) = \delta$. From the probability preservation property (Lemma 6.1) of the Rényi

divergence, we get:

$$V_2(\mathsf{E}) \geq \frac{\delta^{\frac{a}{a-1}}}{R_a(V_1 \| V_2)}, \text{ for any } a \in (1, \infty). \tag{6.7}$$

Using Equations (6.5), (6.6) and (6.7), we obtain that $V_2(\mathsf{E}) \geq \delta^{\frac{a}{a-1}} \exp(-a\pi)$. Taking any value of $a > 1$ provides the result. ∎

Claim 6.14. Let D_H be the range of the random oracle H. If there is a forger $\mathcal{F}$ that makes at most Q signing queries and Q_H random oracle queries, and succeeds in forging a valid signature with probability δ in Hybrid_2, then we can define an algorithm $\mathcal{B}$ which given $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, finds a non-zero $\mathbf{v}$ such that $\|\mathbf{v}\| \leq (2\gamma + 2d\alpha\sqrt{m})$ and $\mathbf{A}\mathbf{v} = \mathbf{0}$, with probability at least

$$\left(\frac{1}{2} - \frac{\varepsilon}{2}\right)\left(\delta - \frac{1}{|D_H|}\right)\left(\frac{\delta - 1/|D_H|}{Q_H + Q} - \frac{1}{|D_H|}\right).$$

This claim and its proof are identical to [Lyu12, Lemma 5.4]. Note that under the conditions of Theorem 6.12, the latter probability lower bound is $\geq \delta^2/(2(Q_H + Q)) - 2^{-\Omega(\lambda)}$. ∎

Note that the condition $\sigma \geq \alpha d\sqrt{mQ}$ from Theorem 6.12 forces to set a modulus q and a SIS bound β that grow linearly with $\sqrt{Q}$. To ensure λ bits of security, one may choose n growing linearly with $\sqrt{Q}$. Overall, if using a Ring-SIS or Module-SIS instantiation, then the bit-length of the signature grows linearly with $n \log q$ and hence with $\log^2 Q$.

Next, we show that the flooding noise used in the above construction is essentially optimal by exhibiting an attack when the flooding noise is smaller.

6.6.3 Optimality of Flooding

In this section, we show that the flooding amount used in the construction in Section 6.6.1 is essentially optimal, and in particular that the dependence on $\sqrt{Q}$ is necessary. In more detail, we show that if the flooding noise is smaller than this, then an adversary can recover the signing key. Note that this attack is folklore, we recall it for the sake of

completeness.

Statistical Attack

Recall that the signature for message M_i is of the form $(\mathbf{z}_i, \mathbf{c}_i)$, where $\mathbf{z}_i = \mathbf{S}\mathbf{c}_i + \mathbf{y}_i$, $\mathbf{c}_i \in \{-1, 0, 1\}^k$, $\|\mathbf{c}_i\|_1 \leq \alpha$, and $\mathbf{S}$ is the signing key. The adversary can obtain many such pairs corresponding to different messages. Let Q be the maximum number of signing queries that the adversary can make. Let $\mathbf{S}_i$ represents the ith row of matrix $\mathbf{S}$. Let $\mathbf{c}_{ij}$, $\mathbf{y}_{ij}$ and $\mathbf{S}_{ij}$ represent the jth entry in vectors $\mathbf{c}_i$, $\mathbf{y}_i$ and $\mathbf{S}_i$ respectively. Consider such tuples $(\mathbf{z}_i, \mathbf{c}_i)$ where $\mathbf{c}_{i1} = 1$. Let $B \subseteq [Q]$ be the set of such indices. The adversary gets approximately $Q/3$ such tuples corresponding to $i \in B$. For each i, using the first row of $\mathbf{S}$, we may write:

$$\mathbf{S}_{11} + \sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij} + \mathbf{y}_{i1} = \mathbf{z}_{i1} \tag{6.8}$$

We denote the average of $\sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij} + \mathbf{y}_{i1}$ over $i \in B$ as avg. We show that unless $\mathbf{y}_{i1}$ is $O(\sqrt{Q})$, we can recover $\mathbf{S}_{11}$. To conduct the attack, we bound each summand of avg separately.

Claim 6.15. Let $t_1 < 1/2$ be a positive constant and Q, k, d, α be as above. Then,

$$\Pr\left[\left|\frac{\sum_{i\in B}\sum_{j=2}^{k}\mathbf{S}_{1j}\mathbf{c}_{ij}}{|B|}\right| < t_1\right] \geq 1 - 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right)$$

Proof. Note that $\sum_{j=2}^{k}\mathbf{S}_{1j}\mathbf{c}_{ij}$ takes values in the range $[-(\alpha-1)d, (\alpha-1)d]$, with mean at 0. In more detail, let X be a random variable, with mean 0 and support $[-(\alpha-1)d, (\alpha-1)d]$, then for some positive constant $t_1 < 1/2$, we have from Hoeffding's bound

$$\begin{aligned}
&\Pr[|\bar{X} - E[X]| \geq t_1] && \leq 2\exp\left(\frac{-(Q/3)t_1^2}{2(\alpha-1)^2d^2}\right) \\
\implies \quad &\Pr[|\bar{X}| \geq t_1] && \leq 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right) \\
\implies \quad &\Pr[|\bar{X}| < t_1] && \geq 1 - 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right)
\end{aligned}$$

Since d is small, in particular if $(6(\alpha-1)^2d^2 < Qt_1^2)$, then $1-2\exp(\frac{-Qt_1^2}{6(\alpha-1)^2d^2})$ is non-negligible. ■

Let us assume that the average of $\mathbf{y}_{i1}$ is also smaller than $1/2-t_1$ with non negligible probability. Then, $\mathsf{avg} < 1/2$ with non negligible probability. Summing both sides of Equation 6.8 over the set B, we get $\mathbf{S}_{11} + \mathsf{avg} = \frac{\sum_{i\in B}\mathbf{z}_{i1}}{|B|}$. In this case the adversary can successfully recover $\mathbf{S}_{11}$ as

$$\mathbf{S}_{11} = \left\lfloor \frac{\sum_{i\in B}\mathbf{z}_{i1}}{|B|} \right\rceil$$

We now examine how large $\mathbf{y}_{i1}$ must be to avoid this attack. Let $Y \leftarrow \mathcal{D}_\sigma$ be the random variable representing the distribution of $\mathbf{y}_{i1}$ values. Then from Hoeffding's bound, for some constant c' and $t_2 < (1/2-t_1)$,

$$\begin{aligned} & \Pr[|\bar{Y} - E[Y]| \geq t_2] \leq 2\exp(-c'Qt_2^2/3\sigma^2) \\ \implies & \Pr[|\bar{Y}| \geq t_2] \leq 2\exp(-c'Qt_2^2/3\sigma^2) \\ \implies & \Pr[\ |\bar{Y}| < t_2] \geq 1-2\exp(-c'Qt_2^2/3\sigma^2) \end{aligned}$$

Thus, if $3\sigma^2 < c'Qt_2^2$, then $1-2\exp(-c'Qt_2^2/3\sigma^2)$ is non-negligible. Hence, for the average of $\mathbf{y}_{i1}$ to be greater than t_2, we need that σ must grow proportional to $\sqrt{Q}$.

From Gaussian to Uniform

In some applications, it may be preferable to use a vector $\mathbf{y}$ whose coordinates are uniform in some interval $[-B, B]$ rather than Gaussian (at Step 1 of the `Sign` algorithm). This is the choice made for the regular Dilithium signature scheme DKL+18. Looking ahead, if the sampling of $\mathbf{y}$ is done under a homomorphic encryption layer, this may be significantly simpler to implement.

To adapt the current proof, the only step that needs to be modified is in the transition between Hybrid 1 and Hybrid 2. A difficulty is that the support of the distribution of $\mathbf{z}$ in Hybrid 2 has to contain the support of the distribution of $\mathbf{z}$ in Hybrid 2, for the Rényi divergence to be defined. For this purpose, we consider a wider interval in Hybrid 2,

which contains all possible intervals $[-B, B]^m + \mathbf{Sc}$ of Hybrid 1. Concretely, in Hybrid 1, the vector $\mathbf{z}$ is sampled from $D_1 = U([-B, B]^m) + \mathbf{Sc}$, whereas in Hybrid 2, the vector $\mathbf{z}$ is sampled from $D_2 = U([-B', B']^m)$. As $\|\mathbf{Sc}\|_\infty \leq \alpha d$, we can take $B' = B + \alpha d$. Assuming that both B and B' are integers, we have

$$R_a(D_1 \| D_2) = \left(\frac{2B'+1}{2B+1}\right)^{Q_S m} \leq \left(1 + \frac{2\alpha d}{B}\right)^{Q_S m}, \quad \text{for any } a \in (1, \infty).$$

We obtain that for transiting from Hybrid 1 to Hybrid 2, it suffices to set $B \geq \Omega(md\alpha Q_S)$.

6.7 THRESHOLD SIGNATURES WITH ADAPTIVE SECURITY

As discussed in Section 6.1, we provide two constructions to improve the selective security achieved by [BGG+18]. Below, we describe our construction in the ROM, which satisfies partially adaptive unforgeability (Definition 6.6). We provide our construction in the standard model with *pre-processing* that satisfies fully adaptive unforgeability (Definition 6.5) in Section 6.8.

6.7.1 Construction for Partially Adaptive Unforgeability

We denote our scheme for threshold signatures with partial adaptivity by $\mathsf{pTS} = (\mathsf{pTS.KeyGen}, \mathsf{pTS.PartSign}, \mathsf{pTS.PartSignVerify}, \mathsf{pTS.Combine}, \mathsf{pTS.Verify})$. We use the same building blocks for construction as those used for the non-adaptive construction. We also use two keyed hash function modelled as random oracles: $H : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_q^N$ and $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^r$.

Construction

$\mathsf{pTS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following:

1. Randomly choose $K \leftarrow \{0,1\}^\lambda$ and N vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \in \mathbb{Z}_q^N$ such that $\sum_{i=1}^N \mathbf{v}_i = \mathbf{0}$.

2. Generate $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and share $\mathsf{FHE.sk}$ into N shares as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{FHE.sk}$.

3. Compute an FHE encryption of the signing key as $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.

4. For each party P_i, randomly choose a tag $\tau_i \in \{0,1\}^*$, a hash key $\mathsf{hkey}_i \leftarrow \{0,1\}^\lambda$ and generate HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$ and HS public and signing keys as $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here, n is the length of input to PartSign circuit which depends on $(\mathsf{FHE.sk}, K, \mathbf{v}_i, \mathsf{hkey}_i)$.

5. Compute $(\pi_{\tau_i}, \pi_i) = \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i), \tau_i)$.

6. Output $\mathsf{TSig.pp} = (\mathsf{FHE.pk}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N)$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{pTS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters TSig.pp, the key share $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i, \pi_i)$ and a message M, do the following:

1. Compute $u_i = H_1(\mathsf{hkey}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u_i)$.

2. Let C_M be the signing circuit, with message M being hardcoded. Compute an FHE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.

3. Compute $r_{i,M} = H(K, M)^T \mathbf{v}_i$ and $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,M} + e_i'$.

4. This step computes a homomorphic signature $\tilde{\pi}_{i,M}$ on partial signature $\sigma_{i,M}$ to provide robustness (Definition 6.8).
Let C_{PS} be the circuit to compute $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T \mathbf{v}_i + e_i'$ in which CT_{σ_M} is hardcoded and the key share $\mathsf{TSig.sk}_i$ is given as the input. Compute $\pi^*_{i,M} = \mathsf{HS.SignEval}(\mathsf{HS.pp}, C_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i), \pi_i)$ and $\tilde{\pi}_{i,M} = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_{i,M}, \pi^*_{i,M})$.

5. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

The algorithms pTS.PartSignVerify, pTS.Combine and pTS.Verify are identical to the corresponding algorithms in section 6.5.1.

Correctness

The correctness can be argued in the same way as that in Section 6.5.1. The pTS.Combine algorithm outputs $\mathsf{FHE.decode}_1(\sum_{i=1}^N \sigma_{i,M})$, where

$\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_i + r_{i,M}$. First observe that $\sum_{i=1}^{N} r_{i,M} = \sum_{i=1}^{N} H(K,M)^T \mathbf{v}_i = H(K,M)^T \sum_{i=1}^{N} \mathbf{v}_i = 0$, because $\sum_{i=1}^{N} \mathbf{v}_i = \mathbf{0}$. Hence, $\mathsf{FHE.decode}_1(\sum_{i=1}^{N} \sigma_{i,M}) = \mathsf{FHE.decode}_1((\sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_i) + \sum_{i=1}^{N} r_{i,M}) = \mathsf{FHE.decode}_1(\sum_{i=1}^{N} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_i) + \mathbf{0}) = \sigma_M$, where the last equality can be derived in the same way as in Section 6.5.1.

6.7.2 Unforgeability

We prove that the above construction satisfy partially adaptive unforgeability via the following theorem.

Theorem 6.16. *Assume the signature scheme* Sig *satisfies unforgeability,* FHE *is a semantically secure fully homomorphic encryption scheme (Definition 6.12),* HS *is context hiding homomorphic signature scheme (Definition 6.21) and* Share *satisfies privacy (Definition 6.28). Then the* pTS *construction above satisfies partially adaptive unforgeability (Definition 6.6) in ROM if the flooding error is of size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, where* Q *is the upper bound on the number of signing queries.*

Proof. The theorem can be proved using the following hybrids:

Hybrid_0 and Hybrid_1 are the same as that in the proof of Theorem 6.4.

Hybrid_2: Same as Hybrid_1, except that the randomness u_i used in sampling e'_i in $\sigma_{i,M}$ is chosen uniformly randomly from $\{0,1\}^r$ and then H_1 is programmed as $H_1(\mathsf{hkey}_i, M) = u_i$. For random oracle queries for hash H_1 by the adversary $\mathcal{A}$ on an input x, the challenger first checks if $H_1(x)$ is already set. If so, then returns that value else chooses a value uniformly randomly from $\{0,1\}^r$ and saves and returns it.

Hybrid_3: Same as Hybrid_2 except that the value of $H(K,M)$ for each M in pre corruption signing query is set in the reverse order, i.e., firstly partial signatures are computed

and then $H(K, M)$ is set accordingly as follows:

1. The challenger computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.

2. It then computes $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$ and generates N shares as $\{s_{i,M}\}_{i=1}^{N} \leftarrow \mathsf{Share}(\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}))$.

3. Returns partial signatures as $\{\sigma_{i,M} = s_{i,M} + e'_i\}_{i=1}^{N}$. Also, if a message M is repeated for signing query, then the challenger uses same $\{s_{i,M}\}_{i=1}^{N}$ shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$ again.

4. When the adversary $\mathcal{A}$ outputs the set S of corrupted parties, the challenger first programs the value of $H(K, M)$ for each M in pre corruption signing queries as described next, and then provides key shares for $i \in S$ to $\mathcal{A}$.

   - Programming $H(K, M)$: $\forall i \in [N]$, compute $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$ and solve for vector $\mathbf{b}_M \in \mathbb{Z}_q^N$ such that $\forall\, i \in [N], \mathbf{b}_M^T \mathbf{v}_i = r_{i,M}$. Set $H(K, M) = \mathbf{b}_M$. Note that since there are $N - 1$ independent equations in N unknowns, such a $\mathbf{b}_M$ exists and can be computed.

5. To answer a random oracle query for hash function H on input x, the challenger first checks if the value is already set, if so then returns that value, else randomly samples a fresh vector $\mathbf{r}_x$ and sets and returns $H(x) = \mathbf{r}_x$.

Hybrid_4: Same as Hybrid_3, except that now the signing queries are answered differently.

For each pre-corruption signing query for a message M, the challenger computes $\sigma_{i,M}$ as follows:

1. Computes $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$ and generates random shares of $\sigma_M \lfloor q/2 \rceil$ as $\{s_{i,M}\}_{i=1}^{N} \leftarrow \mathsf{Share}(\sigma_M \lfloor q/2 \rceil)$ such that $\sum_{i=1}^{N} s_{i,M} = \sigma_M \lfloor q/2 \rceil$.

2. Returns $\sigma_{i,M} = s_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$

When $\mathcal{A}$ outputs the set S of corrupted parties, the challenger does the following:

1. Let $PreQ$ be the set of messages for which signing queries were made before. Then for each $M \in PreQ$ it does the following. For each $i \in S$, computes $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$. Computes $\mathbf{b}_M$ such that $\forall\, i \in S,\ \mathbf{b}_M^T \mathbf{v}_i = r_{i,M}$. Sets $H(K, M) = \mathbf{b}_M$. Such a $\mathbf{b}_M$ exists and can be computed since there are only $N - 1$ equations to satisfy in N unknowns.

2. Returns the secret key shares $\{\mathsf{TSig.sk}_i\}_{i\in S}$.

For each post corruption signing query on message M, the challenger does the following. Let the honest party be P_a, i.e. $S = [N] \setminus \{a\}$.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
2. For each $i \in S$, computes $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T \mathbf{v}_i$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.
3. Returns $\sigma_{a,M} = \sigma_M \lfloor q/2 \rceil - \sum_{i\in S} \sigma'_{i,M} + e'_a$, where $e'_a \leftarrow \mathcal{D}_s$.

Hybrid_5 and Hybrid_6: are the same as Hybrid_4 and Hybrid_5, respectively, defined in the proof of Theorem 6.4.

Indistinguishability of Hybrids. Now we show that the consecutive hybrids are indistinguishable.

Claim 6.17. If the underlying homomorphic signature scheme HS is context hiding then Hybrid_0 and Hybrid_1 are indistinguishable.

Proof. The two hybrids differ only in the way $\tilde{\pi}_{i,M}$ is computed. In Hybrid_0 it is computed using HS.SignEval while in Hybrid_1 it is generated by HS simulator. Hence, from the context hiding property of HS, the two hybrids are indistinguishable. ■

Claim 6.18. If H_1 is modeled as random oracle then Hybrid_1 and Hybrid_2 are indistinguishable.

Proof. The two hybrids differ only in the way u_is are computed while computing partial signatures. In Hybrid_1, $u_i = H_1(\mathsf{hkey}_i, M)$, while in Hybrid_2, it is chosen uniformly randomly and then H_1 is programmed accordingly. Since H_1 is modeled as a random oracle the two hybrids are indistinguishable in the adversary's view. ■

Claim 6.19. Hybrid_2 and Hybrid_3 are statistically indistinguishable.

Proof. The two hybrids differ only in the order in which $H(K, M)$ and $r_{i,M} = H(K, M)^T \mathbf{v}_i$ are computed in pre-corruption queries. In Hybrid_2, $H(K, M)$ is set first, and then $r_{i,M}$ is computed accordingly. In Hybrid_3, $r_{i,M}$ is fixed first, and then $H(K, M)$ is programmed after the adversary reveals the set S of corrupted parties such that $H(K, M)^T \mathbf{v}_i = r_{i,M}$ is satisfied for each $i \in S$. Next, we show that this change in the order of computation does not change the adversary's view.

Let P_a be the honest party. Then, observe that the adversary receives the following values: $H(K, M)$, $\{\mathsf{sk}_i\}_{i\in[N]\setminus\{a\}}$, $\{\mathbf{v}_i\}_{i=1}^N$ and $\{\sigma_{i,M}\}_{i=1}^N$ in the two hybrids. We show that their joint distribution in the two hybrids is indistinguishable.

Firstly, consider $\sigma_{i,M}^{(2)}$ and $\sigma_{i,M}^{(3)}$, where superscripts indicate the respective hybrids. Recall that $\sigma_{i,M}^{(2)} = s_{i,M}^{(2)} + e'_{i,M}$ and $\sigma_{i,M}^{(3)} = s_{i,M}^{(3)} + e'_{i,M}$ where the added noise is sampled from the same distribution in both the hybrids. Hence we focus on $s_{i,M}^{(2)}$ and $s_{i,M}^{(3)}$. $s_{i,M}^{(2)} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^\mathsf{T}\mathbf{v}_i$. Recall that $\{\mathsf{sk}_i\}_{i\in[N]}$ are random secret shares of $\mathsf{FHE.sk}$ and $\{\mathbf{v}_i\}_{i\in[N]}$ are random secret shares of $\mathbf{0}$. Hence, by linearity property of Share, $\{\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})\}_{i\in[N]}$ are secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$, $\{H(K, M)^\mathsf{T}\mathbf{v}_i\}_{i\in[N]}$ are secret shares of 0 and hence $\{\mathsf{s}_{i,M}^{(2)}\}_{i\in[N]}$ are secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$. Moreover, since H is modeled as a random oracle, we have that $\{H(K, M)^\mathsf{T}\mathbf{v}_i\}_{i\in[N]}$ is a random secret sharing of 0, due to which $s_{i,M}^{(2)}$ are a random secret sharing of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$.

On the other hand, $\{s_{i,M}^{(3)}\}_{i\in[N]}$ are also random secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$, by design. Hence, they have the same distribution.

Next, observe that the adversary has sk_i for $i \in [N] \setminus \{a\}$ and hence it can compute $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$ which is supposed to be equal to $H(K, M)^\mathsf{T}\mathbf{v}_i$ for $i \in [N] \setminus \{a\}$. Thus, given $\mathbf{v}_i$, sk_i, $s_{i,M}$ for $i \in [N] \setminus \{a\}$, $H(K, M)$ is a random vector from the set $\{\mathbf{b} \ : \ \mathbf{b}^\mathsf{T}\mathbf{v}_i = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \ \forall \ i \in [N] \setminus \{a\}\}$. Again, the same is true for $H(K, M)$ in Hybrid_3 by design. Hence, the joint distribution

of adversary's view in the two hybrids is indistinguishable.

Finally, since the adversary gets to know the secret K only after revealing the set S of corrupted parties, there is only negligible probability that the adversary makes a random oracle query for input (K, M) before revealing the set S (which could lead to inconsistent values for $H(K, M)$). Hence, setting $H(K, M)$ for pre-corruption queries in the two hybrids in the above described ways, does not change adversary's view. ■

Claim 6.20. Assume that the flooding error is of the order $\text{poly}(\lambda) \cdot \sqrt{Q}$. Then if there is an adversary that can win the unforgeability game in Hybrid_3 with probability ϵ, then its probability of winning the game in Hybrid_4 is at least $\epsilon^2/2$.

Proof. Let the adversary makes Q signing queries and let theP_a be the honest party. Then in the adversary's view the two hybrids differ only in the error term in $\sigma_{a,M}$, as shown below.

Let $e'_a \leftarrow \mathcal{D}_s$. In Hybrid_3, for pre-corruption queries, the partial signature $\sigma_{a,M}$ is computed as follows:

$$
\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sum_{i=1}^{N} s_{i,M} - \sum_{i\in[N]\setminus\{a\}} s_{i,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i\in[N]\setminus\{a\}} s_{i,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i\in[N]\setminus\{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K, M)^{\mathsf{T}}\mathbf{v}_i) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K, M)^T\mathbf{v}_i + H(K, M)^{\mathsf{T}}\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M})
\end{aligned}
$$

$$
\begin{aligned}
&+\mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^T \mathbf{v}_i + H(K,M)^T \mathbf{v}_a + e'_a \\
= \;& \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \mathbf{v}_a + e'_a \\
& \qquad (\because \sum_{i=1}^{N} \mathbf{v}_i = 0; \;\; \sum_{i=1}^{N} \mathsf{sk}_i = \mathsf{FHE.sk})
\end{aligned}
$$

In Hybrid_3, for any post-corruption signing query on message M, the partial signature $\sigma_{a,M}$ is computed as:

$$\mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^{\mathsf{T}} \mathbf{v}_a + e'_a$$

In Hybrid_4, for pre-corruption queries, we have

$$
\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in [N] \setminus \{a\}} s_{i,M} + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in [N] \setminus \{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \mathbf{v}_i) + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^T \mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \mathbf{v}_a + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - H(K,M)^T \sum_{i=1}^{N} \mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \mathbf{v}_a + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^T \mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma}) + H(K,M)^T \mathbf{v}_a + e + e'_a
\end{aligned}
$$

In Hybrid_4, for any post-corruption query for a message M, we have

$$
\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sigma_M . \lfloor q/2 \rceil - \sum_{i \in [N] \setminus \{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \mathbf{v}_i) + e'_a
\end{aligned}
$$

$$
\begin{aligned}
&= \sigma_M.\lfloor q/2 \rceil - \sum_{i=1}^{N}(\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_i) \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i=1}^{N}\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\sum_{i=1}^{N}\mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + (e'_a + e).
\end{aligned}
$$

Thus, the difference in the two hybrids is in the error terms in $\sigma_{a,M}$. In Hybrid_3, the error is e'_a, while in Hybrid_4, it is $e'_a + e$. This is the same case as in Claim 6.7 in Section 6.5. Hence we can use exactly the same analysis using Rényi Divergence as in the proof of Claim 6.7, to complete the proof. ■

Claim 6.21. Assuming the privacy property of secret sharing scheme Share, Hybrid_4 and Hybrid_5 are indistinguishable.

Proof. The only difference between Hybrid_4 and Hybrid_5 is in the way the key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ are generated. In Hybrid_4, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$, while in Hybrid_5, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathbf{0})$. Since the adversary is given the key shares only for an invalid set of parties, the two distributions are identical due to the privacy property of secret sharing scheme Share. ■

Claim 6.22. Assume that FHE is semantically secure. Then Hybrid_5 and Hybrid_6 are computationally indistinguishable.

Proof. The proof is via standard reduction to FHE security and is similar to the proof of Claim 6.9. ■

Finally the proof for Theorem 6.16 completes with the following claim.

Claim 6.23. If the underlying signature scheme Sig is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 6.6 is negligible in Hybrid_6.

Proof. The proof is via standard reduction to Sig security and is similar to the proof of Claim 6.10. ■

6.7.3 Robustness

It can be seen that if HS is a multi data secure homomorphic signature, then the construction of pTS satisfies robustness. The proof is the same as that for Claim 6.11.

6.8 FULLY ADAPTIVE UNFORGEABILITY IN THE PREPROCESSING MODEL

In this section we provide our construction for fully adaptive threshold signatures in the standard model but with pre-processing, where signers must be provided correlated randomness of length proportional to the number of signing queries. We emphasize that this correlated randomness is independent of messages, and that this processing can be done in an offline phase before any messages are made available. The informed reader may notice similarities with the "MPC with Preprocessing" model (please see [FKOS15] and references therein).

6.8.1 Construction

The construction in standard model differs from the one in ROM in the way the random values $r_{i,j}$ are chosen. In this construction, $r_{i,j}$ is sampled directly for all possible signing query j in such a way that for each j, $\sum_{i=1}^{N} r_{i,j} = 0$. This helps to achieve full adaptivity because when key shares of one or more parties in $S' \subseteq [N]$ are revealed to the adversary, it does not fix $r_{i,j}$ values for $i \in [N] \setminus S'$. This gives the challenger the flexibility to simulate partial signature for uncorrupted parties and adjust their randomness $r_{i,j}$ later.

Let Q be the maximum number of signing queries. For a stateless scheme, we use a collision resistant hash function $H\{0,1\}^* \rightarrow [Q]$, which maps a message to an index in $[Q]$. We also use $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^r$ modelled as random oracle as also used in the partially adaptive construction. We denote our scheme for full adaptivity by $\mathsf{fTS} = (\mathsf{fTS.KeyGen}, \mathsf{fTS.PartSign}, \mathsf{fTS.PartSignVerify}, \mathsf{fTS.Combine}, \mathsf{fTS.Verify})$.

Construction

$\mathsf{fTS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following:

1. For $j = 1$ to Q, generate random values $\{r_{ij}\}_{i=1}^N \leftarrow \mathsf{Share}(0)$, such that $\sum_{i=1}^N r_{ij} = 0$.
2. Generate $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and N shares of $\mathsf{FHE.sk}$ as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{FHE.sk}$.
3. Compute an FHE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
4. For each P_i, sample a tag $\tau_i \leftarrow \{0,1\}^*$ and a hash key $\mathsf{hkey}_i \leftarrow \{0,1\}^\lambda$. Generate HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$, and the public and the signing keys as $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here, n is the bit length of input to PartSign circuit which depends on $(\mathsf{FHE.sk}, r_{ij}, \mathsf{hkey}_i)$.
5. Compute $(\pi_{\tau_i}, \pi_i) = \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i), \tau_i)$ for each $i \in [N]$.
6. Output $\mathsf{TSig.pp} = (\mathsf{FHE.pk}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N)$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{fTS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, a partial signing key $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i, \pi_i)$ and a message M, do the following:

1. Compute $j = H(M)$, $u_i = H_1(\mathsf{hkey}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u_i)$.
2. Let C_M be the $\mathsf{Sig.Sign}$ circuit with message M being hardcoded. Compute

FHE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} \leftarrow \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.

3. Compute $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e'_i$.

4. This step computes a homomorphic signature $\tilde{\pi}_{i,M}$ on partial signature $\sigma_{i,M}$ to provide robustness (Definition 6.8).
Let C_{PS} be a circuit with CT_{σ_M} being hardcoded and which takes as input the key share $\mathsf{TSig.sk}_i$ to compute $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e'_i$.

   - Compute $\pi^*_{i,M} = \mathsf{HS.SignEval}(\mathsf{HS.pp}, C_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, r_{ij}, \mathsf{hkey}_i), \pi_i)$.
   - Compute $\tilde{\pi}_{i,M} = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_{i,M}, \pi^*_{i,M})$.

5. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

$\mathsf{fTS.PartSignVerify}(\mathsf{TSig.pp}, M, y_{i,M})$: Upon input the public parameters TSig.pp, message M, and a partial signature $y_{i,M}$, the verifier does the following.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.

2. Defines circuit C_{PS} as described before and computes $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$.

3. Parses $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and outputs $\mathsf{HS.HVerify}(\mathsf{HS.pp}, \alpha, \sigma_{i,M}, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_{i,M}))$.

$\mathsf{fTS.Combine}(\mathsf{TSig.pp}, \{y_{i,M}\}_{i\in[N]})$: Upon input the public parameters TSig.pp and a set of partial signatures $\{y_{i,M}\}_{i\in[N]}$, parse $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and output $\sigma_M = \mathsf{FHE.decode}_1(\sum_{i=1}^{N} \sigma_{i,M})$.

$\mathsf{fTS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input a verification key TSig.vk, a message M and signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

Correctness

From the correctness of FHE.Eval, $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ is the encryption of $C_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the FHE

secret key FHE.sk. So, $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the fTS.Combine algorithm is

$$\begin{aligned}
\mathsf{FHE.decode}_1(\sum_{i=1}^{N} \sigma_{i,M}) &= \mathsf{FHE.decode}_1(\sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^{N} r_{ij} + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + 0 + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{FHE.decode}_1(\sigma_M \lfloor q/2 \rceil + e + \sum_{i=1}^{N} e'_i) = \sigma_M .
\end{aligned}$$

6.8.2 Unforgeability

Theorem 6.24. *Assume the signature scheme* Sig *satisfies unforgeability,* FHE *is a CPA secure fully homomorphic encryption scheme (Definition 6.12),* HS *is context hiding homomorphic signature scheme (Definition 6.21) and* Share *satisfies privacy (Definition 6.28). Then the above construction of* fTS *satisfies adaptive unforgeability (Definition 6.5) if the flooding error is of the size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, where* Q *is the number of signing queries.*

Proof. The security of the construction can be argued using the following hybrids:

Hybrid_0: The real world.

Hybrid_1 : Same as Hybrid_0, except that now instead of using HS.SigEval algorithm to compute the homomorphic signature $\tilde{\pi}_{i,M}$ on $\sigma_{i,M}$, the challenger simulates $\tilde{\pi}_{i,M}$ as $\tilde{\pi}_{i,M} = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_{i,M}, \tau_i, \pi_{\tau_i})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, C_{\mathsf{PS}})$.

Hybrid_2: Same as Hybrid_1, except that now the randomness u_i used in sampling flooding noise in PartSign algorithm is chosen uniformly randomly from $\{0,1\}^r$ and then H_1 is programmed as $H_1(\mathsf{hkey}_i, M) = u_i$. For random oracle queries by the

adversary on an input x, the challenger first checks if $H_1(x)$ is already set. If so, then returns it else chooses a value uniformly randomly from $\{0,1\}^r$ and saves and returns it.

Hybrid_3: Same as Hybrid_2 except that now the r_{ij} values are set in a different order. In particular, for each $i \in [N]$, let $PreQ_i$ be the set of messages for which partial signatures are computed before corrupting P_i. Then, for each $j \in \{H(M) : M \in PreQ_i\}$, r_{ij} is set in reverse order, i.e the challenger first computes the partial signature $\sigma_{i,M}$ and then sets the value for $r_{i,H(M)}$ as defined below. For any signing query on message M, let $S_M \subseteq [N]$ be the set of parties corrupted by the adversary so far. Then to compute $\sigma_{i,M}$, the challenger does the following.

1. If M was queried before then returns $\sigma'_{iM} + e'_i$ for each $i \in [N] \setminus S_M$, where $\sigma'_{i,M}$ is the same value as in the earlier response, but e'_i is sampled afresh.
2. Else, computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{Sig.sk})$ and does the following:
   - For each $i \in S_M$, computes $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,H(M)}$ and sets $\sigma_{i,M} = \sigma'_{i,M} + e'_i$.
   - For $i \in [N] \setminus S_M$,
     - Divide $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{k \in S_M} \sigma'_{k,M}$ into $N - |S_M|$ random shares, $\{s_{i,M}\}_{i \in [N] \setminus S_M}$.
     - Set $\sigma'_{i,M} = s_{i,M}$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$.
3. Return $\{\sigma_{i,M}\}_{i \in [N] \setminus S_M}$ to the adversary.

When the adversary makes a key query for some $i \in [N]$, the challenger does the following.

1. For each $M \in PreQ_i$, computes $r_{i,H(M)} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$.
2. For $j \in [Q] \setminus \{H(M) : M \in PreQ_i\}$ chooses $r_{i,j}$ randomly.

Hybrid_4: Same as Hybrid_3, except the following changes: To answer signing query on any message M, for each $i \in S_M$, the challenger computes the signatures as in Hybrid_3, but for $i \in [N] \setminus S_M$ the challenger simulates the signatures as follows: instead of sharing $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} \sigma'_{i,M}$, the challenger now shares $\sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} \sigma'_{i,M}$ between the uncorrupted parties as described below.

1. Compute $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
2. For each $i \in S_M$, compute $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,H(M)}$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.
3. For uncorrupted parties do the following: divide $\sigma_M \lfloor q/2 \rceil - \sum_{k \in S_M} \sigma'_{k,M}$ into $N - |S_M|$ random shares, $\{s_{i,M}\}_{i \in [N] \setminus S_M}$. Set $\sigma_{i,M} = s_{i,M} + e'_i$ for $i \in [N] \setminus S_M$.
4. Return $\sigma_{i,M}$ for $i \in [N] \setminus S_M$.

Key queries are answered in the same way as in the previous hybrid.

Hybrid_5: Same as Hybrid_4, except that now the challenger shares zero vector as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathbf{0})$ instead of $\mathsf{FHE.sk}$ to generate the key share sk_i in $\mathsf{TSig.sk}_i$.

Hybrid_6: Same as Hybrid_5, except that now $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Proof for indistinguishability between Hybrid_0, Hybrid_1 and Hybrid_2 is the same as that in the proof of Theorem 6.16 in Section 6.7.

Claim 6.25. Hybrid_2 and Hybrid_3 are statistically indistinguishable

Proof. Observe that the two hybrids differ only in the way the $\{r_{i,j}\}$ shares are set. For any message M, in Hybrid_2, $\{r_{i,j}\}_{i \in [N]} \leftarrow \mathsf{Share}(0)$, where $j = H(M)$. In Hybrid_3,

$\{r_{i,j}\}_{i\in S_M}$ are randomly chosen and for $i \in [N] \setminus S_M$, $r_{i,j} = s_{i,j} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$, where $\{s_{i,j}\}_{i\in[N]\setminus S_M}$ are obtained by randomly sharing $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{k\in S_M}(\mathsf{FHE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + r_{k,j})$ into $N - |S_M|$ shares. Thus,

$$\begin{aligned}
\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) &= \sum_{k\in S_M} \mathsf{FHE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + \sum_{k\in S_M} r_{k,j} + \sum_{i\in[N]\setminus S_M} s_{i,j} \\
&= \sum_{k\in S_M} \mathsf{FHE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + \sum_{k\in S_M} r_{k,j} \\
&+ \sum_{i\in[N]\setminus S_M} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i\in[N]\setminus S_M} r_{i,j} \\
&= \sum_{i\in[N]} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i\in[N]} r_{i,j} \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \sum_{i\in[N]} r_{i,j}
\end{aligned}$$

This implies $\sum_{i\in[N]} r_{i,j} = 0$, and since $\{s_{i,j}\}_{i\in[N]\setminus S_M}$ are random shares, we can conclude that $\{r_{i,j}\}_{i\in[N]}$ are random shares of 0, which is same as Hybrid_2. ■

Claim 6.26. Assume that the flooding error is of the order $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$. If there is an adversary who can win the unforgeability game as per Definition 6.5 in Hybrid_3 with probability ϵ, then its probability of winning the game in Hybrid_4 is at least $\epsilon^2/2$.

Proof. Let the adversary issues Q signing queries. Let P_a be the honest party for some $a \in [N]$. Then the two hybrids differ only in the error term in $\sigma_{a,M}$, as shown below.

Consider any signing query for a message M. Let S_M be the set of corrupted parties so far and let $H(M) = j$ and $e'_a \leftarrow \mathcal{D}_s$. Then,

In Hybrid_4, we have:

$$\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i\in S_M} \sigma'_{i,M} - \sum_{\substack{i\in[N]\setminus S_M \\ i\neq a}} s_{i,M} + e'_a
\end{aligned}$$

$$
\begin{aligned}
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) \\
&\quad - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq a}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in [N]} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{a\}} r_{ij} \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i \in [N]} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{a\}} r_{ij} \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N]} r_{ij} + r_{a,j} \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + r_{aj} + e + e'_a
\end{aligned}
$$

On the other hand in Hybrid_3,

$$
\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} \sigma'_{i,M} - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq a}} s_{i,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) \\
&\quad - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq a}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N]} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad - \sum_{i \in [N] \setminus \{a\}} r_{ij} + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \mathsf{FHE.decode}_0(\sum_{i \in [N]} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad - \sum_{i \in [N] \setminus \{a\}} r_{ij} + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + e'_a - \sum_{i \in [N] \setminus \{1\}} r_{ij} \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + r_{aj} + e'_a
\end{aligned}
$$

In the third step, $\sigma'_{i,M}$ is computed as $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$, while $s_{i,M}$ is replaced with $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$ because of the way $r_{i,j}$ is set. In the last step we replace $-\sum_{i\in[N]\setminus\{a\}} r_{ij}$ by r_{aj} because $\sum_{i\in[N]} r_{ij} = 0$. However note that r_{aj} is never actually set since $\mathsf{TSig.sk}_a$ is never queried for.

Thus, the difference in the two hybrids is in the error terms in σ_a. In Hybrid_3, the error is e'_a, while in Hybrid_4, it is $e'_a + e$. This is the same case as in Claim 6.7 in Section 6.5. Hence we can use exactly the same analysis using Rényi Divergence as in the proof of Claim 6.7, to complete the proof. ■

Indistinguishability between $\mathsf{Hybrid}_4, \mathsf{Hybrid}_5$ and Hybrid_6 has the same argument as that for indistinguishability between $\mathsf{Hybrid}_4, \mathsf{Hybrid}_5$ and Hybrid_6 in the proof of Theorem 6.16.

Finally the proof of Theorem 6.24 completes with the following claim.

Claim 6.27. If the underlying signature scheme Sig is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 6.5 is negligible in Hybrid_6.

Proof. The proof is via standard reduction to Sig security and is similar to the proof of Claim 6.10. ■

6.8.3 Robustness

Claim 6.28. If HS is multi data secure homomorphic signature, then the above construction of fTS satisfies robustness.

Proof. The proof is same as the proof for Claim 6.11. ■

6.9 THRESHOLD SIGNATURES FOR t-OUT-OF-N ACCESS STRUCTURES

In this section we give a general construction for $t\text{-}out\text{-}of\text{-}N$ access structure using $\{0,1\}$-LSSS.

6.9.1 Construction

The construction uses following building blocks:

1. A special fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$. Let B be the error bound of the FHE scheme.
2. A UF-CMA signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$.
3. A t out of N $\{0,1\}$-LSSS, Share.

We denote our scheme for threshold signatures with partial adaptivity by $\mathsf{tTS} = (\mathsf{pTS.KeyGen}, \mathsf{tTS.PartSign}, \mathsf{tTS.Combine}, \mathsf{tTS.Verify})$. To keep the construction simple, we have omitted the steps required for robustness. The robustness can be achieved using homomorphic signatures in the same way as in the previous constructions.

Construction

$\mathsf{tTS.KeyGen}(1^\lambda, t)$: Upon input the security parameter λ and the threshold t do the following:

1. Generate the verification and signing keys for the signature scheme $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
2. Generate the keys for the FHE scheme $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and compute $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
3. Share the FHE secret key as: $\{\mathsf{TSig.sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{FHE.sk}, t)$. Note that for $\{0,1\}$-LSSS, each $\mathsf{TSig.sk}_i$ can be a set of more than one secret shares. *Notation*: Let $\mathbf{M}$ be the share matrix (Def. 6.29) of dimension $\ell \times N$. Then for $i \in [N]$, T_i refers to the partition of $[\ell]$ corresponding to party P_i and $\mathsf{TSig.sk}_i = \{\mathsf{sk}_j\}_{j \in T_i}$, where sk_j, is the jth (out of ℓ shares) share of FHE.sk.
4. Output $\mathsf{TSig.pp} = \{\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i\}_{i=1}^N$.

$\mathsf{tTS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, the partial signing key $\mathsf{TSig.sk}_i$ and a message M, do the following:

1. Let C_M be the signing circuit, with message M being hardcoded. Compute $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Output $\sigma_i = \{\hat{\sigma}_j\}_{j \in T_i}$, where $\hat{\sigma}_j = \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j$, where, $e'_j \leftarrow \mathcal{D}_s$.

$\mathsf{tTS.Combine}(\mathsf{TSig.pp}, \{\sigma_i\}_{i \in S})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{\sigma_i\}_{i \in S}$, where $\sigma_i = \{\hat{\sigma}_j\}_{j \in T_i}$ and $S \subseteq [N]$, the $\mathsf{Combine}$ algorithm first checks if $|S| \geq t$. If not, then outputs $\perp$, else computes a minimum valid share set (Def. 6.30) $T \subseteq \bigcup_{i \in S} T_i$ and outputs

$$\sigma_M = \mathsf{FHE.decode}_1(\sum_{j \in T} \hat{\sigma}_j).$$

$\mathsf{tTS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input the verification key $\mathsf{TSig.vk}$, a message M and signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

Correctness

From the correctness of $\mathsf{FHE.Eval}$ algorithm, $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, C_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ is the encryption of $C_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the FHE secret key $\mathsf{FHE.sk}$. So, $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) = \sigma_M \lfloor q/2 \rceil + e$, where e is the error in CT_σ. The signature computed by the $\mathsf{tTS.Combine}$ algorithm is

$$\begin{aligned}
\mathsf{FHE.decode}_1(\sum_{j \in T} \hat{\sigma}_j) &= \mathsf{FHE.decode}_1(\sum_{j \in T} \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + \sum_{j \in T} e'_j) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\sum_{j \in T} \mathsf{sk}_j, \mathsf{CT}_\sigma) + \sum_{j \in T} e'_j) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) + \sum_{j \in T} e'_j)
\end{aligned}$$

(from the correctness of Share algorithm.)

$$= \mathsf{FHE.decode}_1(\sigma_M \lfloor q/2 \rceil + e + \sum_{j \in T} e'_j) = \sigma_M.$$

6.9.2 Unforgeability

Theorem 6.29. *Assume* FHE *is a secure fully homomorphic encryption as per Definition 6.12,* Share *is a* $\{0,1\}$-LSSS *t-out-of-N secret sharing scheme that satisfies Definition 6.28 and* Sig *is a signature scheme that satisfies (UF-CMA) unforgeability. Then the above construction of t-out-of-N threshold signature satisfies selective unforgeability (Definition 6.7).*

Proof. We prove the theorem using following hybrids.

Hybrid_0 : Same as the real world.

Hybrid_1: Same as Hybrid_0 except that now the signing queries are answered differently.

1. Upon receiving the (invalid) party set S^* from $\mathcal{A}$, the challenger commits to a maximal invalid share set T^* which contains $\bigcup_{i \in S^*} T_i$.

2. To answer any partial signing query (M, i) for message M and party P_i ($i \in [N] \setminus S^*$), the challenger computes $\hat{\sigma}_j$ for $j \in T_i$ as follows:

   - If $j \in T_i \cap T^*$, then computes $\hat{\sigma}_j = \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j$, i.e. as in the real world.

   - If $j \notin T_i \cap T^*$, then the challenger does the following: computes a minimal valid share set $T \subseteq T^* \cup \{j\}$ (Note that such a set always exists and contains j because T^* is a maximal invalid share set and $j \notin T^*$, hence $T^* \cup \{j\}$ is a valid share set.) and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$. Then it computes $\hat{\sigma}_j$ as

$$\hat{\sigma}_j = \lfloor q/2 \rceil \cdot \sigma_M - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + e'_j.$$

Hybrid_2: Same as Hybrid_1, except that instead of sharing FHE.sk the challenger now

shares $\mathbf{0}$ to compute key shares as $(\mathsf{TSig.sk}_1, \ldots, \mathsf{TSig.sk}_N) \leftarrow \mathsf{Share}(\mathbf{0}, t)$.

Hybrid_3: Same as Hybrid_2, except that $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Claim 6.30. If the flooding error is of the size $\mathrm{poly}(\lambda)\sqrt{Q}$, then if there is an adversary who can win the unforgeability game in Hybrid_0 with probability ϵ, then its probability of winning the game in Hybrid_1 is at least $\epsilon^2/2$.

Proof. Let the number of signing queries that an adversary can make be bounded by Q.

The two hybrids differ only in the error term in partial signatures returned by the challenger. Let the adversary issues partial signing query for (M, i). Let T_i, S^* and T^* be as defined in Hybrid_1. Then for $j \in T_i \cap T^*$, $\hat{\sigma}_j$ is computed in the same way in both the hybrids. The difference is in the error term in $\hat{\sigma}_j$ for $j \in T_i \setminus T^*$.

Let us focus on one such j. Let $e'_j \leftarrow \mathcal{D}_s$ and T be a minimal valid share set contained in $T^* \cup \{j\}$.

In Hybrid_0, we have:

$$\hat{\sigma}_j = \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j.$$

In Hybrid_1, we have:

$$\begin{aligned}
\hat{\sigma}_j &= \sigma_M . \lfloor q/2 \rceil - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_M . \lfloor q/2 \rceil - \sum_{j' \in T} \mathsf{FHE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_M . \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{j' \in T} \mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j
\end{aligned}$$

$$
\begin{aligned}
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_M.\lfloor q/2 \rceil - \sigma_M.\lfloor q/2 \rceil + e + \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + (e'_j + e)
\end{aligned}
$$

Thus, the difference in the two hybrids is in the error terms in $\hat{\sigma}_j$ for $j \in T_i \setminus T^*$. In Hybrid_0, the error is e'_j, while in Hybrid_1, it is $e'_j + e$. The proof uses Rényi Divergence and is similar to the proof given for Claim 6.7. We refer to the distribution to be considered in Hybrid_1 and Hybrid_2 by D_1 and D_2, respectively. Let E_k be the random variable for error vector corresponding to the error terms in ciphertexts returned in response to the k-th query. Let Y_k be the random variable for k-th query which is of the form (M_k, i_k). Then as we discussed in proof of Claim 6.7, $D_t = (\mathcal{E}_Q^{(t)}, \mathcal{M}_Q^{(t)}, \mathcal{E}_{Q-1}^{(t)}, \mathcal{M}_{Q-1}^{(t)}, \ldots, \mathcal{E}_1^{(t)}, \mathcal{M}_1^{(t)})$ for $t \in \{1, 2\}$, where $\mathcal{E}_k^{(t)}$ is the distribution of E_k in Hybrid_t and $\mathcal{M}_k^{(t)}$ is the distribution of Y_k in Hybrid_t. From the above analysis, we have that for any given query (M_k, i_k), $\mathcal{E}_k^{(1)} = \mathcal{D}_{\mathbb{Z}^{n_k}, s, \mathbf{0}}$ and $\mathcal{E}_k^{(2)} = \mathcal{D}_{\mathbb{Z}^{n_k}, s, \mathbf{e}_k}$, where $\mathbf{e}_k = (e_k, \ldots, e_k)$ is a vector of length $n_k = |T_{i_k} \setminus T^*|$ and $|e_k| \leq B_{eval}$ and $n_k \leq \ell$, where ℓ is the number of rows in the share matrix (see Definition 6.29) and is bounded by $\mathrm{poly}(N)$. Then,

$$
R_a(D_1 \| D_2) = R_a(\mathcal{E}_Q^{(1)}, \mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{E}_Q^{(2)}, \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}). \quad (6.9)
$$

Applying the multiplicativity property of the Rényi divergence (Lemma 6.1), we obtain that $R_a(D_1 \| D_1)$ is bounded from above by

$$
\begin{aligned}
&\max_{x \in X} R_a(\mathcal{E}_Q^{(1)} | X = x \parallel \mathcal{E}_Q^{(2)} | X = x) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}) \\
= &\max_{x \in X} R_a(\mathcal{D}_{\mathbb{Z}^{n_Q}, s, \mathbf{0}} | X = x \parallel \mathcal{D}_{\mathbb{Z}^{n_Q}, s, \mathbf{e}_Q} | X = x) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}),
\end{aligned} \quad (6.10)
$$

where $X = (Y_Q, E_{Q-1}, \ldots, E_1)$.

Then applying Lemma 6.2 in Equation (6.10), we get

$$
R_a(D_1 \| D_2) \leq \max_{x \in X} \exp(a\pi \|\mathbf{e}_Q\|^2 / s^2) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)})
$$

$$
\begin{aligned}
&= \max_{x \in X} \exp(a\pi n_Q e_Q^2/s^2) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}) \\
&\leq \exp(a\pi\ell B_{eval}^2/s^2) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}).
\end{aligned}
$$

Further, from the data processing inequality (Lemma 6.1),

$$
\begin{aligned}
&R_a(\mathcal{M}_Q^{(1)}, \mathcal{E}_{Q-1}^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}) \\
&\qquad \leq R_a(\mathcal{E}_{Q-1}^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}),
\end{aligned}
$$

Hence, we get

$$
\begin{aligned}
R_a(D_1 \| D_2) &\leq \exp(a\pi\ell B_{eval}^2/s^2) \cdot R_a(\mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\
&\leq \exp(\frac{a\pi Q\ell B_{eval}^2}{s^2}),
\end{aligned}
$$

where the second inequality follows from induction.

Setting $s = B_{eval} \cdot \sqrt{\ell \cdot Q\lambda}$, we get

$$
R_a(D_0 \| D_1) \leq \exp\left(\frac{a\pi}{\lambda}\right)
$$

Therefore,

$$
\begin{aligned}
D_1(E) &\geq \frac{D_0(E)^{\frac{a}{a-1}}}{R_a(D_0 \| D_1)} \\
&\geq D_0(E)^{\frac{a}{a-1}} \exp\left(-\frac{a\pi}{\lambda}\right)
\end{aligned}
$$

The claim is proved by taking $a = 2$. Thus, if the probability of success in Hybrid_0 is non-negligible then it is non-negligible in Hybrid_1 as well. ■

Claim 6.31. Assume that Share is secure sharing scheme, then Hybrid_1 and Hybrid_2 are indistinguishable.

Proof. The two hybrids differ only in the generation of key shares. In Hybrid_1, $\{\mathsf{TSig.sk}_i\}_{i \in [N]} = \mathsf{Share}(\mathsf{FHE.sk}, t)$, while in Hybrid_2 $\{\mathsf{TSig.sk}_i\}_{i \in [N]}$ is computed as $\mathsf{Share}(\mathbf{0}, t)$. Hence by the privacy property (Definition 6.28) of Share the two hybrids

are identical in the adversary's view since it receives the key shares only for an invalid set of participants. ■

Claim 6.32. Assume the FHE scheme is secure (Definition 6.12). Then Hybrid_2 and Hybrid_3 are indistinguishable.

Proof. The proof is similar to the proof of Claim 6.9. ■

Finally, the proof of Theorem 6.29 completes with the following claim. The proof of the claim is similar to the proof of Claim 6.10 and is omitted.

Claim 6.33. If the underlying signature scheme Sig is unforgeable, then the the adversary cannot win the unforgeability game in Hybrid_3 with non-negligible advantage.

■

6.9.3 Robustness

To add robustness in the above scheme, we can use context hiding secure homomorphic signature in the same way as in Section 6.5. In particular, the KeyGen algorithm also includes in $\mathsf{TSig.sk}_i$ a HS signature of party P_i's key shares. To prove the honest evaluation of PartSign algorithm, the signer homomorphically computes a signature on $\sigma_{i,M}$ and gives it to the verifier. The unforgeability property of HS provides robustness and its context hiding property ensures that unforgeability of TS is maintained.

6.9.4 Construction for adaptive unforgeability

The above construction of t-out-of-N threshold signatures can be made partially adaptive unforgeable in the same way as in Section 6.7. In particular, for t-out-of-N access structure, the random shares of $\mathbf{0}$, i.e. vector $\mathbf{v}_i$ in $\mathsf{TSig.sk}_i$ for $i \in [N]$ are now computed as $\{\mathbf{v}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathbf{0}, t)$. Similarly for fully adaptive unforgeablity the construction can be modified in the same way as in Section 6.8. In particular, for t-out-of-N adaptation, the random shares of 0 in $\mathsf{TSig.sk}_i$ are now generated as: $\forall\ j \in [Q], \{r_{i,j}\}_{i\in[N]} \leftarrow \mathsf{Share}(0, t)$.

CHAPTER 7

PRACTICAL, ROUND-OPTIMAL LATTICE-BASED BLIND SIGNATURES

7.1 INTRODUCTION

Blind signatures are a fundamental cryptographic primitive with numerous applications in e-cash [Cha82], e-voting [IKSA03] cryptocurrencies [YL19] and many others. In a blind signature scheme [Cha82], a user $\mathcal{U}$, holding a public key and message, may request a signature from a signer $\mathcal{S}$, holding a signing key, such that the signer is not able to link a message-signature pair with a protocol execution, and the user is not able to forge signatures even after multiple interactions with the signer.

In this work, we provide a lattice-based blind signature that is overall practical and supports an unbounded number of signature queries and additionally enjoys optimal round complexity.

Prior Work. Blind signatures have been studied for several decades, and admit instantiations from a variety of assumptions [CP92; PS00; FPS20; KLX22; GG14; GRS$^+$11; Fis06; LNP22a]. Given their wide applicability, there has been a significant thrust towards obtaining practical efficiency. Constructions based on standard assumptions are primarily feasibility results [GRS$^+$11; Fis06] which do not admit practical instantiations. In light of this, in the number-theoretic regime, reasonable new assumptions were introduced to obtain efficient constructions. For instance, in the group setting, several candidates [CP92; Oka92; PS00; HKL19; FPS20] are based on the hardness of the non-standard ROS/mROS problem (note that the ROS problem was recently broken [BLL$^+$21]) or rely on the algebraic group and the generic group models [Abe01; TZ22; KLX22], which are

very strong idealizations. The situation is analogous in the regime of pairings [BLS01; Bol03; GG14] or RSA [BNPS03].

Post-Quantum Regime. Under post-quantum assumptions, the situation is much more unsatisfactory – even disregarding efficiency, several lattice-based blind signatures [Rüc10; ABB20b; ABB20a; BECE+20; LSK+19; PHVBS19] were found to have errors in their security proofs [HKLN20]. The recent construction by Hauck *et al.* [HKLN20] aimed to fix the errors but the resulting construction is completely impractical – using their suggested parameters, the constructed blind signature has size ≈ 7.73MB, for security against adversaries limited to getting 7 signatures. The very recent work of Lyubashevsky *et al.* [LNP22a] achieves better parameters (signature size of about 150KB), but the cost of their signing algorithm grows linearly in the maximum number of signatures that an adversary can query. This makes it impractical for situations where the number of signatures is large or cannot be apriori bounded. Finally, there are constructions based on codes [BGSS17] and systems of algebraic equations [PSM17] but these are either impractical or do not satisfy the standard definition of security.

Other Related Work. Subsequent to the public appearance of the present work, del Pino and Katsumata [dPK22] also provided a two round lattice-based blind signature. Their techniques and final result are incomparable to ours – on one hand, their signature is of size 102.6KB, which is more than twice as large as ours, and their transcript size is 851KB, which is about 18 times as large as ours. On the other hand, their construction relies on the hardness of the standard MSIS and MLWE assumptions, while ours relies on a new hardness assumption called the one-more-ISIS assumption (described below). Additionally, they show how to upgrade their construction to be secure in the quantum random oracle model (albeit at the cost of making the transcript size 770 times larger than ours) while we do not consider this extension in the present work.

7.2 OUR RESULTS.

In this work, we provide the first overall practical, lattice-based blind signature, which additionally enjoys optimal round complexity. Our scheme relies on the Gentry, Peikert and Vaikuntanathan (GPV) signature [GPV08] and non-interactive zero-knowledge proofs for linear relations with small unknowns, which are significantly more efficient than their general purpose counterparts. Its security stems from a new and arguably natural assumption that we introduce, called one-more-ISIS. This assumption can be seen as a lattice analogue of the one-more-RSA assumption by Bellare *et al.* [JoC'03]. Informally, the one-more-ISIS assumption states that for any polynomially bounded ℓ, it is difficult to forge $\ell + 1$ GPV signatures [GPV08], even when given access to up to ℓ inversions of arbitrarily chosen syndromes.

Our construction supports an unbounded number of signatures and is overall more efficient than all prior candidates. While it is based on a new assumption, we believe that for a practice oriented primitive like blind signatures, it is justified to introduce plausible assumptions as was done in the number-theoretic regime. We provide detailed cryptanalysis to justify our new assumption.

7.3 OUR TECHNIQUES

The starting point of our work is a two round blind signature by Fischlin [Fis06], which relies on the CRS model. To begin, we adapt this scheme to the ROM and instantiate it with efficient lattice based signatures and non-interactive zero knowledge proofs (NIZK). Due to the extensive research in efficient lattice based signatures [GPV08; Lyu12; ESS+19; GLP12; FHK+; BG14; DKL+18] and proof systems [LNSW13; DRS18; BCR+19; YAZ+19; BLS19; ESS+19; ENS20; LNS21] over the last 15 years, this already provides a candidate which is "somewhat reasonable" in practice.

Adapting Fischlin's Protocol. Our adaptation of Fischlin's protocol uses a public key encryption scheme PKE and a non-interactive zero knowledge argument of knowledge $\mathsf{NIZKAoK}$ as building blocks. To begin, we consider the honest signer model for blindness, in which it is assumed that the signing and verification keys are generated honestly, though the signer can deviate arbitrarily from the signing protocol. This assumption will subsequently be removed. We summarize this protocol next. In what follows, we assume some familiarity with the signature scheme of Gentry, Peikert and Vaikuntanathan (GPV); please refer to [GPV08] for a refresher.

In the setup phase, we run $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and discard $\mathsf{PKE.sk}$. Next, following the GPV signature scheme, we sample a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{T_C} \in \mathbb{Z}^{m \times m}$ of it. We set the signing key of the blind signature as $\mathsf{BSig.sk} = \mathbf{T_C}$, and the verification key as $\mathsf{BSig.vk} = (\mathbf{C}, \mathsf{PKE.pk})$.

To sign the message g, the user $\mathcal{U}$ samples $\mathsf{PKE.Enc}$ randomness r and computes $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mu; r)$. It sends ct to the signer. Upon receiving ct, the signer $\mathcal{S}$ computes a GPV signature on ct and returns this to the user. In more detail, it computes $H(\mathsf{ct})$ and uses the trapdoor $\mathbf{T_C}$ to sample $\mathbf{y}$ such that $\mathbf{y}$ is short and $\mathbf{Cy} = H(\mathsf{ct})$ (modulo q). It sends $\mathbf{y}$ to the user. Here H is a hash function, modeled as a random oracle in the security proof.

Upon receiving $\mathbf{y}$, the user $\mathcal{U}$ verifies that $\mathbf{y}$ is small and that $\mathbf{Cy} = H(\mathsf{ct})$ and aborts if this fails. It generates a non-interactive zero-knowledge argument of knowledge ($\mathsf{NIZKAoK}$) π for following statement: Given $\mathsf{BSig.vk} = (\mathbf{C}, \mathsf{PKE.pk})$ and μ, there exists PKE randomness r and a vector $\mathbf{y}$ such that

$$\|\mathbf{y}\| \leq \beta \ \wedge \ \mathbf{Cy} = H(\mathsf{Enc}(\mathsf{PKE.pk}, \mu; r)).$$

In the above, β is some appropriate bound. Finally, the user outputs π as the signature. To verify the blind signature, the verifier checks that the proof π is valid. Thus, the

final signature in the blind signature protocol is a NIZKAoK that the user knows a GPV signature for an *encryption* of the message.

For full-fledged blindness, it suffices to ensure that PKE.pk has been honestly generated by the adversarial signer, without a corresponding decryption key. This can be achieved, for example, by choosing PKE such that PKE.pk is computationally indistinguishable from uniform, and then setting PKE.pk as the output of another hash function H' modeled as a random oracle, on an arbitrary public input.

Since the witness of the NIZKAoK includes the randomness r used to compute the ciphertext, and the ciphertext is inside a (complex) hash function, the statement that we require to prove becomes very complex and resorting to general purpose NIZKAoK seems unavoidable. Despite amazing recent advances in efficient general purpose NIZKAoK [BCR+19; AHIV17], the resulting parameters are formidable – as discussed in Section 7.5, we estimate a proof size of more than 100KB and prover time complexity of one hour or more. Even worse, the prover of the NIZKAoK is the user in the blind signature, who is generally expected to be computationally light. This leads to a blind signature with very large user time complexity, which is very dissatisfying, both in theory and practice.

An Efficient Construction from one-more-ISIS. We begin by observing that general purpose NIZKAoKs are the primary source of inefficiency in the above protocol, and "lightening" the usage of NIZKAoK would result in a significantly lighter overall protocol. Intuitively, some usage of NIZKs feels unavoidable if we want to stick to a two round protocol, but can we simplify the statement that is proved? Our main new idea is to leverage a new, arguably natural assumption, which we call one-more-ISIS so that the problematic general purpose NIZKAoK above may be replaced by an efficient lattice based NIZK for linear statements with small unknowns, for which practical constructions have been developed recently [ENS20; LNS21; LNP22b]. Armed with these ideas, we

provide a simple, overall efficient protocol as follows.

For setup, we run $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and discard $\mathsf{PKE.sk}$. Again, discarding $\mathsf{PKE.sk}$ can be achieved in the real world by setting $\mathsf{PKE.pk}$ as the output of a hash function on a public value (this requires ensuring that the distributions match). Next, we sample a matrix $\mathbf{C}$ together with trapdoor $\mathbf{T_C}$ as before. At this stage, we depart from the previous protocol – instead of encrypting the message g to achieve blindness, we will rely on a much simpler "one time pad" style blinding mechanism. For this, we sample another matrix $\mathbf{A}$ and set $\mathsf{BSig.sk} = \mathbf{T_C}$, $\mathsf{BSig.vk} = (\mathbf{C}, \mathbf{A}, \mathsf{PKE.pk})$. For full fledged blindness, we would also need to set $\mathbf{A}$ as the output of a random oracle, together with $\mathsf{PKE.pk}$ as discussed above.

For signing a message g, a user $\mathcal{U}$ samples a vector $\mathbf{x}$ from a suitable distribution such that $\mathbf{Ax}$ is indistinguishable from uniform. It computes $\mathbf{t} = \mathbf{Ax} + H(\mathsf{g})$ and sends $\mathbf{t}$ to the signer. Note that for a suitable choice of $\mathbf{x}$, the term $H(\mathsf{g})$ and hence g is hidden from the view of the signer. Upon receiving $\mathbf{t}$, signer $\mathcal{S}$ uses the trapdoor $\mathbf{T_C}$ to sample a short vector $\mathbf{y}$ such that $\mathbf{Cy} = \mathbf{t}$ (modulo q). It sends $\mathbf{y}$ to the user. Upon receiving $\mathbf{y}$, user $\mathcal{U}$ verifies that $\mathbf{y}$ is short and satisfies $\mathbf{Cy} = \mathbf{t}$. It samples $\mathsf{PKE.Enc}$ randomness r and computes

$$\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x} \| \mathbf{y}; r).$$

It generates a NIZK π for following statement: Given $\mathsf{BSig.vk} = (\mathbf{C}, \mathbf{A}, \mathsf{PKE.pk})$, ct and μ, there exist r and vectors $\mathbf{x}, \mathbf{y}$ such that

$$\begin{aligned} \|\mathbf{x}\| \leq \beta_1 \quad \wedge \quad \|\mathbf{y}\| \leq \beta_2 \quad \wedge \quad \mathbf{Cy} - \mathbf{Ax} = H(\mathsf{g}) \\ \wedge \quad \mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x} \| \mathbf{y}; r). \end{aligned}$$

In the above, β_1 and β_2 are appropriate parameters and H is the random oracle hash function. The signature is (π, ct), and verification consists in verifying the NIZK π as before.

Note that the above statement also involves the hash function H which is modeled as a random oracle in the security proof. But, crucially, the input μ to H is known, implying that $H(\mu)$ can be seen as a *public* quantity and this does not make the proof complex. By using Regev's encryption scheme [Reg09] (or variants of it), one can ensure that the statement to be proved is linear in the unknowns, which are themselves required to be small. As a result, we can circumvent the use of a general-purpose NIZKAoK and can instead rely on NIZK for linear relations with small unknowns [ENS20; LNS21; LNP22b]. This lets us reduce the signature size to 45.19KB, as against more than 100KB. More importantly, the cost of generating and verifying the proof becomes very small.

The astute reader may wonder why the witnesses $\mathbf{x}\|\mathbf{y}$ are being encrypted. In the unforgeability proof, this allows to circumvent rewinding when extracting GPV preimages from the output of the adversary. Rewinding would incur a loss that is exponential in the number of preimages that the attacker requested from the signer. Please see Section 7.6 for more details.

The resultant protocol is extremely simple and appears quite similar to the first protocol we presented, which in turn is a natural adaptation of Fischlin's protocol from 2006 [Fis06]. The reader may wonder whether replacing the ciphertext computed by the user in the first step by a one time pad is the only difference from the first scheme, and if so, why efficient lattice-based blind signatures have remained elusive for so long. The key new insight of our work is in formulating a meaningful new assumption that allows reducing security of this very natural construction to it. We describe our assumption next and discuss how it implies security of our candidate.

The one-more-ISIS Assumption. The one-more-ISIS$_{q,n,m,\sigma,\beta}$ assumption is defined using the following experiment between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$. First, $\mathcal{C}$ uniformly samples a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ and sends it to $\mathcal{A}$. Then $\mathcal{A}$ adaptively makes two types of queries: syndrome queries, to which $\mathcal{C}$ replies with a uniformly sampled vector

$\mathbf{t} \leftarrow \mathbb{Z}_q^n$, and preimage queries, where $\mathcal{A}$ queries a vector $\mathbf{t}' \in \mathbb{Z}_q^n$, to which C replies with a short vector $\mathbf{y}' \leftarrow D_{\mathbb{Z}^m,\sigma}$ such that $\mathbf{C}\mathbf{y}' = \mathbf{t}'$. If ℓ is the total number of preimage queries, we ask the adversary to output $\ell + 1$ pairs of the form $\{(\mathbf{y}_j, \mathbf{t}_j)\}_{j\in[\ell+1]}$, such that $\mathbf{C}\mathbf{y}_j = \mathbf{t}_j$, $\|\mathbf{y}_j\| \leq \beta$ and $\mathbf{t}_j$ were provided via syndrome queries, for all $j \in [\ell + 1]$. We say that the one-more-ISIS$_{q,n,m,\sigma,\beta}$ problem is hard if the probability that $\mathcal{A}$ succeeds in the above game is negligible.

Note that this definition is reminiscent to the chosen target version of the one-more-RSA inversion problem from [BNPS03]. It is also closely related to the k-SIS problem [BF11] which was introduced in the context of linearly homomorphic signatures. The k-SIS problem is as follows: Given a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$, and k short vectors $\mathbf{e}_1, \ldots, \mathbf{e}_k \in \mathbb{Z}^m$ satisfying $\mathbf{A} \cdot \mathbf{e}_i = \mathbf{0} \bmod q$, find a short vector $\mathbf{e} \in \mathbb{Z}^m$ satisfying $\mathbf{A} \cdot \mathbf{e} = \mathbf{0} \mod q$, such that $\mathbf{e}$ is not in $\mathbb{Q}$-span$(\mathbf{e}_1, \ldots, \mathbf{e}_k)$. In [BF11], the linearly homomorphic signature must intuitively sign a subspace. Hence for k-SIS, the goal is to restrict the attacker to the subspace of the signatures it has already seen; this prevents it from obtaining signatures of vectors out of the vector subspace that has already been signed. In contrast, in our one-more-ISIS, we do not want to restrict the subspace and indeed allow the attacker to query the oracle more times than the dimension of the whole space. But we are more demanding on the norm of the vector that the attacker must find.

In particular, even if the attacker manages to obtain a trapdoor for the matrix $\mathbf{C}$ via repeated preimage queries to the vector $\mathbf{0}$, this trapdoor will not be of sufficiently good quality to lead to an attack. In more detail, such a trapdoor enables sampling preimages to arbitrary images, and hence the attacker can output $\ell + 1$ pairs of the form $\{(\mathbf{y}_j, \mathbf{t}_j)\}_{j\in[\ell+1]}$, such that $\mathbf{C}\mathbf{y}_j = \mathbf{t}_j$ and $\mathbf{t}_j$ were provided via syndrome queries, for all $j \in [\ell + 1]$. However, it will be unable to meet the constraint that $\|\mathbf{y}_j\| \leq \beta$. We believe this assumption is very natural and are optimistic that it may have other applications.

Given our new assumption, one more unforgeability follows very naturally. In the proof,

the challenger can sample the PKE public and secret keys using the PKE setup algorithm, and not discard the secret key. Assuming correctness of PKE and with knowledge of PKE.sk, the challenger can extract the pairs $(\mathbf{x}_j, \mathbf{y}_j)$ corresponding to the signature of each message μ_j. We have by soundness of the NIZK that $\mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j = H(\mathsf{g}_j)$. By setting $\mathbf{A} = \mathbf{C} \cdot \mathbf{R}$ for a low norm matrix $\mathbf{R}$, we can (i) use the leftover hash lemma to argue that $\mathbf{A}$ appears uniform, and (ii) rewrite $\mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j$ as $\mathbf{C}(\mathbf{y}_j - \mathbf{R}\mathbf{x}_j)$. Finally, by programming the random oracle so that $H(\mathsf{g}_j)$ is a syndrome queried by the one-more-ISIS adversary yields the proof. Please see Section 7.6 for more details.

To justify our assumption, we attempted to cryptanalyze it. For some parameter regimes, the problem can be solved in polynomial time but, as far as we know, the problem is exponentially hard for the regimes that we use in the blind signature scheme. Broadly, we consider two approaches to solve one-more-ISIS: combinatorial and lattice-based algorithms, and we provide complexity results for one-more-ISIS using these approaches. We also formulate new cryptanalytic questions that the one-more-ISIS assumption raises. Please see Section 7.6.6 for more details.

Estimating Performance. We provide a detailed analysis of the performance of our new candidate in Section 7.6.5. To instantiate our new protocol based on one-more-ISIS, we use the following building blocks:

- For the hash function, we use SHA-3-256;
- For the trapdoor generation and preimage sampling, we follow Falcon-512 [FHK+];
- For the IND-CPA secure PKE, we adapt the CRYSTALS-Kyber encryption scheme [ABD+17];
- For the NIZK scheme, we follow the recent protocol of [LNP22b, Figure 10].

To make these building blocks compatible with each other, we need a working modulus that must satisfy the following constraints:

- Its prime factors must be sufficiently large to avoid soundness improving repetitions

in the zero knowledge proof;

- The moduli of the underlying signature and encryption schemes should divide the working modulus so that the relations required to be proven are simpler;
- The polynomial $x^{128} + 1$ defining the ring used in the NIZK scheme should split in exactly two prime factors modulo all factors of the working modulus, due to technical reasons related to the NIZK.

Satisfying the above constraints requires a delicate balancing of parameters, which results in a number of changes in the building blocks. First, we must modify Falcon's modulus because $x^{128} + 1$ splits completely modulo the original Falcon modulus, which violates the last constraint above. We must also instantiate the CRYSTALS-Kyber framework so that it complies with the zero-knowledge proof π from [LNP22b] and enjoys perfect correctness. Since it is the zero-knowledge proof that makes most of the overall signature size as well as the complexity to generate it, we are mainly interested in making the generation of π efficient, while potentially sacrificing the efficiency of the other routines.

We provide a detailed guideline on how to instantiate the NIZKAoK protocol from [LNP22b, Figure 10], and how to choose the parameters for the other building blocks so as to obtain a concrete estimate for all the parameters of the resultant blind signature scheme. We provide a python script (included with the submission) that estimates the concrete security of the building blocks, as well as the size of the resulting signatures. The resulting protocol has security relying on Ring-LWE [SSTX09; LPR10], Module-LWE [BGV12; LS15] and the Module-SIS [LS15] variant of one-more-ISIS.

Using our script, we obtain a signature of size less than 44KB, for a classical core-SVP hardness of 109 bits (following the security methodology from [ADPS16]). Note that bit security is typically estimated to be higher than core-SVP hardness (see [ABD+17]), and we expect it to be of the order of 128 bits. The transcript has size less than 1.5KB. The costs of the signer and user in the signing protocol, as well as that of the verifier are also very low. To see this, note that the signer must simply compute a GPV pre-image, the user must compute a ciphertext and proof for a linear statement with small unknowns,

while the verifier must verify this proof. Thus, in the end, we obtain a protocol which enjoys security under a post-quantum assumption and is overall more efficient than all prior candidates.

Other Related Works. Aside from lattice based blind signatures, there are a few other constructions from conjectured post-quantum assumptions. The most relevant to our work is the code-based construction of Blazy *et al.* [BGSS17], relying on the CFS signature scheme [CFS01] and Stern zero-knowledge proofs [Ste96]. Like in our one-more-ISIS construction, their construction relies on a new assumption, related to CFS. However, there are important differences with our work. In CFS, not all syndromes can be inverted, and the procedure needs to be repeated if no inversion is possible. Hence, the resulting blind signature scheme is not round optimal. Moreover, due to the poor scaling of CFS signatures and the use of Stern proofs, their construction achieves a signature size of several MB. A blind signature based on multivariate polynomial systems was described in [PSM17], with a non-standard unforgeability security property.

Organisation of the chapter. The rest of the chapter is organised as follows. In Section 7.4, we provide the preliminaries used in the chapter. Then we begin with instantiating a blind signature scheme from Fischlin's signature in Section 7.5. In Section 7.6, we define our one-more-ISIS assumption and provide our construction of blind signature under the assumption in Section 7.6.2. This construction satisfies honest signer blindness, which can easily be extended to full fledged blindness. We provide the construction for full-fledged blindness in Appendix 7.A.

7.4 PRELIMINARIES

In this section, we provide the preliminaries used in this chapter.

Notations used in this chapter. Any vector $\mathbf{v}$ in this chapter is a column vector by default, and $\mathbf{v}^\mathsf{T}$ is a row vector. For any vector $\mathbf{v}$, we denote its ith element by $\mathbf{v}[i]$. Similarly, for any matrix $\mathbf{M}$, $\mathbf{M}[i][j]$ represents the element in the jth column of ith row. For a set S, $|S|$ represents the cardinality of S, while if $x \in \mathbb{R}$, then $|x|$ represents absolute value of x. $\mathcal{D}_{\Lambda,s,\mathbf{c}}$ represents discrete Gaussian distribution over lattice Λ, with center $\mathbf{c}$ and standard deviation parameter s. When $\mathbf{c} = 0$, we omit it. Similarly, we omit Λ, if $\Lambda = \mathbb{Z}$. For a distribution D over a countable set $\mathcal{X}$, we let $H_\infty(D) = -\max_{x\in\mathcal{X}} \log_2 D(x)$ denote the min-entropy of D. The statistical distance between two distributions D_0 and D_1 over $\mathcal{X}$ is defined as $\frac{1}{2}\sum_{x\in\mathcal{X}} |D_0(x) - D_1(x)|$.

We use standard definitions for pseudo-random functions (PRF), public-key encryption (PKE) and signatures as provided in Chapter 2. For lattice-related preliminaries, we refer to Chapter 2.

We place ourselves in a setup that allows the attackers to run in time $2^{o(\lambda)}$ and succeed with probability $2^{-o(\lambda)}$, but that forbids them to make more than $\mathrm{poly}(\lambda)$ interactions with honest users. Compared to the setup of polynomially bounded attackers, this allows to better reflect practice and to better differentiate between operations that the adversary can do on its own and are only limited by the adversary runtime (such as hash queries) and operations that require interaction with a honest user and are much more limited (such as signature queries). We note that if we limit ourselves to polynomially bounded adversaries, then all our reductions of our security proofs involve polynomial-time reductions and would not require subexponential hardness assumptions.

7.4.1 Blind Signatures

To begin, we introduce some notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y)\rangle$, we denote the joint execution of $\mathcal{X}$ and $\mathcal{Y}$ where $\mathcal{X}$ has private input x, $\mathcal{Y}$ has private input y and $\mathcal{X}$ receives private output a while $\mathcal{Y}$ receives private output b.

Definition 7.1 (Blind Signature)**.** A blind signature scheme BS consists of PPT algorithms Gen, Vrfy along with interactive PPT algorithms $\mathcal{S}, \mathcal{U}$ such that for any λ:

- $\mathsf{Gen}(1^\lambda)$ generates a key pair $(\mathsf{BSig.sk}, \mathsf{BSig.vk})$.
- The joint execution of $\mathcal{S}(\mathsf{BSig.sk})$ and $\mathcal{U}(\mathsf{BSig.vk}, \mu)$, where $\mu \in \{0,1\}^*$, generates an output σ for the user and no output for the signer; this is denoted as $(\perp, \sigma) \leftarrow \langle \mathcal{S}(\mathsf{BSig.sk}), \mathcal{U}(\mathsf{BSig.vk}, \mu) \rangle$.
- Algorithm $\mathsf{Vrfy}(\mathsf{BSig.vk}, \mu, \sigma)$ outputs a bit b.

The scheme must satisfy completeness: for any $(\mathsf{BSig.sk}, \mathsf{BSig.vk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mu \in \{0,1\}^*$ and σ output by $\mathcal{U}$ in the joint execution of $\mathcal{S}(\mathsf{BSig.sk})$ and $\mathcal{U}(\mathsf{BSig.vk}, \mu)$, it holds that $\mathsf{Vrfy}(\mathsf{BSig.vk}, \mu, \sigma) = 1$ with probability $1 - \lambda^{-\omega(1)}$.

Blind signatures must satisfy two security properties: one more unforgeability and blindness [JLO97].

Definition 7.2 (One More Unforgeability)**.** The blind signature $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ is one more unforgeable if for any polynomial Q_S, and any algorithm $\mathcal{U}^*$ with run-time $2^{o(\lambda)}$, the success probability of $\mathcal{U}^*$ in the following game is $2^{-\Omega(\lambda)}$:

1. $\mathsf{Gen}(1^\lambda)$ outputs $(\mathsf{BSig.sk}, \mathsf{BSig.vk})$, and algorithm $\mathcal{U}^*$ is given $\mathsf{BSig.vk}$.
2. Algorithm $\mathcal{U}^*$ interacts concurrently with Q_S instances $\mathcal{S}^1_{\mathsf{BSig.sk}}, \ldots, \mathcal{S}^{Q_S}_{\mathsf{BSig.sk}}$.
3. Algorithm $\mathcal{U}^*$ outputs $(\mu_1, \sigma_1, \ldots, \mu_{Q_S+1}, \sigma_{Q_S+1})$.

Algorithm $\mathcal{U}^*$ succeeds if $\mathsf{Vrfy}(\mathsf{BSig.vk}, \mu_i, \sigma_i) = 1$ for all $i \in [Q_S + 1]$ and the μ_i's are distinct.

The blindness condition says that it should be infeasible for any malicious signer $\mathcal{S}^*$ to decide which of two messages μ_0 and μ_1 of its choice has been signed first in two executions with a honest user $\mathcal{U}$. If one of these executions has returned $\perp$, then the signer is not informed about the other signature either. We will focus on the following notion of honest signer blindness.

Definition 7.3 (Honest Signer Blindness)**.** The blind signature $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ satisfies honest signer blindness if for any algorithm $\mathcal{S}^*$ with run-time $2^{o(\lambda)}$, the advantage of $\mathcal{S}^*$ in the following game is $2^{-\Omega(\lambda)}$:

1. $\mathsf{Gen}(1^\lambda)$ outputs $(\mathsf{BSig.sk}, \mathsf{BSig.vk})$ and gives it to $\mathcal{S}^*$; algorithm S^* outputs two messages μ_0, μ_1 of its choice.

2. A random bit b is chosen and $\mathcal{S}^*$ interacts concurrently with $\mathcal{U}_0 := \mathcal{U}(\mathsf{BSig.vk}, \mu_b)$ and $\mathcal{U}_1 := \mathcal{U}(\mathsf{BSig.vk}, \mu_{\bar{b}})$ possibly maliciously; when $\mathcal{U}_0$ and $\mathcal{U}_1$ have completed their executions, the values $\sigma_b, \sigma_{\bar{b}}$ are defined as follows:

   - If either $\mathcal{U}_0$ or $\mathcal{U}_1$ aborts, then $(\sigma_b, \sigma_{\bar{b}}) := (\perp, \perp)$.

   - Otherwise, let σ_b (resp. $\sigma_{\bar{b}}$) be the output of $\mathcal{U}_0$ (resp. $\mathcal{U}_1$).

   Algorithm $\mathcal{S}^*$ is given (σ_0, σ_1).

3. Algorithm $\mathcal{S}^*$ outputs a bit b'.

Algorithm $\mathcal{S}^*$ succeeds if $b' = b$. If succ denotes the latter event, then the advantage of $\mathcal{S}^*$ is defined as $|\Pr[\mathsf{succ}] - 1/2|$.

Full-fledged blindness lets the adversary $\mathcal{S}^*$ sample its own pair $(\mathsf{BSig.sk}, \mathsf{BSig.vk})$ at Step 1 (possibly maliciously), and gives $\mathsf{BSig.vk}$ to the challenger.

7.4.2 Non-Interactive Zero Knowledge Arguments

Definition 7.4 (Non Interactive Zero Knowledge Argument)**.** A non-interactive zero-knowledge (NIZK) argument system Π for an NP relation R consists of three PPT algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \rightarrow \mathsf{crs}$: On input a security parameter λ, the Gen algorithm outputs a common reference string crs; in the random oracle model, this algorithm may be skipped, since the crs can be generated by P and V by querying the random oracle on some fixed value.

- $\mathsf{P}(\mathsf{crs}, x, w) \rightarrow \pi$: On input the common reference string crs, a statement $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$, a witness w such that $(x, w) \in R$, the prover P outputs a proof π.

- $\mathsf{V}(\mathsf{crs}, x, \pi) \rightarrow \mathsf{accept/reject}$: On input a common reference string crs, a statement $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$ and a proof π, the verifier V outputs accept or reject.

The argument system Π should satisfy the following properties.

- **Completeness:** For any $(x, w) \in R$, we have

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1] \geq 1 - \lambda^{-\omega(1)}.$$

- **Soundness:** Let L be the language corresponding to NP relation R. For any $x \in \{0,1\}^{\text{poly}(\lambda)}$ such that $x \notin L$ and any $2^{o(\lambda)}$ time prover P^*, we have

$$\Pr[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \pi \leftarrow \mathsf{P}^*(\mathsf{crs}, x) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1] \leq 2^{-\Omega(\lambda)}.$$

- **Honest Verifier Zero Knowledge:** There is a PPT simulator Sim such that, for all statements x for which there exists w with $R(x,w) = 1$, for any $2^{o(\lambda)}$ time adversary $\mathcal{A}$, we have:

$$\begin{aligned} &\Big| \Pr\left[1 \leftarrow \mathcal{A}\left((\mathsf{crs}, x, \pi) \ : \ \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda),\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)\right)\right] \\ &- \Pr\left[1 \leftarrow \mathcal{A}\left((\mathsf{crs}, x, \pi) \ : \ (\mathsf{crs}, \pi) \leftarrow \mathsf{Sim}(1^\lambda, x)\right)\right] \Big| \leq 2^{-\Omega(\lambda)}. \end{aligned}$$

Definition 7.5 (Argument of Knowledge)**.** The argument system $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is called an argument of knowledge for the relation R if it is complete and knowledge-sound as defined below.

- **Knowledge Sound:** For any $2^{o(\lambda)}$ time prover P^*, there exists an extractor $\mathcal{E}$ with expected run-time polynomial in λ and the run-time of P^*, such that for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{r|l} \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), & \\ (x, s) \leftarrow \mathcal{A}(\mathsf{crs}), & \\ \pi^* \leftarrow \mathsf{P}^*(\mathsf{crs}, x, s), & (x,w) \notin R \wedge b = \mathsf{accept} \\ b \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi^*), & \\ w \leftarrow \mathcal{E}^{\mathsf{P}^*(\mathsf{crs}, x, s)}(\mathsf{crs}, x, \pi^*, b) & \end{array}\right] \leq 2^{-\Omega(\lambda)}.$$

If an argument of knowledge is also non-interactive zero knowledge, it is termed as a non-interactive zero knowledge argument of knowledge, abbreviated as NIZKAoK.

7.5 STARTING POINT: INSTANTIATING FISCHLIN'S BLIND SIGNATURE

A simple way to obtain a two-round blind signature from lattices is to instantiate Fischlin's construction [Fis06].

7.5.1 Construction

The construction uses the following building blocks:

1. A hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ that will be modeled as random oracle in the unforgeability proof.

2. A CPA-secure PKE scheme PKE that is perfectly correct.

3. A $\mathsf{NIZKAoK}$ for the statement of Equation (7.1).

The construction is given as follows:

Setup. $\mathsf{Gen}(1^\lambda)$: Upon input the security parameter λ, define $n, m, q, \sigma, \beta = \sigma\sqrt{m}$ as functions of λ such that q is prime, $\mathsf{SIS}_{q,n,m,2\beta}$ is hard and the scheme is both efficient and complete; then do the following:

- Run $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and discard $\mathsf{PKE.sk}$.
- Compute $(\mathbf{C}, \mathbf{T_C}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
- Output $\mathsf{BSig.sk} = \mathbf{T_C}$, $\mathsf{BSig.vk} = (\mathbf{C}, \mathsf{PKE.pk})$.

Signing. $\langle \mathcal{S}(\mathsf{BSig.sk}), \mathcal{U}(\mathsf{BSig.vk}, \mathsf{g}) \rangle$:

1. **User:** Given the key $\mathsf{BSig.vk}$ and a message g, user $\mathcal{U}$ does the following:
   - It samples $\mathsf{PKE.Enc}$ randomness r and computes $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mu; r)$.
   - It sends ct to the signer.
2. **Signer:** Upon receiving ct, signer $\mathcal{S}$ does the following:
   - It computes $H(\mathsf{ct})$ and samples $\mathbf{y} \leftarrow \mathsf{SamplePre}(\mathbf{C}, \mathbf{T_C}, H(\mathsf{ct}), \sigma)$; we have that $\mathbf{y}$ is short and $\mathbf{Cy} = H(\mathsf{ct})$.
   - It sends $\mathbf{y}$ to the user.
3. **User:** Upon receiving $\mathbf{y}$, user $\mathcal{U}$ does the following:
   - It verifies that $\|\mathbf{y}\| \leq \beta$ and $\mathbf{Cy} = H(\mathsf{ct})$ and aborts if this fails.
   - It generates a $\mathsf{NIZKAoK}$ π for following statement: Given $\mathsf{BSig.vk} = (\mathbf{C}, \mathsf{PKE.pk})$ and μ, there exists r and a vector $\mathbf{y}$ such that
   $$\|\mathbf{y}\| \leq \beta \ \wedge \ \mathbf{Cy} = H(\mathsf{Enc}(\mathsf{PKE.pk}, \mu; r)). \tag{7.1}$$
   - The signature is π.

Verifying. The verifier accepts if the proof π is valid, and rejects if it is not.

The parameters q, n, m, σ are set such that $n = \Omega(\lambda)$, Lemmas 2.2 and 2.3 are applicable, and $\mathsf{SIS}_{q,m,n,2\beta}$ is hard with $\beta = \sigma\sqrt{m}$. The completeness of the scheme follows from the choice of β (using the Gaussian tail bound from Lemma 2.1) and the completeness of the NIZKAoK.

Note that Steps 1 and 2 of the signing algorithm can be implemented quite efficiently. Step 3 is much more costly and results in a large signature bit-size. This is because the statement of Equation (7.1) involves the hash function H (in particular, the input of H must be kept secret). Note that we make a non-black-box use of H in the scheme, but require it to be modeled as a random oracle in the unforgeability proof.

Security We show that the construction satisfies one more unforgeability and blindness.

7.5.2 Unforgeability

Theorem 7.1. *Assume that* $\mathsf{SIS}_{q,n,m,2\beta}$ *is hard and the* NIZKAoK *is knowledge sound. Then the blind signature scheme above, in Section 7.5.1, is one more unforgeable in the random oracle model.*

Proof. We argue one more unforgeability using the following hybrids.

Hybrid_0: This is the genuine one more unforgeability experiment.

Hybrid_1: In this hybrid, the challenger (which plays the role of the signer) does not discard the decryption key PKE.sk. For every sign query c_j, it uses PKE.sk to decrypt c_j into a plaintext μ_j (which can be $\perp$ in case decryption fails). It stores the μ_j's.

Hybrid_2: The difference between this hybrid and the previous one is in how the hash and sign queries are answered. On a fresh input c for a hash query, the challenger first samples $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ and returns $H(c) = \mathbf{C}\mathbf{y}$. To answer a signing query for an input c, the challenger returns the corresponding $\mathbf{y}$ that it must have sampled while answering the hash query for c. If the sign query is made before the corresponding hash query, then the challenger first sets the hash value as above and then returns the corresponding $\mathbf{y}$.

Indistinguishability of hybrids

1. The differences between Hybrid_0 and Hybrid_1 are only concerning the inner computations of the challenger and not its interactions with the adversary. Hence, the two hybrids are identical in the view of the adversary.

2. By Lemmas 2.2, 2.3 and 2.1, the views of the adversary in Hybrid_1 and Hybrid_2 are within statistical distance $(Q_S + Q_H) \cdot 2^{-\Omega(\lambda)}$ from one another, where Q_S is the number of signing queries and Q_H is the number of hash queries[1].

Assume now that the adversary succeeds in Hybrid_2 with probability ε. When it succeeds, it generates distinct messages $(\mu_i)_{i \leq Q_S+1}$ and corresponding signatures, i.e., proofs $(\pi_i)_{i \leq Q_S+1}$ for the statement of Equation (7.1), such that all these proofs are accepted. As the adversary makes at most Q_S sign queries, at least one of these μ_i's cannot be part of the μ_j's stored by the challenger: let $\mu^\star$ be an arbitrary such message and $\pi^\star$ be its associated proof.

Using the knowledge soundness of the $\mathsf{NIZKAoK}$ on $\pi^\star$, the challenger extracts a witness $(r^\star, \mathbf{y}^\star)$ such that $\|\mathbf{y}^\star\| \leq \beta$ and $\mathbf{C}\mathbf{y}^\star = H(\mathsf{ct}^\star)$ with $\mathsf{ct}^\star = \mathsf{Enc}(\mathsf{PKE.pk}, \mu^\star; r^\star)$. By perfect correctness of PKE, the ciphertext $\mathsf{ct}^\star$ decrypts to $\mu^\star$. By definition, the message $\mu^\star$ cannot have been queried for a signature. However, it must have been queried for a hash, as otherwise the equality $\mathbf{C}\mathbf{y}^\star = H(\mathsf{ct}^\star)$ would hold with probability at most q^{-n}. This implies that the challenger has previously sampled a vector $\mathbf{y} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ such that $\mathbf{C}\mathbf{y} = H(\mathsf{ct}^\star)$. By Lemma 2.1, we have $\|\mathbf{y}\| \leq \beta = \sigma\sqrt{m}$ with probability $1 - 2^{-\Omega(\lambda)}$ and $\mathbf{y} = \mathbf{y}^\star$ with probability $2^{-\Omega(\lambda)}$. We conclude that $\mathbf{y} - \mathbf{y}^\star$ is non-zero, has norm $\leq 2\beta$

[1] We note here that $\mathsf{SamplePre}$ is assumed to be deterministic (see Section 2.1.2), without which the claim would not be true.

and satisfies $\mathbf{C}(\mathbf{y} - \mathbf{y}^\star) = \mathbf{0}$, providing a solution to the $\mathsf{SIS}_{q,n,m,2\beta}$ instance $\mathbf{C}$. ■

7.5.3 Blindness

Theorem 7.2. *Assume that* PKE *is* IND-CPA *secure and the* NIZKAoK *is zero-knowledge. Then the blind signature scheme above, in Section 7.5.1, satisfies honest signer blindness.*

Proof. We argue blindness using the following hybrids.

Hybrid_0: This is the genuine honest signer blindness experiment.

Hybrid_1: In this hybrid, the proofs π_b and $\pi_{\overline{b}}$ are replaced with simulated proofs.

Hybrid_2: In this hybrid, the ciphertexts ct_b and $\mathsf{ct}_{\overline{b}}$ are changed to independent encryptions of 0.

Indistinguishability of hybrids

1. Hybrid_0 and Hybrid_1 are indistinguishable in the view of the adversary, because of the zero-knowledge property of the NIZKAoK.
2. Hybrid_1 and Hybrid_2 are indistinguishable in the view of the adversary, because of the IND-CPA security of PKE.

In Hybrid_2, the distinguishing advantage of the adversary is 0, because its views for $b = 0$ and $b = 1$ are statistically identical. ■

Full-Fledged Blindness. Note that the scheme as stated may not satisfy full-fledged blindness. In particular, if the malicious signer does not discard PKE.sk in the setup phase, it could use it to decrypt the ciphertexts in the challenge phase and break blindness. However, the security proof above can be extended to handle full-fledged blindness if we can ensure that PKE.pk has been honestly generated by the adversarial signer, without a corresponding decryption key. For example, if PKE.pk is computationally indistinguishable from uniform, then we could replace PKE.pk in the scheme by the

output of another hash function H' modeled as a random oracle, on an arbitrary public input. Since the secret key must anyway be discarded in the construction, setting the public key as the output of the random oracle ensures that the adversarial signer cannot know the corresponding secret key. In the (full fledged blindness) security proof, we would then introduce a very first game in which the output of H' is replaced by a properly generated PKE.pk. Note that a maliciously generated $\mathbf{C}$ has no impact on blindness since it is not involved in the user's message to the signer.

7.5.4 Efficiency Estimate

We consider the following instantiation of the building blocks.

- For PKE, we can take any lattice-based public-key encryption scheme. It is only required to be IND-CPA, but it must be perfectly correct. The latter property can typically be guaranteed by tail-cutting error distributions and increasing the working modulus sufficiently. Also, lattice-based encryption schemes typically have public keys that are computationally indistinguishable from uniform, as required for the full fledged blindness adaptation described above. For example, one could use a variant of the NewHope scheme [ADPS16], modified to provide perfect correctness. It is expected that ciphertexts will be of bitlengths below a few KB.

- For the underlying signature scheme, we recommend using the Falcon scheme [FHK+], which is an efficient instantiation of the TrapGen-SamplePre framework from [GPV08]. With this choice, the second transcript will have the size below 1KB. Also, that makes the signer particularly efficient – for instance, using Falcon [FHK+], signing time is in the range 0.15 – 0.3 ms depending on the choice of parameters.

- As the hash function is modeled as a random oracle in the unforgeability proof, one could use SHA-3-256. With the above choices for the public-key encryption and signature schemes, one may need more than 15 sponge absorbing steps for reading the input and 7 sponge squeezing steps to write the output.

- Unfortunately, as the statement of Equation (7.1) involves a hash function H that is modeled as a random oracle in the unforgeability proof, it seems we are bound to use an all-purpose NIZKAoK. For example, one could use an instantiation of Aurora [BCR+19]. Estimating a precise cost is difficult, but we do not expect a proof of size below 100KB. We also do not expect the prover runtime to be below 1 hour, whereas the verifier runtime could be significantly lower. It could be beneficial to use hash functions designed to be compatible with all-purpose NIZKAoK, such as [AAB+20; GKR+21].

7.6 TWO ROUND BLIND SIGNATURE FROM ONE-MORE-ISIS

In this section, we describe a significantly more practical scheme, under a new assumption.

7.6.1 The One-More-ISIS Assumption

We first introduce the one-more-ISIS hardness assumption. As it is a new assumption, we provide a detailed assessment of potential attacks, in Subsection 7.6.6.

Informally, the one-more-ISIS assumption states that for any polynomially bounded ℓ, it is difficult to forge $\ell + 1$ GPV signatures [GPV08], even when given access to up to ℓ inversions of arbitrary syndromes. We stress that these are not signature queries, as a query for a message μ corresponds to a *uniformly distributed* syndrome $H(\mu)$ (modeling H by a random oracle), whereas here the attacker is allowed to make inversion queries for *arbitrary* syndromes. As a result, one-more-ISIS could possibly be easier to solve than it is to break the chosen-message security of the GPV signature scheme.

Definition 7.6. Let q, n, m, σ, β be functions of security parameter λ. The one-more-ISIS$_{q,n,m,\sigma,\beta}$ assumption is defined using the following experiment.

1. The challenger C uniformly samples a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ and sends $\mathbf{C}$ to adversary $\mathcal{A}$.
2. The adversary adaptively makes queries of the following types to the challenger, in any order.
   - **Syndrome queries.** The adversary $\mathcal{A}$ requests C for a challenge vector, to which C replies with a uniformly sampled vector $\mathbf{t} \leftarrow \mathbb{Z}_q^n$. We denote the set of received vectors by S.
   - **Preimage queries.** The adversary $\mathcal{A}$ queries a vector $\mathbf{t}' \in \mathbb{Z}_q^n$, to which C replies with a short vector $\mathbf{y}' \leftarrow D_{\mathbb{Z}^m,\sigma}$ such that $\mathbf{C}\mathbf{y}' = \mathbf{t}'$. We denote by ℓ the total number of preimage queries.
3. In the end, the adversary $\mathcal{A}$ outputs $\ell + 1$ pairs of the form $\{(\mathbf{y}_j, \mathbf{t}_j)\}_{j \in [\ell+1]}$.
4. The adversary wins if $\mathbf{C}\mathbf{y}_j = \mathbf{t}_j$, $\|\mathbf{y}_j\| \leq \beta$ and $\mathbf{t}_j \in S$ for all $j \in [\ell + 1]$.

The one-more-ISIS$_{q,n,m,\sigma,\beta}$ assumption states that for every adversary $\mathcal{A}$ running in time $2^{o(\lambda)}$ making at most $\lambda^{O(1)}$ preimage queries and $2^{o(\lambda)}$ syndrome queries, the probability (over the randomness of $\mathcal{A}$ and C) that $\mathcal{A}$ wins is $2^{-\Omega(\lambda)}$.

The definition is reminiscent of the chosen target version of the one-more-RSA inversion problem from [BNPS03]. We could define a variant of one-more-ISIS inspired from the known target version of the one-more-RSA inversion problem from [BNPS03], in which the set S is restricted to be of size $\ell + 1$. The choice (chosen target) of formulation made in Definition 7.6 is driven by the security proof of the blind signature scheme. In the RSA setting, the chosen and known target versions reduce to one another, but this seems difficult to adapt to the ISIS setting.

7.6.2 Construction

The construction uses the following building blocks:

1. A hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ that will be modeled as random oracle in the unforgeability proof.
2. A NIZK for the statement of Equation (7.2).
3. A CPA-secure PKE scheme PKE that is perfectly correct.

The construction is provided below.

Setup. $\mathsf{Gen}(1^\lambda)$: Upon input the security parameter λ, define $n, m, q, \sigma, \beta = \sigma\sqrt{m}$ as functions of λ such that q is prime, one-more-ISIS$_{q,n,m,\sigma,2\beta}$ is hard and the scheme is both efficient and complete; then do the following:

- Run $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and discard PKE.sk.
- Compute $(\mathbf{C}, \mathbf{T_C}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
- Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$.
- Output $\mathsf{BSig.sk} = \mathbf{T_C}$, $\mathsf{BSig.vk} = (\mathbf{C}, \mathbf{A}, \mathsf{PKE.pk})$.

Signing. $\langle \mathcal{S}(\mathsf{BSig.sk}), \mathcal{U}(\mathsf{BSig.vk}, \mathsf{g}) \rangle$:

1. **User:** Given the key BSig.vk and a message g, user $\mathcal{U}$ does the following:
   - It samples $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma/m}$.
   - It computes $\mathbf{t} = \mathbf{A}\mathbf{x} + H(\mathsf{g})$.

- It sends $\mathbf{t}$ to the signer.

2. **Signer:** Upon receiving $\mathbf{t}$, signer $\mathcal{S}$ does the following:
   - It samples a short vector $\mathbf{y} \leftarrow \mathsf{SamplePre}(\mathbf{C}, \mathbf{T}_{\mathbf{C}}, \mathbf{t}, \sigma)$; we have $\mathbf{Cy} = \mathbf{t}$.
   - It sends $\mathbf{y}$ to the user.
3. **User:** Upon receiving $\mathbf{y}$, user $\mathcal{U}$ does the following:
   - It verifies that $\|\mathbf{y}\| \leq \beta$ and satisfies $\mathbf{Cy} = \mathbf{t}$.
   - It samples $\mathsf{PKE.Enc}$ randomness r and computes
   $$\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r).$$
   - It generates a NIZK π for following statement: Given $\mathsf{BSig.vk} = (\mathbf{C}, \mathbf{A}, \mathsf{PKE.pk})$, ct and μ, there exists r and vectors $\mathbf{x}, \mathbf{y}$ such that
   $$\|\mathbf{x}\| \leq \beta/m \wedge \|\mathbf{y}\| \leq \beta \wedge \mathbf{Cy} - \mathbf{Ax} = H(\mathsf{g}) \wedge \mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r). \tag{7.2}$$
   - The signature is (π, ct).

Verifying. The verifier accepts if the proof π is valid, and rejects if it is not.

The parameters q, n, m, σ are set such that Lemmas 2.2 and 2.3 are applicable, the distribution of $\mathbf{Ax}$ is close to uniform at Step 1 of the signing algorithm (using Lemmas 2.3 and 2.1 with standard deviation parameter $\sigma/m = \Omega(1)$), and $\mathsf{one\text{-}more\text{-}ISIS}_{q,m,n,\sigma,2\beta}$ is hard with $\beta = \sigma\sqrt{m}$.

Completeness We make the following observations to argue completeness. From the correctness of $\mathsf{SamplePre}$, the vector $\mathbf{y}$ is small and satisfies $\mathbf{Cy} = \mathbf{t}$, where $\mathbf{t} = \mathbf{Ax} + H(\mu)$. This gives us $\mathbf{Cy} - \mathbf{Ax} = H(\mu)$. Furthermore, the vector $\mathbf{x}$ is small by design and $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r)$ by construction. Hence, the proof π for Equation (7.2) verifies and the user accepts the proof because of the completeness of NIZK.

We now make a few remarks about the construction. Observe that we choose $\mathbf{x}$ to have norm at most β/m, which is a factor m smaller than that of $\mathbf{y}$. This is because in the security proof, we will construct solutions to the $\mathsf{one\text{-}more\text{-}ISIS}_{q,n,m,\sigma,2\beta}$ problem as $\mathbf{y} - \mathbf{R}\mathbf{x}$ (see Step 5 of the unforgeability proof), where $\mathbf{R} \leftarrow \{0,1\}^{m\times m}$. Thus, choosing $\|\mathbf{x}\| \leq \beta/m$ and $\|\mathbf{y}\| \leq \beta$ allows us to bound the norm of the $\mathsf{one\text{-}more\text{-}ISIS}$ solution by 2β as desired. Note that by increasing the ratio between the norms of $\mathbf{x}$ and $\mathbf{y}$ further, one can decrease the quantity 2β to a value that is arbitrarily close to β (hence possibly weakening the hardness assumption). Another important component is the inclusion of ciphertext $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r)$ in the signature. It enables to circumvent rewinding in the extraction of all the witnesses $(\mathbf{x}_i\|\mathbf{y}_i)$ of the $Q_S + 1$ message-signature pairs output by the adversary, in the proof of unforgeability (see Step 5). Without it, the reduction may need to rewind $Q_S + 1$ times to extract all the pairs $(\mathbf{x}_i, \mathbf{y}_i)$, to construct the $\mathsf{one\text{-}more\text{-}ISIS}$ solution, leading to a security loss exponential in Q_S.

Security. We show that our construction satisfies one more unforgeability and blindness.

7.6.3 Unforgeability

Theorem 7.3. *Assume that* NIZK *is sound. Then if there exists an adversary* $\mathcal{A}$ *in the random oracle model that issues* Q_S *signing queries and any number of hash queries and outputs* $Q_S + 1$ *signatures with probability* δ*, then there exists an algorithm* $\mathcal{B}$ *that runs in essentially the same time as* $\mathcal{A}$ *and requests* Q_S *preimage queries and wins the* $\mathsf{one\text{-}more\text{-}ISIS}_{q,n,m,\sigma,2\beta}$ *game with probability at least* $\delta - 2^{-\Omega(\lambda)} - (Q_S+1)(2^{-\Omega(\lambda)} + q^{-n})$.

Proof. We construct the proof using the following hybrids.

Hybrid_0: This is the genuine one more unforgeability experiment.

Hybrid_1 : In this hybrid, the challenger does not discard the decryption key $\mathsf{PKE.sk}$. For every signature $\sigma_j = (\pi_j, \mathsf{ct}_j)$ output by the adversary (for $j \in [Q_S + 1]$), it uses

$\mathsf{PKE.sk}$ to decrypt ct_j into a plaintext $(\mathbf{x}_j \| \mathbf{y}_j)$ (which can be $\perp$ in case decryption fails). It stores the $(\mathbf{x}_j \| \mathbf{y}_j)$'s.

Hybrid_2 : This hybrid differs from the previous one in the way matrix $\mathbf{A}$ is chosen. The challenger first samples a binary matrix $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$ and sets $\mathbf{A} = \mathbf{CR}$.

Indistinguishability of hybrids

In the following, we let $\mathsf{Adv}_i^{\mathsf{omuf}}$ represent the advantage of $\mathcal{A}$ in the one more unforgeability game in Hybrid_i. Then $\mathsf{Adv}_0^{\mathsf{omuf}} = \delta$.

1. The differences between Hybrid_0 and Hybrid_1 are only concerning the inner computations of the challenger and not its interactions with the adversary. Hence, the two hybrids are identical in the view of the adversary. Thus $\mathsf{Adv}_1^{\mathsf{omuf}} = \mathsf{Adv}_0^{\mathsf{omuf}} = \delta$.

2. The only difference between Hybrid_1 and Hybrid_2 is that in the latter $\mathbf{A}$ is computed as $\mathbf{CR}$, where $\mathbf{R}$ is a uniform binary matrix, instead of sampling it uniformly randomly from $\mathbb{Z}_q^{n \times m}$. The two hybrids are indistinguishable because Lemmas 2.3 and 2.1 imply that $(\mathbf{C}, \mathbf{A})$ is within statistical distance $2^{-\Omega(\lambda)}$ from $(\mathbf{C}, \mathbf{CR})$. Thus $\mathsf{Adv}_2^{\mathsf{omuf}} \geq \mathsf{Adv}_1^{\mathsf{omuf}} - 2^{-\Omega(\lambda)} = \delta - 2^{-\Omega(\lambda)}$.

We conclude with the following claim.

Claim 7.4. Assume that the NIZK argument system is sound. Then if there is an adversary $\mathcal{A}$ in the random oracle model that makes at most Q_S signing queries and succeeds in generating $Q_S + 1$ signatures with probability ε in Hybrid_2, then there exists a $\mathsf{one\text{-}more\text{-}ISIS}$ adversary $\mathcal{B}$, with essentially the same run time as $\mathcal{A}$, with Q_S preimage queries with success probability at least $\varepsilon - (Q_S + 1)(2^{-\Omega(\lambda)} + q^{-n})$.

Proof. The reduction $\mathcal{B}$ is as follows.

1. Upon being challenged by the $\mathsf{one\text{-}more\text{-}ISIS}$ challenger C, with matrix $\mathbf{C}$, algorithm $\mathcal{B}$ does the following:

   - It uniformly samples a binary matrix $\mathbf{R}$ and sets $\mathbf{A} = \mathbf{CR}$.
   - It samples $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
   - It invokes $\mathcal{A}$ with $(\mathbf{A}, \mathbf{C}, \mathsf{PKE.pk})$ as verification key.

2. In response to each (fresh) hash query on input μ from $\mathcal{A}$, algorithm $\mathcal{B}$ makes a syndrome query to C. Challenger C returns a uniform vector $\mathbf{t} \in \mathbb{Z}_q^n$, which $\mathcal{B}$ forwards to $\mathcal{A}$ as $H(\mu)$.

3. To answer a signing query on input $\mathbf{t}'$, algorithm $\mathcal{B}$ forwards $\mathbf{t}'$ to C as a preimage query. Challenger C returns a short vector $\mathbf{y}'$, such that $\mathbf{C}\mathbf{y}' = \mathbf{t}'$. Algorithm $\mathcal{B}$ forwards $\mathbf{y}'$ to $\mathcal{A}$.

4. Eventually, adversary $\mathcal{A}$ outputs $Q_S + 1$ message-signature pairs $\{\mu_j, (\pi_j, \mathsf{ct}_j)\}_{j \in [Q_S+1]}$.

5. If the π_j's pass verification, then algorithm $\mathcal{B}$ decrypts the ct_j's and obtains $Q_S + 1$ corresponding pairs of short vectors $(\mathbf{x}_j, \mathbf{y}_j)$. If all μ_j's have been hash-queried by $\mathcal{A}$ and the vectors $(\mathbf{x}_j, \mathbf{y}_j)$ satisfy Equation (7.2) for all $j \in [Q_S + 1]$, then $\mathcal{B}$ outputs $\{(\mathbf{y}_j - \mathbf{R}\mathbf{x}_j, H(\mu_j))\}_{j \in [Q_S+1]}$. If any decryption fails or any of the above conditions is not satisfied, then $\mathcal{B}$ aborts.

First note that by the soundness of NIZK, the probability that a statement with a valid proof is false is bounded above by $2^{-\Omega(\lambda)}$. Now, since the ciphertext ct is part of the statement, we have by perfect correctness of PKE that it decrypts to the correct value. Hence, the overall probability that a decryption fails is $\leq (Q_S + 1) \cdot 2^{-\Omega(\lambda)}$. Next, we claim that for each μ_j, adversary $\mathcal{A}$ must have issued a corresponding hash query to $\mathcal{B}$. This is because otherwise, there is only a q^{-n} probability that a fresh $H(\mu_j)$ is equal to $\mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j$. Additionally, by the soundness of NIZK, it holds that for all $j \in [Q_S + 1]$:

$$\|\mathbf{x}_j\| \leq \beta/m \ \wedge \ \|\mathbf{y}_j\| \leq \beta \ \wedge \ \mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j = H(\mu_j).$$

Observe that because of the way hash queries are answered by $\mathcal{B}$, the value $H(\mu_j)$ is one of the syndromes returned by C. Define $\mathbf{t}_j = H(\mu_j)$. Then we get, for all $j \in [Q_S + 1]$,

$$\mathbf{t}_j = \mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j = \mathbf{C}\mathbf{y}_j - \mathbf{C}\mathbf{R}\mathbf{x}_j = \mathbf{C}(\mathbf{y}_j - \mathbf{R}\mathbf{x}_j).$$

Since $\mathbf{R}$ is a binary matrix, we have $\|\mathbf{y}_j - \mathbf{R}\mathbf{x}_j\| \leq 2\beta$ for all j.

Note that $\mathcal{B}$ issues one preimage query for each signing query from $\mathcal{A}$. Since $\mathcal{A}$ can issue at most Q_S signing queries, algorithm $\mathcal{B}$ also issues at most Q_S preimage queries to C. Hence $\mathcal{B}$ is a valid adversary in the one-more-ISIS game. ■

■

Next we show that the construction satisfies honest signer blindness.

7.6.4 Blindness

Theorem 7.5. *Assume that* NIZK *is zero-knowledge. Then if there exists a signer* $\mathcal{S}^*$ *in the random oracle model that wins the honest signer blindness game for the blind signature scheme in Section 7.6.2 with advantage* δ*, then there exists an adversary* $\mathcal{B}$*, with essentially the same runtime as* $\mathcal{S}^*$*, that wins the* IND-CPA *security game for* PKE *with advantage at least* $\delta/2 - 2^{-\Omega(\lambda)}$.

Proof. We argue blindness using following hybrids.

Hybrid_0 : This is the genuine honest signer blindness experiment.

Hybrid_1 : This hybrid differs from the previous one in the way the proofs π_0 and π_1 are computed: instead of genuinely computing the NIZKs, the challenger simulates them without using the witnesses.

Hybrid_2 : This hybrid differs from the previous hybrid in that both ct_0 and ct_1 encrypt $\mathbf{0}$ instead of $(\mathbf{x}_0\|\mathbf{y}_0)$ and $(\mathbf{x}_1\|\mathbf{y}_1)$, respectively.

Hybrid_3 : This hybrid differs from the previous hybrid in the way the challenger computes $\mathbf{t}_0$ and $\mathbf{t}_1$. Instead of sampling $\mathbf{x}_0$ (resp. $\mathbf{x}_1$) and computing $\mathbf{t}_0 = \mathbf{A}\mathbf{x}_0 + H(\mu_b)$ (resp. $\mathbf{t}_1 = \mathbf{A}\mathbf{x}_1 + H(\mu_{\bar{b}})$), it samples $\mathbf{u}_0$ (resp. $\mathbf{u}_1$) uniformly and sets $\mathbf{t}_0 = \mathbf{u}_0 + H(\mu_b)$ (resp. $\mathbf{t}_1 = \mathbf{u}_1 + H(\mu_{\bar{b}})$).

Indistinguishability of hybrids

In the following, we let $\mathsf{Adv}_i^{\mathsf{bl}}$ represent the advantage of $\mathcal{S}^*$ in the honest signer blindness game in Hybrid_i. Then $\mathsf{Adv}_0^{\mathsf{bl}} = \delta$.

1. The only difference between Hybrid_0 and Hybrid_1 is in the way π_0 and π_1 are computed. The two hybrids are indistinguishable because of the zero-knowledge property of the NIZK. Thus $\mathsf{Adv}_1^{\mathsf{bl}} \geq \mathsf{Adv}_0^{\mathsf{bl}} - 2^{-\Omega(\lambda)} = \delta - 2^{-\Omega(\lambda)}$.

2. The only difference between Hybrid_1 and Hybrid_2 is in the messages being encrypted by ct_0 and ct_1. The two hybrids are computationally indistinguishable because of the IND-CPA security of PKE. In particular, if the advantage of $\mathcal{S}^*$ in the honest signer blindness game in Hybrid_2 is $\mathsf{Adv}_2^{\mathsf{bl}}$, then there exists an adversary $\mathcal{B}$ against IND-CPA security of PKE with advantage $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$ such that $2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} \geq \mathsf{Adv}_1^{\mathsf{bl}} - \mathsf{Adv}_2^{\mathsf{bl}} \geq \delta - 2^{-\Omega(\lambda)} - \mathsf{Adv}_2^{\mathsf{bl}}$. (Here, we consider twice of $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$ since both ct_0 and ct_1 are replaced with encryptions of $\mathbf{0}$ and hence the IND-CPA security of PKE is called twice.).

3. The only difference between Hybrid_2 and Hybrid_3 is in the choice of the masking term for $H(\mathsf{g})$. Since the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ are only used in the computations of the vectors $\mathbf{t}_0$ and $\mathbf{t}_1$, we have by the leftover hash lemma[2] (Lemmas 2.3 and 2.1), that $\mathbf{A}\mathbf{x}_0$ and $\mathbf{A}\mathbf{x}_1$ are statistically indistinguishable from uniform $\mathbf{u}_0$ and $\mathbf{u}_1$. Hence, Hybrid_2 and Hybrid_3 are statistically indistinguishable. More concretely, we have $\mathsf{Adv}_3^{\mathsf{bl}} \geq \mathsf{Adv}_2^{\mathsf{bl}} - 2^{-\Omega(\lambda)} \geq \delta - 2^{-\Omega(\lambda)} - 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} - 2^{-\Omega(\lambda)}$.

However, in Hybrid_3, the adversary $\mathcal{S}^*$ has zero advantage in guessing the bit b since it is information theoretically hidden. Hence $\delta - 2^{-\Omega(\lambda)} - 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} - 2^{-\Omega(\lambda)} \leq 0$, which is equivalent to $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} \geq \delta/2 - 2^{-\Omega(\lambda)}$. ■

Full-Fledged Blindness. Similarly to the construction in Section 7.5, the security proof above can be extended to handle full-fledged blindness if we can ensure that PKE.pk has been honestly generated by the adversarial signer, without a corresponding decryption key and that the matrix $\mathbf{A}$ is uniform. By choosing a suitable encryption scheme so that PKE.pk is computationally indistinguishable to uniform, one can set PKE.pk as the output of a random oracle on a publicly-known value. To ensure $\mathbf{A}$ is uniform, it can similarly be set as the output of a random oracle on a publicly known value. Please see Appendix 7.A for more details.

[2]We observe that in place of LHL, we can also use LWE in this step, by letting $\mathbf{A} = (\mathbf{A}'\|\mathbf{I})$ and $\mathbf{x}^\mathsf{T} = (\mathbf{x}'^\mathsf{T}\|\mathbf{e}^\mathsf{T})$. This would change statistical closeness to computational indistinguishability. We use this variant in the concrete instantiation described below.

7.6.5 Concrete Instantiation

The goal of this section is to describe a concrete instantiation of the scheme from Section 7.6.2, and analyze the size of the resulting signature. We rely on the following building blocks:

- for the hash function, we use SHA-3-256;
- for the trapdoor generation $\mathsf{TrapGen}$ and preimage sampling $\mathsf{SamplePre}$ algorithms, we follow Falcon-512 [FHK+];
- for the $\mathsf{IND\text{-}CPA}$ secure PKE, we use an instantiation of the scheme from [LPS10] under the Module-LWE assumption, which may be viewed as a simplification of CRYSTALS-Kyber [ABD+17];
- for the NIZK scheme, we follow the protocol from [LNP22b, Figure 10].

Since it is the NIZK scheme that makes most of the signature size as well as the cost to generate it, we are mainly interested in making the generation of the proof π efficient, while potentially sacrificing the efficiency of the other components.

Choosing the moduli. For compatibility with [LNPS21], the modulus of the ZK proof is chosen as a product of primes that are congruent to 5 modulo 8. Further, we require these primes to be above $2^{12.8}$, to avoid too many soundness-boosting repetitions. The smallest such prime is 7213. Concretely, we set the preimage sampling modulus to $q_{\mathrm{F}} = 7213$, the PKE modulus to $q_{\mathsf{PKE}} = 7213^2$ and the ZK modulus to $q_{\mathrm{zk}} = 7213^2 \cdot 123637$. We chose q_{zk} as a multiple of q_{F} and q_{PKE} to simplify the linear relations to be proven (see [LNP22b, Section 6.3]).

Trapdoor generation and preimage sampling. We instantiate Falcon-512 over the ring $\mathcal{R}_{512} = \mathbb{Z}[x]/(x^{512}+1)$, where the computations are taken modulo prime $q_{\mathrm{F}} = 7213$. It allows us to build our $\mathsf{TrapGen}$ algorithm as [FHK+, Algorithm 5] that generates an NTRU secret key as the trapdoor, and to use Klein's sampler [Kle00] (also known as the GPV sampler [GPV08]) for our $\mathsf{SamplePre}$ algorithm. Our modulus q_{F} slightly

differs from the one proposed in [FHK+], since the zero-knowledge proof construction we use requires $x^{128} + 1$ to have only few (two in our case) factors modulo q_F. Note that our modulus is a little smaller than Falcon's (12289): as discussed in [ETWY22], moduli in this range have limited impact on the security. Also, this modulus change does not significantly impact the efficiency of the Falcon-512 preimage sampler [FHK+, Algorithm 10], as the modulus plays a limited role in it.

The TrapGen routine generates an NTRU secret key $\mathbf{f}, \mathbf{g} \in \mathcal{R}_{512}$, with coefficients of each polynomial $\mathbf{f}$ and $\mathbf{g}$ taken from $\mathcal{D}_{\mathbb{Z},1.17\sqrt{q_F/(2\cdot 512)}}$ (see [FHK+, Algorithm 5]), and builds up a short basis for the corresponding NTRU lattice (as in [FHK+, Algorithm 5]). The public key is $\mathbf{h} = \mathbf{g}/\mathbf{f} \bmod q_F$, defining the NTRU lattice $\{\mathbf{y} = (\mathbf{y}_1 \| \mathbf{y}_2) \in \mathcal{R}_{512}^2 : \mathbf{h} \cdot \mathbf{y}_1 + \mathbf{y}_2 = \mathbf{0} \bmod q_F\}$. The short basis enables a SamplePre routine that, given on input $\mathbf{t}$, outputs a preimage $\mathbf{y} = (\mathbf{y}_1 \| \mathbf{y}_2)$ such that $\|\mathbf{y}\| < 1.1 \cdot \sqrt{2 \cdot 512} \cdot \sigma_F$, where $\sigma_F = \frac{1.17}{\pi}\sqrt{q_F \cdot \log(4 \cdot 512 \cdot (1 + \sqrt{128 \cdot 2^{64}}))/2}$ (see [FHK+, Eq. (2.13-2.14)]). For this instantiation, the relation "$\mathbf{Cy} = \mathbf{t}$" from the construction in Section 7.6.2 translates into

$$\mathbf{h} \cdot \mathbf{y}_1 + \mathbf{y}_2 = \mathbf{t} \bmod q_F. \tag{7.3}$$

Another minor difference with Falcon-512 is that we perform rejection sampling to guarantee that $\mathbf{y}_1$ has infinity norm below a prescribed bound. Concretely, this bound is set to $\lceil 4.15 \cdot \sigma_F \rceil$ so that this acceptance probability is ≥ 0.52. This is to ensure that $\mathbf{y}_1$ always belongs to the plaintext space of the encryption scheme described below.

With all the above, we now have concrete values for the parameters n, m, q, β: $n = 512$, $m = 1024$, $q = q_F$, $\beta = 1.1 \cdot \sqrt{2 \cdot 512} \cdot \sigma_F$. Going forward, the transcript of the blind signature scheme will consist of $\mathbf{t}$ and $\mathbf{y}_1$ (note that $\mathbf{y}_2$ can be recovered as $\mathbf{t} - \mathbf{h} \cdot \mathbf{y}_1$). Using the figures above, we obtain a transcript size of 1.37KB.

Blinding the message. We now explain how we instantiate Step (1) of the signing protocol in the construction in Section 7.6.2 We sample $\mathbf{h}' \in \mathcal{R}_{512}$ uniformly modulo q_F: the vector $(\mathbf{h}'\|1)$ plays the role of $\mathbf{A}$ from the construction in Section 7.6.2. We then choose $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}_{512}$ with coefficients bounded in ℓ_∞-norm and set

$$\mathbf{t} = \mathbf{h}'\mathbf{x}_1 + \mathbf{x}_2 + H(\mu) \bmod q_F. \tag{7.4}$$

In particular, we choose $\|(\mathbf{x}_1\|\mathbf{x}_2)\|_\infty \leq 2$, and use the Ring-LWE assumption to argue the computational indistinguishability of $\mathbf{t}$ from uniform (as opposed to a statistical argument as in the proof of Theorem 7.5).

Important for our construction is the ability to transform mod-q_F linear relations defined over the ring $\mathcal{R}_{512}$ to mod-q_F linear relations defined over $\mathcal{R}_{128}$. Following [LNPS21, Section 2.8] we can map one linear relation from $\mathcal{R}_{512}$ to 4 linear relations from $\mathcal{R}_{128}$, thus Eqs. (7.3) and (7.4) can both be viewed as 4 relations over $\mathcal{R}_{128}$. This will become relevant in the zero-knowledge proof.

IND-CPA secure PKE We use $\mathcal{R}_{128} = \mathbb{Z}[x]/(x^{128} + 1)$ as underlying ring, by compatibility with the proof system (though we could have kept Kyber's $\mathbb{Z}[x]/(x^{256} + 1)$ and viewed it as an extension of $\mathcal{R}_{128}$). We let $\mathcal{S}_\tau$ denote the set of elements from $\mathcal{R}_{128}$ with ℓ_∞-norm $\leq \tau$. The rank of the plaintext space (12) is 3 times the Falcon dimension, as we will encrypt $\mathbf{m} = (\mathbf{y}_1\|\mathbf{x}_1\|\mathbf{x}_2)$: note that we do not encrypt $\mathbf{y}_2$ as it can be recovered from $\mathbf{m}$ and $H(\mu)$ by using Eqs. (7.3) and (7.4). The ℓ_∞-norm bound on all small variables ($\tau = 3$) and the Module-LWE rank (8) are set to obtain a sufficiently high hardness.

In Figure 7.1, all computations are performed modulo $q_{\mathsf{PKE}} = 7213^2$, which is set high enough to guarantee correctness. Note that we rely neither on ciphertext compression nor on the binomial distribution as in CRYSTALS-Kyber, for the sake of simplicity. With these parameters, the ciphertext occupies 8.01KB.

Setup

- τ: ℓ_∞-norm bound on all short elements in the scheme
- q_{PKE}: a modulus
- $p < q_{\mathsf{PKE}}$: a positive integer

KeyGen()

- Sample $\mathbf{A}_1 \in \mathcal{R}_{128}^{8\times 8}$ uniform modulo q_{PKE}
- Sample $\mathbf{S}_1, \mathbf{S}_2 \leftarrow U(\mathcal{S}_\tau^{12\times 8})$
- Compute $\mathbf{A}_2 = \mathbf{S}_1 \cdot \mathbf{A}_1 + \mathbf{S}_2$
- Set $\mathsf{pk} = (\mathbf{A}_1, \mathbf{A}_2)$ and $\mathsf{sk} = \mathbf{S}_1$

Enc($\mathsf{pk} = (\mathbf{A}_1, \mathbf{A}_2)$, $\mathbf{m} \in \mathcal{R}_{128}^{12}$)

- Sample $\mathbf{s}, \mathbf{e}_1 \leftarrow U(\mathcal{S}_\tau^8)$, and $\mathbf{e}_2 \leftarrow U(\mathcal{S}_\tau^{12})$
- Compute $\mathbf{c}_1 = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{e}_1$
- Compute $\mathbf{c}_2 = \mathbf{A}_2 \cdot \mathbf{s} + \mathbf{e}_2 + p \cdot \mathbf{m}$
- Return $\mathsf{ct} = (\mathbf{c}_1, \mathbf{c}_2)$

Dec($\mathsf{sk} = \mathbf{S}_1$, $\mathsf{ct} = (\mathbf{c}_1, \mathbf{c}_2)$)

- Compute $\mathbf{t} = \mathbf{c}_2 - \mathbf{S}_1 \cdot \mathbf{c}_1$
- Return $(\mathbf{t} - \mathbf{t} \bmod p)/p$

Figure 7.1: Instantiation of PKE for the construction of blind signature in Section 7.6.2

The correctness follows from the fact that for a properly formed ciphertext $\mathsf{ct} = (\mathbf{c}_1, \mathbf{c}_2)$, we have $\mathbf{t} = \mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1 + p\mathbf{m} \bmod q_{\mathsf{PKE}}$. For well-chosen parameters, this is $< q_{\mathsf{PKE}}/2$, and we recover $\mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1 + p\mathbf{m} \bmod q_{\mathsf{PKE}}$ over the integers. To recover $\mathbf{m}$, it suffices to take the quotient modulo p (provided that $\mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1$ is sufficiently small). Overall, for the decryption to be (perfectly) correct we require that

(I) $\|\mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1 + p\mathbf{m}\|_\infty < q_{\mathsf{PKE}}/2$, so that $\mathbf{c}_2 - \mathbf{S}_1 \cdot \mathbf{c}_1$ is not scrambled in the first step of the decryption algorithm;

(II) $\|\mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1\|_\infty < p/2$, so that $\mathbf{S}_2 \cdot \mathbf{s} + \mathbf{e}_2 - \mathbf{S}_1 \cdot \mathbf{e}_1$ is not scrambled in the

second step of the decryption algorithm.

These requirements should hold for all $\mathbf{S}_1, \mathbf{S}_2$ sampled during key generation, and for all $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{m}$ as small as guaranteed by the zero-knowledge proof. Note that the latter is more demanding than requesting it for $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{m}$ as small as honestly generated, because the proof is for the ℓ_2-norm (rather than ℓ_∞-norm) and it batches several norm bounds together to reduce the number of proved norm bounds, at the expense of a constant factor increase in norm bound. The above conditions are satisfied for $p = 49126$.

Zero-knowledge proof We instantiate the zero-knowledge proof using [LNP22b, Figure 10]. We need to prove knowledge of $\mathbf{y} = (\mathbf{y}_1 \| \mathbf{y}_2)$ and $\mathbf{x} = (\mathbf{x}_1 \| \mathbf{x}_2)$ with $\mathbf{y}_1, \mathbf{y}_2, \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}_{512} \cong \mathcal{R}_{128}^4$ with small norms, such that (combining Eqs. (7.3) and (7.4)):

$$\mathbf{h} \cdot \mathbf{y}_1 - \mathbf{h}' \cdot \mathbf{x}_1 + \mathbf{y}_2 - \mathbf{x}_2 = H(\mu) \bmod q_{\mathrm{F}}. \tag{7.5}$$

We also need to prove the well-formedness of ct, i.e., the existence of $\mathbf{s}, \mathbf{e}_1 \in \mathcal{R}_{128}^8$ and $\mathbf{e}_2 \in \mathcal{R}_{128}^{12}$ that are small and satisfy the relations of the **Enc** algorithm from Figure 7.1 (modulo q_{PKE}) for the message $\mathbf{m} = (\mathbf{y}_1 \| \mathbf{x}_1 \| \mathbf{x}_2)$.

We commit to the vector $(\mathbf{y}_1 \| \mathbf{y}_2 \| \mathbf{x}_1 \| \mathbf{x}_2 \| \mathbf{s} \| \mathbf{e}_1 \| \mathbf{e}_2)$, prove two linear relations involving this vector (one coming from Eq. (7.5) and the other from the encryption), and prove three ℓ_2-norm bounds:

(I) $\|(\mathbf{y}_1 \| \mathbf{y}_2)\| \leq \beta$,

(II) $\|(\mathbf{x}_1 \| \mathbf{x}_2)\| \leq \sqrt{2 \cdot 512} \cdot 2$,

(III) $\|(\mathbf{s} \| \mathbf{e}_1 \| \mathbf{e}_2)\| \leq \tau \cdot \sqrt{(8 + 8 + 12) \cdot 128}$.

We shall not repeat the steps of the zero-knowledge proof from [LNP22b], but instead make a guideline on how to instantiate the protocol in [LNP22b, Figure 10]. The reader is advised to follow it using Table 7.1. Note that the two linear relations we need to prove incur negligible additional cost in terms of size, see [LNP22b, Figure 4].

variable	description	instantiation
ρ	# of quadratic eqs.	0
ρ_{eval}	# of evaluations with const. coeff. 0	0
v_e	# exact norm proofs	3
v_d	# non-exact norm proofs	0
$\mathbf{s}_1$	committed message in the Ajtai part	$(\mathbf{y}\|\|\mathbf{x}\|\|\mathbf{s}\|\|\mathbf{e}_1\|\|\mathbf{e}_2)$
$\mathbf{m}$	committed message in the BDLOP part	$\emptyset$
$\mathbf{s}$	$(\mathbf{s}_1\|\|\mathbf{m})$	$(\mathbf{y}\|\|\mathbf{x}\|\|\mathbf{s}\|\|\mathbf{e}_1\|\|\mathbf{e}_2)$
$\mathbf{E}_1$	public matrix proving that $\|\|\mathbf{E}_1\mathbf{s} - \mathbf{v}_1\|\| \leq \beta_1$	$\begin{pmatrix} \mathbf{Id}_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{Id}_4 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
β_1	upper bound on $\|\|\mathbf{E}_1\mathbf{s} - \mathbf{v}_1\|\|$	β
$\mathbf{E}_2$	public matrix proving that $\|\|\mathbf{E}_2\mathbf{s} - \mathbf{v}_2\|\| \leq \beta_2$	$\begin{pmatrix} 0 & 0 & \mathbf{Id}_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{Id}_4 & 0 & 0 & 0 \end{pmatrix}$
β_2	upper bound on $\|\|\mathbf{E}_2\mathbf{s} - \mathbf{v}_2\|\|$	$\sqrt{2 \cdot 512} \cdot 2$
$\mathbf{E}_3$	public matrix proving that $\|\|\mathbf{E}_3\mathbf{s} - \mathbf{v}_3\|\| \leq \beta_3$	$\begin{pmatrix} 0 & 0 & 0 & 0 & \mathbf{Id}_8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{Id}_8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{Id}_{12} \end{pmatrix}$
β_3	upper bound on $\|\|\mathbf{E}_3\mathbf{s} - \mathbf{v}_3\|\|$	$\tau \cdot \sqrt{28 \cdot 128}$
$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	public vectors proving that $\|\|\mathbf{E}_i\mathbf{s} - \mathbf{v}_i\|\| \leq \beta_i$	$\mathbf{0}, \mathbf{0}, \mathbf{0}$
$\|\|\mathbf{x}\|\|$	norm of $((\beta_i^{(e)})^2 - \|\|\mathbf{E}_i\mathbf{s} - \mathbf{v}_i\|\|^2)_i$	$\sqrt{3 \cdot 128}$
p_1, p_2, p_3	number of rows of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$	8, 8, 28
$c^{(e)}$	$128 \cdot \sum_i (p_i + 1)$	6016
$\alpha^{(e)}$	upper bound on $\|\|(\mathbf{E}_1\mathbf{s} - \mathbf{v}_1\|\| \ldots \|\|\mathbf{E}_3\mathbf{s} - \mathbf{v}_3\|\|\mathbf{x})\|\|$	5840

Table 7.1: Instantiation of the protocol from [LNP22b, Figure 10]. The left-most column 'variable' and the middle column 'description' refer to the notations from [LNP22b, Figure 10], the right-most column 'instantiation' refers to our notations.

For concrete parameter selection, we refer the reader to Table 7.2. We instantiate the variables κ, l, η, ν as in [LNP22b].

The parameters $\gamma_1, \gamma_2, \gamma_e$ are rejection sampling parameters. They are set so that the expected number of rejections before producing a valid signature is small. Using [LNP22b, Section 6.1], we obtain that the expected number is $\approx 2\exp(\frac{14}{\gamma_1} + \frac{1}{2\gamma_1^2} + \frac{1}{2\gamma_2^2} + \frac{1}{2\gamma_e^2}) \approx 10.4$.

The parameter $m_1 = 4 \cdot 4 + 2 \cdot 8 + 12 + 3 = 47$ counts the length of the committed message as a vector over $\mathcal{R}_{128}$ (adding 3 as we have three norm equations). The parameters n and m_2 are chosen such that the Module-SIS and Module-LWE problems that underlie the zero-knowledge protocol are sufficiently hard.

Following the compression technique of [DKL+18], we can reduce the proof size by cutting low-order bits of the commitment. There are two variables responsible for this cut: γ and D. To choose these variables we follow the approach from [LNP22b, Section 6.1].

With these parameters, the proof π has size 37.18KB, and the overall signature (including π and ct) has size 45.19KB. (Recall the transcript has size 1.37KB.) This is for classical core-SVP hardness of 109 bits. Below, we give precise figures for our security estimates. These estimates as well as the sizes of signature components can be verified via a script available at https://gitlab.com/ElenaKirshanova/onemoresis_estimates. For concrete estimates of ModuleLWE and ModuleSIS assumptions we rely on the work from [ABD+17], and for the NTRU assumption security – on the work from [DvW21]. To compute the proof size π, we adapt the strategy from [LNP22b, Section 6.1] to our setting with the exception that we take the estimates on the entropy of a discrete Gaussian variable from [ESLR22]. This choice is inline with the recent work [ETWY22, Section 5].

Security Let us summarize our security assumptions and their corresponding bit security levels.

variable	value	variable	value	variable	value
q_{zk}	6432507821053	κ	2	n	11
l	2	λ	10	m_1	47
γ_1	10	η	59	m_2	34
γ_2	1.5	ν	1	ℓ	0
γ_e	5	D	16	γ	2^{24}

Table 7.2: Concrete parameter selection for the zero-knowledge protocol from [LNP22b, Figure 10]. The columns 'variable' refer to the notations from [LNP22b].

1. The security of Falcon's signature scheme relies on two assumptions:

   (I) Key recovery security relies on the NTRU assumption (i.e., it is hard to recover $\mathbf{f}, \mathbf{g}$ from $\mathbf{h} = \mathbf{g}/\mathbf{h}$). The estimator from [DvW21] states that this has core-SVP hardness of 135 bits.

   (II) Forgery security relies on Module-SIS hardness. To estimate it, we use the Dilithium script [DKL+18], which states that this has core-SVP hardness of 129 bits.

2. Given $\mathbf{h}'$, we argue that $\mathbf{h}'\mathbf{x}_1 + \mathbf{x}_2$ hides $\mathbf{x}_1, \mathbf{x}_2$ under the Module-LWE assumption. Again, we use the script from [DKL+18], which states that this has core-SVP hardness of 122 bits.

3. The security of the encryption scheme (for both the secret key and the ciphertext) relies on the Module-LWE assumption. Here we reach core-SVP hardness of 120 bits.

4. The zero-knowledge proof relies on the Module-SIS and Module-LWE assumptions (technically, the construction of [LNP22b] relies on the so-called Extended-Module-LWE, whose hardness is conjectured to be the same as plain Module-LWE). For both Module-SIS and Module-LWE, we obtain 109 bits of core-SVP hardness.

5. Finally, we estimate the hardness of solving one-more-ISIS with the norm bound to be the norm of the extracted solution. For this, we assume that $\mathbf{h}'$ is set as $\mathbf{x}'_1 \cdot \mathbf{h} + \mathbf{x}'_2$ in the unforgeability proof instead of "$\mathbf{A} = \mathbf{CR}$" (see the proof of Theorem 7.3), using the same Module-LWE assumption as we did to argue computational indistinguishability from uniform of $\mathbf{x}_1 \cdot \mathbf{h}' + \mathbf{x}_2$, i.e., with $\|\mathbf{x}'_1\|_\infty, \|\mathbf{x}'_2\|_\infty \leq 2$. The extracted solution is $(\mathbf{y}_1 - \mathbf{x}_1\mathbf{x}'_1 \| \mathbf{y}_2 - \mathbf{x}_1\mathbf{x}'_2 - \mathbf{x}_2)$, which has ℓ_2-norm $\leq \sqrt{\beta^2 + 2^4 \cdot 512 \cdot 4 + 2^2 \cdot 512 \cdot 2}$. Having $\mathbf{h}$, the hardness of finding a preimage of such norm is again a Module-SIS instance, which we estimate to be at 109 bits of core-SVP hardness. The one-more-ISIS attacks described in the next section all have higher costs.

7.6.6 Security Analysis of One-More-ISIS

The purpose of this section is to argue why we believe that the new computational problem we introduce, one-more-ISIS, is hard. We did not succeed in obtaining a reduction from a well-studied problem to one-more-ISIS, but we still expect that for the parameter ranges relevant to our constructions, this problem cannot be solved by polynomial or even sub-exponential time attackers.

The hardness of the one-more-ISIS problem as stated in Definition 7.6 primarily depends on the precise relation between β, the upper bound on the norm of the vectors $\mathbf{y}_i$'s the adversary must output, and the dimensions m and n of the input matrix $\mathbf{C}$. We also assume that σ – the standard deviation parameter of the preimage queries – is of order $\Omega(\sqrt{m})$, which what we would expect from an efficient sampler, e.g. [GPV08]. Note that a significantly smaller standard deviation, e.g., of order $O(1)$, would invalidate the hardness of the one-more-ISIS assumption as extremely short $\mathbf{y}$'s would enable an adversary to solve one-more-ISIS (see the discussion below). In this section we make the hardness of the one-more-ISIS problem explicit by describing the parameter regimes for which this problem can be solved in polynomial time, and for which, as far as we know, the problem is exponentially hard. We consider two approaches to solve one-more-ISIS: combinatorial attacks and lattice-based attacks.

Combinatorial attacks. We start by showing an elementary polynomial time algorithm that achieves $\beta = \Theta(\sqrt{mn}\sigma)$ and requires $(q \cdot n)$ ISIS preimage oracle calls.

Consider the set of n-dimensional vectors $A = \{\mathbf{e}_i \cdot a : \ i \in [n], a \in \mathbb{Z}_q\}$, where the $\mathbf{e}_i$'s are the canonical-basis vectors. The set A is of size $q \cdot n$. The adversary runs preimage queries for all vectors from A and receives Gaussian vectors $\mathbf{y}'$'s. Thanks to the Gaussian tail bound (see Lemma 2.1), we have $\|\mathbf{y}'\| \leq 2\sqrt{m}\sigma$ with probability greater than $1 - 2^{-m}$ for all $\mathbf{y}'$'s. Any element from $\mathbb{Z}_q^n$, and thus the challenge $\mathbf{t}$, can be expressed as a sum of at most n vectors from A (one for each coordinate). The adversary then sums the

corresponding $\mathbf{y}'$'s it received from the ISIS preimage oracle and obtains a new $\mathbf{y}$ such that $\mathbf{C}\mathbf{y} = \mathbf{t}$. The resulting $\mathbf{y}$ is a valid one-more-ISIS solution for $\beta = \Theta(\sqrt{nm} \cdot \sigma)$ with probability $1 - 2^{-\Omega(m)}$.

The algorithm can be generalized to a larger set A. The generalization, presented in Figure 7.2, makes the attack less efficient, but reduces the bound on β. It is parametrized by Q, the upper bound on the number of the preimage queries the attacker can issue. This is also the assumed upper bound on the memory capacity of the attacker, since the attack requires that all the responses are stored.

Input: The ISIS preimage oracle $O^{\mathsf{ISIS}}(\cdot)$, a number Q of queries to O^{ISIS}, and $\mathbf{t} \in \mathbb{Z}_q^n$.

Output: A short vector $\mathbf{y} \in \mathbb{Z}_q^m$ such that $\mathbf{C}\mathbf{y} = \mathbf{t} \bmod q$.

1. Set $w = \lfloor \frac{\log(Q/n^2)}{\log q} \rfloor$.
2. Let $A = \left\{ \sum_{\substack{w\cdot(i-1)<j \\ \leq \max\{w\cdot i, n\}}} \mathbf{e}_j \cdot a_j : \forall i \in \left[\left\lceil \frac{n}{w} \right\rceil\right], a_j \in \mathbb{Z}_q \right\}$.
3. For all $\mathbf{a} \in A$, set $T[\mathbf{a}] = O^{\mathsf{ISIS}}(\mathbf{a})$.
4. Write $\mathbf{t} = \mathbf{a}_{i_1} + \ldots + \mathbf{a}_{i_{\lceil n/w \rceil}}$.
5. Output $\mathbf{y} = T[\mathbf{a}_{i_1}] + \ldots + T[\mathbf{a}_{i_{\lceil n/w \rceil}}]$.

Figure 7.2: Combinatorial Attack on one-more-ISIS.

The correctness of the algorithm in Figure 7.2 is direct: any $\mathbf{t} \in \mathbb{Z}_q^n$ can be efficiently written as a sum of at most $\lceil n/w \rceil$ elements from the set A constructed on Step 2. Note that $|A| \leq n^2 q^w$: by definition of w, the algorithm makes $\leq Q$ queries. Finally, we can bound the norm of the output as $\|\mathbf{y}\| < 2\sqrt{\lfloor \frac{n}{w} \rfloor \cdot m} \cdot \sigma = \Theta\left(\sqrt{1 + \frac{n \log q}{\log(Q/n^2)}} \cdot \sqrt{m} \cdot \sigma\right)$, with probability greater than $1 - 2^{-\Omega(m)}$. The algorithm is correct for any $1 \leq w \leq n$ computed on Step 1, providing a trade-off between the runtime (which is essentially the number Q of preimage queries) and the bound on β.

Lattice-based attacks. A strategy to attack one-more-ISIS is to use a discrete Gaussian sampler algorithm [Kle00; GPV08]. This allows to solve one-more-ISIS in poly(m) time with $\beta = \Omega(m\sigma)$ using $O(m^2)$ preimage queries. More precisely, the attacker performs the following:

1. Given $\mathbf{C}$, compute a basis of $\Lambda_q^\perp(C)$.
2. Query the preimage ISIS oracle $\Theta(m^2)$ times for $\mathbf{t} = \mathbf{0}$. From the oracle's answers and from the basis of $\Lambda_q^\perp(C)$ constructed in the previous step, compute a basis $\mathbf{B}$ for $\Lambda_q^\perp(\mathbf{C}) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{C}\mathbf{y} = \mathbf{0} \bmod q\}$ such that the norms of Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}}$ of $\mathbf{B}$ are bounded from above.
3. Given an input $\mathbf{t} \in \mathbb{Z}_q^n$ find any $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{C}\mathbf{z} = \mathbf{t}$.
4. Run Babai's Nearest Plane algorithm [Bab85] on input $(\mathbf{B}, \mathbf{z})$. Let $\mathbf{v} \in \Lambda_q^\perp(C)$ be the output. Return $\mathbf{z} - \mathbf{v}$ as a one-more-ISIS solution for $\mathbf{t}$.

Let us analyse the quality of the vectors returned by the above procedure. First, thanks to standard properties of lattice Gaussian distributions, it indeed suffices to query the ISIS preimage oracle $\Theta(m^2)$ times in Step 1, in order to obtain a set of m linearly independent vectors from $\Lambda_q^\perp(\mathbf{C})$ with at least constant probability bounded away from 0 (see [Reg09, Corollary 3.16]). According to [BF11, Proposition 4.7.], these linearly independent vectors will be of norm bounded from above and below by $\Theta(\sqrt{m}\sigma)$. Out of this set, we can efficiently extract m linearly independent vectors by checking which ones form a subspace of the desired rank. Using [MG02, Lemma 7.1] we can convert this set to a basis $\mathbf{B}$, such that $\|\widetilde{\mathbf{B}}\| < \sqrt{m}\sigma$. Finally, Babai's Nearest Plane algorithm in Step 4 outputs a vector $\mathbf{v}$ such that $\|\mathbf{v} - \mathbf{z}\| \leq \frac{1}{2}(\sum_{i\in[m]} \|\widetilde{\mathbf{b}}_i\|^2)^{1/2}$, where the right-hand side of the inequality is bounded from above by $m\sigma$ with probability greater than $1 - 2^{-\Omega(m)}$. Furthermore, the returned vector $\mathbf{e} = \mathbf{z} - \mathbf{v}$ satisfies $\mathbf{C}\mathbf{e} = \mathbf{C}\mathbf{z} - \mathbf{C}\mathbf{v} = \mathbf{t}$ as $\mathbf{C}\mathbf{v} = 0$, hence giving a one-more-ISIS solution for $\beta = O(m\sigma)$. As the vectors $\widetilde{\mathbf{b}}_i$ are already somewhat short thanks to the Gaussian tail bound, we do not expect a significant decay in their norms when converting them to a basis and/or applying a basis reduction algorithm, like LLL or BKZ, on $\mathbf{B}$. Hence, the norm of $\mathbf{e}$ is expected to be close to the above upper bound, resulting in the one-more-ISIS solution for $\beta = \Theta(m\sigma)$.

Another strategy to improve the above bounds on β at higher costs is to obtain a basis of the lattice $\Lambda_q^\perp(\mathbf{C})$ that is *shorter* than what the ISIS preimage oracle offers. We can go as far as the Minkowski's bound suggests, i.e., we can achieve $\|\mathbf{B}\| = \lambda_1(\Lambda_q^\perp(\mathbf{C})) \leq \min_{m' \leq m} \sqrt{m'} \cdot q^{n/m'}$ (here we assume that all lattice minima have essentially the same norms, which is expected to be the case when $\mathbf{C}$ is sampled uniformly). The latter bound is $O(\sqrt{n \ln q})$ when $m = \Omega(n \log q)$. Vectors of such a small norm can be found by calling shortest vector problem solvers on $\Lambda_q^\perp(\mathbf{C})$. The fastest known such algorithms run in time $2^{O(m)}$ (see, e.g., [BDGL16]). This exponential time attack enables us to solve one-more-ISIS for $\beta = \Theta(\sqrt{mn \ln q})$ by invoking Babai's Nearest Plane algorithm on the obtained short basis. Note that the ISIS preimage oracle is only used to obtain a basis of $\Lambda_q^\perp(\mathbf{C})$. A trade-off between the quality of β and the runtime is possible: a b-BKZ reduction [HPS11; SE94] yields a basis $\mathbf{B}$ with $\|\mathbf{B}\| \leq b^{O(m/b)} \cdot \lambda_1(\Lambda_q^\perp(\mathbf{C}))$ in time $2^{O(b)}$, thus leading to $\beta = b^{O(m/b)} \cdot \sqrt{mn \ln q}$. Note that in order to outperform the bound on β we have in the polynomial time regime, the BKZ parameter b has to be of order $\Theta(m/\log \sigma)$, when $m = \Theta(n \log q)$.

To summarize, we have the run-times for solving one-more-ISIS:

- there exists a combinatorial algorithm that achieves $\beta = \Theta\left(\sqrt{1 + \frac{n \log q}{\log(Q/n^2)}} \cdot \sqrt{m}\sigma\right)$ in time Q and using $Q \geq nq$ preimage queries;
- there exists a lattice-based algorithm that achieves $\beta = \Theta(m\sigma)$ in polynomial time using $O(m^2)$ preimage queries; except for very few queries, it is outperformed by the combinatorial algorithm;
- there exists a lattice-based algorithm that achieves $\beta = 2^{O(\frac{m \log\log T}{\log T})} \sqrt{mn \log q}$ in time T without any preimage query (except to obtain a basis of $\Lambda_q^\perp(\mathbf{C})$).

Open questions and potential directions. Let us now formulate two cryptanalytic questions that the new one-more-ISIS hardness assumptions raises.

I. Improving algorithms for the shortest vector problem with preimage queries. One might wonder whether we can accelerate existing shortest vector solvers, such as sieving algorithms [AKS01; NV08; BDGL16], once we already have a somewhat short basis. Just from the nature of sieving algorithms it does not seem to be the case: even to obtain a small constant reduction in the norm of the current shortest vector, sieving generates and processes $2^{O(m)}$ vectors which already constitutes its asymptotic cost.

II. Improving Babai's Nearest Plane with a short generating set. Given access to ISIS preimages, another direction one can consider is to try to accelerate the *closest vector problem* (CVP) solvers on $\Lambda_q^{\perp}(\mathbf{C})$, by exploiting the fact that we have many short vectors from this lattice. The presence of many short vectors helps to heuristically improve the Voronoi cell-based CVP algorithms [DLdW19]. Yet their heuristic correctness and analysis rely on the presence of the *shortest* vectors from $\Lambda_q^{\perp}(\mathbf{C})$, which, as we believe, the preimage ISIS queries do not help to obtain fast.

APPENDIX

7.A FULL-FLEDGED BLINDNESS

In this section, we provide a construction of a blind signature with full-fledged blindness. The construction differs from the one in Section 7.6.2 in the way the PKE encryption key PKE.pk is generated.

7.A.1 Construction

The construction uses the following building blocks:

1. A hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ that will be modeled as a random oracle in the unforgeability proof.

2. A NIZK for the statement of Equation (7.6)

3. A perfectly correct IND-CPA secure public key encryption scheme PKE = (PKE.KeyGen, PKE.Enc, PKE.Dec) having the property that the public key generated by PKE.KeyGen is computationally indistinguishable from a uniformly sampled value from its range.

4. A hash function $\mathsf{H}_{\mathsf{PKE}} : \{\} \rightarrow \mathcal{K}_{\mathsf{PKE}}$ that will be modeled as a random oracle in the unforgeability proof. Here, represents an empty bitstring and $\mathcal{K}_{\mathsf{PKE}}$ is the public key space of PKE.

5. A hash function $\mathsf{H}_{\mathsf{A}} : \{\} \rightarrow \mathbb{Z}_q^{n \times m}$ that will be modeled as a random oracle in the unforgeability proof.

The construction is as below:

Setup. $\mathsf{Gen}(1^\lambda)$: Upon input the security parameter λ, define $n, m, q, \sigma, \beta = \sigma\sqrt{m}$ as functions of λ such that q is prime, $\mathsf{one\text{-}more\text{-}ISIS}_{q,n,m,\sigma,2\beta}$ is hard and the scheme is both efficient and complete; then do the following:

- Set $\mathsf{PKE.pk} = \mathsf{H}_{\mathsf{PKE}}()$.
- Compute $(\mathbf{C}, \mathbf{T_C}) \leftarrow \mathsf{TrapGen}(n, m, q)$.
- Sample $\mathbf{A} = \mathsf{H}_{\mathsf{A}}()$.
- Output $\mathsf{BSig.sk} = \mathbf{T_C}$, $\mathsf{BSig.vk} = \mathbf{C}$.

Signing. $\langle \mathcal{S}(\mathsf{BSig.sk}), \mathcal{U}(\mathsf{BSig.vk}, \mathsf{g}) \rangle$:

1. **User:** Given the key BSig.vk and a message g, user $\mathcal{U}$ does the following:
   - It computes $\mathbf{A} = \mathsf{H}_{\mathsf{A}}()$.
   - It samples $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \sigma/m}$.
   - It computes $\mathbf{t} = \mathbf{Ax} + H(\mathsf{g})$.
   - It sends $\mathbf{t}$ to the signer.
2. **Signer:** Upon receiving $\mathbf{t}$, signer $\mathcal{S}$ does the following:
   - It samples a short vector $\mathbf{y} \leftarrow \mathsf{SamplePre}(\mathbf{C}, \mathbf{T_C}, \mathbf{t}, \sigma)$; we have $\mathbf{Cy} = \mathbf{t}$.
   - It sends $\mathbf{y}$ to the user.
3. **User:** Upon receiving $\mathbf{y}$, user $\mathcal{U}$ does the following:
   - It computes $\mathsf{PKE.pk} = \mathsf{H}_{\mathsf{PKE}}()$.
   - It verifies that $\|\mathbf{y}\| \leq \beta$ and satisfies $\mathbf{Cy} = \mathbf{t}$.

- It samples $\mathsf{PKE.Enc}$ randomness r and computes

$$\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r).$$

- It generates a NIZK π for following statement[3]: Given $\mathbf{C}, \mathbf{A} = \mathsf{H_A}(), \mathsf{PKE.pk} = \mathsf{H_{PKE}}(), \mathsf{ct}$ and μ, there exists r and vectors $\mathbf{x}, \mathbf{y}$ such that

$$\|\mathbf{x}\| \leq \beta/m \wedge \|\mathbf{y}\| \leq \beta \wedge \mathbf{Cy} - \mathbf{Ax} = H(\mathsf{g}) \wedge \mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{PKE.pk}, \mathbf{x}\|\mathbf{y}; r). \tag{7.6}$$

- The signature is (π, ct).

Verifying. The verifier computes $\mathsf{PKE.pk} = \mathsf{H_{PKE}}()$ and $\mathbf{A} = \mathsf{H_A}()$ and accepts if the proof π is valid, and rejects if it is not.

Parameters and Completeness The parameters setting and the completeness argument are the same as in Section 7.6.2.

Security We show that our construction satisfies one more unforgeability and blindness.

7.A.2 Unforgeability

Theorem 7.6. *Assume that* NIZK *is sound. Then if there exists an adversary* $\mathcal{A}$ *in the random oracle model that issues* Q_S *signing queries and any number of hash queries and outputs* $Q_S + 1$ *signatures with probability* δ*, then there exist algorithms* $\mathcal{B}$ *and* C*, both having essentially the same runtime as* $\mathcal{A}$*, where* $\mathcal{B}$ *requests* Q_S *preimage queries and wins the* $\mathsf{one\text{-}more\text{-}ISIS}_{q,n,m,\sigma,2\beta}$ *game with probability* $\mathsf{Adv_{omisis}}$ *and* C *distinguishes the public key of the* PKE *scheme from uniform with advantage* $\mathsf{Adv_{PKE}}$ *such that*

$$\mathsf{Adv_{PKE}} + \mathsf{Adv_{omisis}} \geq \delta - 2^{-\Omega(\lambda)} - (Q_S + 1)(2^{-\Omega(\lambda)} + q^{-n}).$$

Proof. We construct the proof using the following hybrids.

[3]Note that this is the same statement as in (7.2) in Section 7.6.2.

Hybrid_0: This is the genuine one more unforgeability experiment.

Hybrid_1 : In this hybrid, the challenger computes $\mathsf{PKE.pk}$ differently. It first runs $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and then programs $\mathsf{H}_{\mathsf{PKE}}() = \mathsf{PKE.pk}$. It stores $\mathsf{PKE.sk}$.

Hybrid_2 : In this hybrid, for every signature $\sigma_j = (\pi_j, \mathsf{ct}_j)$ output by the adversary (for $j \in [Q_S+1]$), the challenger uses $\mathsf{PKE.sk}$ to decrypt ct_j into a plaintext $(\mathbf{x}_j \| \mathbf{y}_j)$ (which can be $\perp$ in case decryption fails). It stores the $(\mathbf{x}_j \| \mathbf{y}_j)$'s.

Hybrid_3 : This hybrid differs from the previous hybrid in the way matrix $\mathbf{A}$ is computed. In this hybrid, the challenger first samples $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and then programs $\mathsf{H}_{\mathsf{A}}() = \mathbf{A}$.

Hybrid_4 : This hybrid differs from the previous one in the way matrix $\mathbf{A}$ is chosen. The challenger first samples a binary matrix $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$ and sets $\mathbf{A} = \mathbf{CR}$.

Indistinguishability of hybrids

In the following, we let $\mathsf{Adv}_i^{\mathsf{omuf}}$ represent the advantage of $\mathcal{A}$ in the one more unforgeability game in Hybrid_i. Then $\mathsf{Adv}_0^{\mathsf{omuf}} = \delta$.

1. The only difference between Hybrid_0 and Hybrid_1 is in the way $\mathsf{PKE.pk}$ is computed. Hence if $\mathcal{A}$ succeeds in the one more unforgeability game in Hybrid_1 with probability $\mathsf{Adv}_1^{\mathsf{omuf}}$, then there exists an algorithm C which runs in essentially the same time as $\mathcal{A}$ and distinguishes the public key of the PKE scheme from uniform with advantage at least $\mathsf{Adv}_0^{\mathsf{omuf}} - \mathsf{Adv}_1^{\mathsf{omuf}}$. Let us denote the advantage of C with $\mathsf{Adv}_{\mathsf{PKE}}$. Then $\mathsf{Adv}_1^{\mathsf{omuf}} \geq \mathsf{Adv}_0^{\mathsf{omuf}} - \mathsf{Adv}_{\mathsf{PKE}} = \delta - \mathsf{Adv}_{\mathsf{PKE}}$.

2. The differences between Hybrid_1 and Hybrid_2 are only concerning the inner computations of the challenger and not its interactions with the adversary. Hence, the two hybrids are identical in the view of the adversary. Thus $\mathsf{Adv}_2^{\mathsf{omuf}} = \mathsf{Adv}_1^{\mathsf{omuf}} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}}$.

3. The only difference between Hybrid_2 and Hybrid_3 is that in the latter $\mathbf{A}$ is first chosen uniformly from $\mathbb{Z}_q^{n \times m}$ and then $\mathsf{H}_{\mathsf{A}}()$ is programmed to be $\mathbf{A}$. Hence, the two hybrids are identical in the adversary's view in the random oracle model. Thus $\mathsf{Adv}_3^{\mathsf{omuf}} = \mathsf{Adv}_2^{\mathsf{omuf}} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}}$.

4. The only difference between Hybrid_3 and Hybrid_4 is that in the latter $\mathbf{A}$ is computed as $\mathbf{CR}$, where $\mathbf{R}$ is a uniform binary matrix, instead of sampling it uniformly randomly from $\mathbb{Z}_q^{n\times m}$. The two hybrids are indistinguishable because Lemmas 2.3 and 2.1 imply that $(\mathbf{C}, \mathbf{A})$ is within statistical distance $2^{-\Omega(\lambda)}$ from $(\mathbf{C}, \mathbf{CR})$. Thus $\mathsf{Adv}_4^{\mathsf{omuf}} \geq \mathsf{Adv}_3^{\mathsf{omuf}} - 2^{-\Omega(\lambda)} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}} - 2^{-\Omega(\lambda)}$.

We conclude with the following claim.

Claim 7.7. Assume that the NIZK argument system is sound and PKE is perfectly correct. Then if there is an adversary $\mathcal{A}$ in the random oracle model that makes at most Q_S signing queries and succeeds in generating $Q_S + 1$ signatures in Hybrid_4 with probability ε, then there exists a one-more-ISIS adversary $\mathcal{B}$ with Q_S preimage queries with success probability at least $\varepsilon - (Q_S + 1)(2^{-\Omega(\lambda)} + q^{-n})$.

Proof. The reduction $\mathcal{B}$ is as follows.

1. Upon being challenged by the one-more-ISIS challenger C, with matrix $\mathbf{C}$, algorithm $\mathcal{B}$ does the following:

   - It uniformly samples a binary matrix $\mathbf{R}$ and sets $\mathbf{A} = \mathbf{CR}$. It programs $\mathsf{H}_{\mathsf{A}}() = \mathbf{A}$.
   - It samples $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sets $\mathsf{H}_{\mathsf{PKE}}() = \mathsf{PKE.pk}$.
   - It invokes $\mathcal{A}$ with $\mathbf{C}$ as the verification key.

2. In response to each (fresh) hash query on input μ from $\mathcal{A}$, algorithm $\mathcal{B}$ makes a syndrome query to C. Challenger C returns a uniform vector $\mathbf{t} \in \mathbb{Z}_q^n$, which $\mathcal{B}$ forwards to $\mathcal{A}$ as $H(\mu)$.

3. To answer a signing query on input $\mathbf{t}'$, algorithm $\mathcal{B}$ forwards $\mathbf{t}'$ to C as a preimage query. Challenger C returns a short vector $\mathbf{y}'$, such that $\mathbf{Cy}' = \mathbf{t}'$. Algorithm $\mathcal{B}$ forwards $\mathbf{y}'$ to $\mathcal{A}$.

4. Eventually, adversary $\mathcal{A}$ outputs $Q_S + 1$ message-signature pairs $\{\mu_j, (\pi_j, \mathsf{ct}_j)\}_{j\in[Q_S+1]}$.

5. If the π_j's pass verification, then algorithm $\mathcal{B}$ decrypts the ct_j's and obtains $Q_S + 1$ corresponding pairs of short vectors $(\mathbf{x}_j, \mathbf{y}_j)$. If all μ_j's have been hash-queried by $\mathcal{A}$ and the vectors $(\mathbf{x}_j, \mathbf{y}_j)$ satisfy Equation (7.6) for all $j \in [Q_S + 1]$, then $\mathcal{B}$ outputs $\{(\mathbf{y}_j - \mathbf{Rx}_j, H(\mu_j))\}_{j\in[Q_S+1]}$. If any decryption fails or any of the above conditions is not satisfied, then $\mathcal{B}$ aborts.

First note that by the soundness of NIZK, the probability that a statement with a valid proof is false is bounded above by $2^{-\Omega(\lambda)}$. Now, since the ciphertext ct is part of the statement, we have by perfect correctness of PKE that it decrypts to the correct value. Hence, the overall probability that a decryption fails is $\leq (Q_S + 1) \cdot 2^{-\Omega(\lambda)}$.

Next, we claim that for each μ_j, adversary $\mathcal{A}$ must have issued a corresponding hash query to $\mathcal{B}$. This is because otherwise, there is only a q^{-n} probability that a fresh $H(\mu_j)$ is equal to $\mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j$. Additionally, by the soundness of NIZK, it holds that for all $j \in [Q_S + 1]$:

$$\|\mathbf{x}_j\| \leq \beta/m \ \wedge \ \|\mathbf{y}_j\| \leq \beta \ \wedge \ \mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j = H(\mu_j).$$

Observe that because of the way hash queries are answered by $\mathcal{B}$, the value $H(\mu_j)$ is one of the syndromes returned by C. Define $\mathbf{t}_j = H(\mu_j)$. Then we get, for all $j \in [Q_S + 1]$,

$$\mathbf{t}_j = \mathbf{C}\mathbf{y}_j - \mathbf{A}\mathbf{x}_j = \mathbf{C}\mathbf{y}_j - \mathbf{C}\mathbf{R}\mathbf{x}_j = \mathbf{C}(\mathbf{y}_j - \mathbf{R}\mathbf{x}_j).$$

Since $\mathbf{R}$ is a binary matrix, we have $\|\mathbf{y}_j - \mathbf{R}\mathbf{x}_j\| \leq 2\beta$ for all j. Thus, the success probability of $\mathcal{B}$ is at least $\varepsilon - (Q_S + 1)(2^{-\Omega(\lambda)} + q^{-n})$.

Note that $\mathcal{B}$ issues one preimage query for each signing query from $\mathcal{A}$. Since $\mathcal{A}$ can issue at most Q_S signing queries, algorithm $\mathcal{B}$ also issues at most Q_S preimage queries to C. Hence $\mathcal{B}$ is a valid adversary in the one-more-ISIS game. ■
■

7.A.3 Blindness

Theorem 7.8. *Assume that* NIZK *is zero-knowledge. Then if there exists a signer $\mathcal{S}^*$ in the random oracle model that wins the full-fledged blindness game for the blind signature scheme in Section 7.A.1 with advantage δ, then there exist adversaries $\mathcal{B}$ and C, both running in essentially the same time as $\mathcal{S}^*$, where $\mathcal{B}$ wins the* IND-CPA *security game for* PKE *with advantage* $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$ *and C distinguishes the public key of the* PKE *scheme*

from uniform with advantage $\mathsf{Adv}_{\mathsf{PKE}}$ *such that*

$$2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} + \mathsf{Adv}_{\mathsf{PKE}} \geq \delta - 2^{-\Omega(\lambda)}.$$

Proof. We argue blindness using the following hybrids.

Hybrid_0 : This is the genuine full-fledged blindness experiment.

Hybrid_1 : This hybrid differs from the previous one in the way $\mathsf{PKE.pk}$ is computed: the challenger now samples $(\mathsf{PKE.pk}, \mathsf{PKE.sk}) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$ and sets $\mathsf{H}_{\mathsf{PKE}}() = \mathsf{PKE.pk}$.

Hybrid_2 : This hybrid differs from the previous one in the way the proofs π_0 and π_1 are computed: instead of genuinely computing the NIZKs, the challenger simulates them without using the witnesses.

Hybrid_3 : This hybrid differs from the previous hybrid in that both ct_0 and ct_1 encrypt $\mathbf{0}$ instead of $(\mathbf{x}_0 \| \mathbf{y}_0)$ and $(\mathbf{x}_1 \| \mathbf{y}_1)$, respectively.

Hybrid_4 : This hybrid differs from the previous hybrid in the way matrix $\mathbf{A}$ is computed. In this hybrid, the challenger first samples $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and then programs $\mathsf{H}_{\mathsf{A}}() = \mathbf{A}$.

Hybrid_5 : This hybrid differs from the previous hybrid in the way the challenger computes $\mathbf{t}_0$ and $\mathbf{t}_1$. Instead of sampling $\mathbf{x}_0$ (resp. $\mathbf{x}_1$) and computing $\mathbf{t}_0 = \mathbf{A}\mathbf{x}_0 + H(\mu_b)$ (resp. $\mathbf{t}_1 = \mathbf{A}\mathbf{x}_1 + H(\mu_{\bar{b}})$), it samples $\mathbf{u}_0$ (resp. $\mathbf{u}_1$) uniformly and sets $\mathbf{t}_0 = \mathbf{u}_0 + H(\mu_b)$ (resp. $\mathbf{t}_1 = \mathbf{u}_1 + H(\mu_{\bar{b}})$).

Indistinguishability of hybrids

In the following, we let $\mathsf{Adv}_i^{\mathsf{bl}}$ represent the advantage of $\mathcal{S}^*$ in the full-fledged blindness game in Hybrid_i. Then $\mathsf{Adv}_0^{\mathsf{bl}}$ is δ.

1. The only difference between Hybrid_0 and Hybrid_1 is in the way $\mathsf{PKE.pk}$ is computed. Hence if $\mathcal{S}^*$ succeeds in the full-fledged blindness game in Hybrid_1 with advantage $\mathsf{Adv}_1^{\mathsf{bl}}$, then we can design an algorithm C which runs in essentially the same time as $\mathcal{S}^*$ and distinguishes the public key of the PKE scheme from uniform with advantage at least $\mathsf{Adv}_0^{\mathsf{bl}} - \mathsf{Adv}_1^{\mathsf{bl}}$. Thus if the advantage of C is represented by $\mathsf{Adv}_{\mathsf{PKE}}$, we get $\mathsf{Adv}_1^{\mathsf{bl}} \geq \mathsf{Adv}_0^{\mathsf{bl}} - \mathsf{Adv}_{\mathsf{PKE}} = \delta - \mathsf{Adv}_{\mathsf{PKE}}$.

2. The only difference between Hybrid_1 and Hybrid_2 is in the way π_0 and π_1 are computed. The two hybrids are indistinguishable because of the zero-knowledge property of the NIZK. Hence $\mathsf{Adv}_2^{\mathsf{bl}} \geq \mathsf{Adv}_1^{\mathsf{bl}} - 2^{-\Omega(\lambda)} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}} - 2^{-\Omega(\lambda)}$.

3. The only difference between Hybrid_2 and Hybrid_3 is in the messages being encrypted by ct_0 and ct_1. The two hybrids are computationally indistinguishable because of the IND-CPA security of PKE. In particular, if advantage of $\mathcal{S}^*$ in the full-fledged blindness game in Hybrid_3 is $\mathsf{Adv}_3^{\mathsf{bl}}$, then there exists an adversary $\mathcal{B}$ against $\mathsf{IND\text{-}CPA}$ security of PKE with advantage $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$ such that $2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} \geq \mathsf{Adv}_2^{\mathsf{bl}} - \mathsf{Adv}_3^{\mathsf{bl}} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}} - 2^{-\Omega(\lambda)} - \mathsf{Adv}_3^{\mathsf{bl}}$. (Here, we consider twice of $\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$ since both ct_0 and ct_1 are replaced with encryptions of $\mathbf{0}$ and hence the $\mathsf{IND\text{-}CPA}$ security of PKE is called twice.).

4. The only difference between Hybrid_3 and Hybrid_4 is in the computation of matrix $\mathbf{A}$: the challenger first samples $\mathbf{A}$ uniformly from $\mathbb{Z}_q^{n\times m}$ and then programs $\mathsf{H}_{\mathsf{A}}() = \mathbf{A}$. The two hybrids are therefore, identical in the adversary's view in the random oracle model. Hence $\mathsf{Adv}_4^{\mathsf{bl}} = \mathsf{Adv}_3^{\mathsf{bl}} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}} - 2^{-\Omega(\lambda)} - 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}}$.

5. The only difference between Hybrid_4 and Hybrid_5 is in the choice of the masking term for $H(\mathsf{g})$. Since the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ are only used in the computations of the vectors $\mathbf{t}_0$ and $\mathbf{t}_1$, we have by the leftover hash lemma (Lemmas 2.3 and 2.1), that $\mathbf{A}\mathbf{x}_0$ and $\mathbf{A}\mathbf{x}_1$ are statistically indistinguishable from uniform $\mathbf{u}_0$ and $\mathbf{u}_1$. Hence Hybrid_4 and Hybrid_5 are indistinguishable. More concretely, we have $\mathsf{Adv}_5^{\mathsf{bl}} \geq \mathsf{Adv}_4^{\mathsf{bl}} - 2^{-\Omega(\lambda)} \geq \delta - \mathsf{Adv}_{\mathsf{PKE}} - 2^{-\Omega(\lambda)} - 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} - 2^{-\Omega(\lambda)}$.

However, in Hybrid_5, the adversary $\mathcal{S}^*$ has zero advantage in guessing the bit b since it is information theoretically hidden. Hence $\delta - \mathsf{Adv}_{\mathsf{PKE}} - 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} - 2^{-\Omega(\lambda)} \leq 0$, which is equivalent to $\mathsf{Adv}_{\mathsf{PKE}} + 2\mathsf{Adv}_{\mathsf{IND\text{-}CPA}} \geq \delta - 2^{-\Omega(\lambda)}$. ■

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

In this thesis, we studied different cryptographic primitives of practical importance in distributed settings. We categorized these primitives in two parts, where: (i) the data is distributed, and (ii) the authority is distributed.

Distributed Data

In the first part, We began with the study of attribute based encryption and predicate encryption in multi-input settings, where the data to be encrypted is the concatenation of multiple data generated independently at different locations. We initiated their study by defining MIABE and MIPE and formalizing their security definition under unbounded collusions. We defined two notions of security - the first (regular) notion does not allow any decrypting key query as is common in the case of single input ABE/PE. The second, stronger notion allows decrypting keys as well, with the necessary restrictions to prevent trivial attacks. We then gave two schemes for 2ABE for NC_1 using pairings and LWE. We proved their security in the bilinear generic group model and standard model using KOALA assumption, respectively. Using heuristic assumptions from [BV22], we constructed 2ABE for poly class and 3ABE for NC_1. We gave a generic compiler to translate any kABE to kPE, for a constant k, using lockable obfuscation which in turn, can be constructed from LWE. For stronger security, $k = 2$. This, along with our results for MIABE, implied 2PE for NC_1 and heuristic 2PE for P and 3PE for NC_1. We further significantly enhanced our results for MIABE to *any* constant arity, using the recently introduced lattice-based assumptions of evasive and tensor LWE. For NC_1 circuits, we used evasive LWE, while for circuits in P, we used evasive LWE along with suitable strengthening of the tensor LWE assumption. We note that for 2ABE in P, we only used evasive and tensor LWE as introduced by [Wee22] (without any strengthening). Since

these constructions are based only on lattice-based assumptions, they have the additional advantage of potentially being quantum safe. Using our compiler for MIPE, we get constant input PE for the same circuit classes under the same set of assumptions and LWE.

Then we moved towards building FE, which is a generalization of ABE and PE, in multi-party settings for attribute weighted sums (AWS) functionality. We gave the first constructions for attribute based multi-input FE (MIFE), multi-client FE (MCFE) and dynamic decentralized FE (DDFE) for AWS functionality. Previously, constructions for these primitives were known only for inner product functionality. Our constructions are based on pairings and proven selectively secure under the matrix DDH assumption.

We summarize our results from the first part in Table 8.1.

Primitive	Arity	Ref.	Decentr–alized	Corrup-tion	Security	Assumptions	Function class	Main contribution
2ABE	2	Sec. 3.6	data	No	Strong	LWE, and pairings (in GGM)	NC_1	introduced MIPE, formalized MIABE and MIPE. gave the first constructions for these primitives
2ABE	2	Sec. 3.7	data	No	Strong	LWE+KOALA	NC_1	
3ABE	3	Sec. 3.10	data	No	heuristic	heuristic	NC_1	
2ABE	2	Sec. 3.11	data	No	heuristic	heuristic	P	
2ABE	2	Sec. 4.6	data	No	Strong	evasive LWE and tensor LWE	P	
kABE	constant	Sec. 4.7.1	data	No	Strong	evasive LWE	NC_1	
kABE	constant	Sec. 4.7.3	data	No	Strong	evasive LWE, strong tensor LWE	P	
AB-MIFE	poly	Sec. 5.6	data	yes	Selective	Matrix DDH	AWS	
MCFE	poly	Sec. 5.7	data	yes	Selective	Matrix DDH	AWS	
DDFE	poly	Sec. 5.8	data and authority	yes	Selective	Matrix DDH	AWS	
kPE to kABE compiler from LWE • Strong security for $k = 2$ (3.9) • Weak security for any constant $k > 2$ (3.8)								

Table 8.1: Summary of our contributions in the first part of the thesis

Several interesting future directions emerge from our work.

1. *Extending the arity.* A dream goal in this direction would be to construct MIABE/MIPE for polynomial arity. Since MIABE with polynomial arity implies WE [BJK+18], it seems hard to construct a scheme with polynomial arity from LWE since construction for WE from LWE has remained elusive. However, WE

can be constructed from recently introduced assumptions of evasive LWE [Tsa22; VWW23]. So, one can hope to construct MIABE/MIPE with polynomial arity from these assumptions for general circuit classes like NC_1 and P. We constructed MIABE with constant arity using evasive and tensor LWE [Wee22]. It would also be very interesting to construct MIABE/MIPE for constant arity from standard LWE.

2. *Supporting class* P *from well studied assumptions.* Another significant extension would be to construct MIABE/MIPE for P from the well studied assumptions. We recall that our candidate construction for 2ABE for class P, is either based on the techniques developed in the context of succinct CPABE in [BV22], and similarly to [BV22], has no formal proof, or from evasive and tensor LWE. Our constant arity MIABE construction for P further requires strong tensor LWE. We would want to construct them from standard LWE. It would also be of independent interest to develop a proof for the assumption used in [BV22].

3. *Supporting user corruption.* In our schemes for MIABE and MIPE all the users share the same encryption key, which is a secret[1]. If the encryption key leaks from any one of the users, it will basically leak the key for all the users which will break the security. Hence, our construction can not support user corruption. Ideally, we would like to construct a scheme where each user has a different encryption key and the security should hold even if the encryption key of some of the users is leaked.

4. *Strengthening the* MIPE *compiler.* Our MIABE to MIPE compiler for stronger security supports only arity two. We would like to extend it to any constant arity as is the case for our compiler with regular security. Even further, we would like to design a compiler for polynomial arity.

Distributed Authority

Continuing with the theme of distributed cryptography, in the second part of the thesis, we focused on primitives with distributed authority. We studied threshold signatures and blind signatures. In threshold signatures, we improved Boneh et al.'s round-optimal lattice based threshold signatures in terms of efficiency and security. The main source of efficiency came from the use of Rényi divergence based distance measurement between two distributions instead of statistical distance which helped us reduce the noise flooding used in [BGG+18] from exponential to polynomial. This in turn, reduced the signature size from $\tilde{O}(\lambda^3)$ to $\tilde{O}(\lambda)$. We gave two more constructions that achieved partially

[1] In our construction for constant arity, the user holding the message encrypts using the public key, but all the other users use the common secret key.

adaptive and fully adaptive security as opposed to selective security in Boneh et al.'s construction. We then continued this thread with blind signatures where we constructed the first practical, round-optimal lattice-based blind signature. Our construction achieved overall better parameters. We achieved signature size of approx. 44KB and transcript size of 1.4 KB for 128 bits of security parameter. The security of our construction followed from a new and natural lattice-based assumption of one-more-ISIS, that we introduced. We studied our assumption with various attack strategies to gain confidence in its hardness. We summarize our results in the second part of the thesis in Table 8.2.

Primitive	Ref.	Decentr-alized	Security	Assumptions	Remark
Threshold Signature	Sec. 6.5	authority	selective unforgeability	LWE	improved efficiency- sig size reduced from $\tilde{O}(\lambda^3)$ to $\tilde{O}(\lambda)$
Threshold Signature	Sec. 6.7	authority	partially adaptive unforgeability	LWE	the first round optimal, lattice based threshold signature with adaptivity
Threshold Signature	Sec. 6.8	authority	fully adaptive unforgeability	LWE	Limitation: key size in fully adaptive construction grows linearly with number of signatures
Rejection free [Lyu12] signature	Sec. 6.6	-	unforgeability	LWE	
Blind Signature	Sec. 7.6.2	authority	one-more-unforgeability full-fledged blindness	one-more-ISIS	first overall practical and round optimal constr. from lattices. sig. size 45 KB, transcript 1.4 KB for 128 bits of security introduced one-more-ISIS assumption

Table 8.2: Summary of our contributions in the second part of the thesis

Below we list some directions for future work.

1. *Direct construction of threshold signature scheme:* Recall that our threshold signature scheme, similarly to Boneh et al., uses homomorphic encryption scheme where each party computes an encryption of the signature, as an intermediate step, by homomorphically evaluating the signing circuit on the encryption of the signing key, given as public parameters. Known constructions for lattice based signatures have complex circuits for which existing FHE constructions are not practical. This limits the practical usage of our scheme as well. While constructing a practical FHE scheme is an important active area of research in itself, it would be interesting to design a direct, practical construction for threshold signatures without using FHE.

2. *Fully adaptive threshold signature with succinct keys:* Our fully adaptive threshold

signature scheme has the limitation that the size of the signing key is linearly dependent on the number of signing queries Q_S. We would like to remove this dependency on Q_S.

3. *More applications of* one-more-ISIS *assumption:* It would be interesting to find more applications for our one-more-ISIS assumption. For example, we would like to explore the possibility of using the assumption in constructing other variants of blind signatures, like partial blind signatures, where only a part of the message is blinded.

4. *Threshold blind signatures:* A natural combination of threshold signature and blind signature defines the notion of threshold blind signatures, where a user needs to interact with at least t out of N signers to get a signature for his/her message. The notion of threshold blind signatures already exists in the literature [KM15; CXW06; CXYN07; JL99]. However, most of the constructions are from discrete log or RSA based assumptions. It would be interesting to explore the possibility of combining the techniques developed for threshold signatures and blind signatures in this thesis to construct lattice based threshold blind signatures.

5. *Decentralizing the* Setup *algorithm in our constructions:* In our constructions of different primitives, the setup algorithm is a centralized algorithm, thus placing the entire trust in a single trusted authority which again means a single point of security failure. So, it is interesting and desirable to decentralize the setup algorithm so that there is no single trusted authority. For example, in threshold signatures, we would like to let the parties choose their own partial signing keys. Similarly, in MIABE/MIPE, it would be interesting to construct a scheme where users choose their own encryption key, or at least these keys are derived from each party's/user's contribution.

BIBLIOGRAPHY

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020. https://doi.org/10.13154/tosc.v2020.i3.1-45.

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 553–572. Springer, 2010. http://dx.doi.org/10.1007/978-3-642-13190-5_28.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, volume 6223 of *LNCS*, pages 98–115. Springer, 2010. https://doi.org/10.1007/978-3-642-14623-7_6.

[ABB20a] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. BLAZE: practical lattice-based blind signatures for privacy-preserving applications. In *Financial Cryptography and Data Security*, volume 12059 of *LNCS*, pages 484–502. Springer, 2020. https://doi.org/10.1007/978-3-030-51280-4_26.

[ABB20b] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. On lattice-based interactive protocols: An approach with less or no aborts. In *ACISP*, volume 12248 of *LNCS*, pages 41–61. Springer, 2020. https://doi.org/10.1007/978-3-030-55304-3_3.

[ABD+17] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation, 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Round-1-Submissions.

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 136–151. Springer, 2001. https://doi.org/10.1007/3-540-44987-6_9.

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT*, volume 11923 of *LNCS*, pages 552–582. Springer, 2019. https://doi.org/10.1007/978-3-030-34618-8_19.

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *PKC*, volume 11443 of *LNCS*, pages 128–157. Springer, 2019. https://doi.org/10.1007/978-3-030-17259-6_5.

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, volume 10991 of *LNCS*, pages 597–627. Springer, 2018. https://doi.org/10.1007/978-3-319-96884-1_20.

[ACF+20] Shweta Agrawal, Michael Clear, Ophir Frieder, Sanjam Garg, Adam O'Neill, and Justin Thaler. Ad hoc multi-input functional encryption. In *ITCS*, volume 151 of *LIPIcs*, pages 40:1–40:41. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. https://doi.org/10.4230/LIPIcs.ITCS.2020.40.

[ACGU20] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In *ASICRYPT*, volume 12493 of *LNCS*, pages 467–497. Springer, 2020. 10.1007/978-3-030-64840-4_16.

[AD17] Martin R Albrecht and Amit Deo. Large modulus ring-LWE ≥ module-LWE. In *ASIACRYPT*, volume 10624 of *LNCS*, pages 267–296. Springer, 2017. https://doi.org/10.1007/978-3-319-70694-8_10.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—A new hope. In *USENIX Security*, volume 2016 of *SEC '16*, pages 327–343. USENIX Association, 2016. https://dl.acm.org/doi/10.5555/3241094.3241120.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011. https://doi.org/10.1007/978-3-642-25385-0_2.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New techniques for bootstrapping and instantiation. In *EUROCRYPT*, volume 11476 of *LNCS*, pages 191–225. Springer, 2019. https://doi.org/10.1007/978-3-030-17653-2_7.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT*, volume 10210 of *LNCS*, pages 601–626. Springer, 2017. https://doi.org/10.1007/978-3-319-56620-7_21.

[AGT21a] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO*, volume 12828 of *LNCS*, pages 208–238. Springer, 2021. https://doi.org/10.1007/978-3-030-84259-8_8.

[AGT21b] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *TCC*, volume 13043 of *LNCS*, pages 224–255. Springer, 2021. https://doi.org/10.1007/978-3-030-90453-1_8.

[AGT22] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC*, volume 13747 of *LNCS*, pages 711–740. Springer, 2022. https://doi.org/10.1007/978-3-031-22318-1_25.

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In *CRYPTO*, volume 12170 of *LNCS*, pages 685–716. Springer, 2020. https://doi.org/10.1007/978-3-030-56784-2_23.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM SIGSAC CCS*, CCS '17, pages 2087–2104. ACM, 2017. https://doi.org/10.1145/3133956.3134104.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175. IEEE, 2004. https://doi.org/10.1109/FOCS.2004.20.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, volume 9215 of *LNCS*, pages 308–326. Springer, 2015. https://doi.org/10.1007/978-3-662-47989-6_15.

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. In *CRYPTO*, volume 11694 of *LNCS*, pages 284–332. Springer, 2019. https://doi.org/10.1007/978-3-030-26954-8_10.

[AJS23] Paul Lou Aayush Jain, Huijia Lin and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum io. In *EUROCRYPT*, volume 14004 of *LNCS*, pages 205–235. Springer, 2023. https://doi.org/10.1007/978-3-031-30545-0_8.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, pages 99–108. ACM, 1996. https://dl.acm.org/doi/pdf/10.1145/

237814.237838.

[Ajt99] Miklos Ajtai. Generating hard instances of the short basis problem. In *ICALP*, volume 1644 of *LNCS*, pages 1–9. Springer, 1999. https://doi.org/10.1007/3-540-48523-6_1.

[AKS01] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, STOC, pages 601–610. ACM, 2001. https://doi.org/10.1145/380752.380857.

[AKYY23] Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Trace and revoke with optimal parameters from polynomial hardness. In *EUROCRYPT*, volume 14006 of *LNCS*, pages 605–636. Springer, 2023. https://doi.org/10.1007/978-3-031-30620-4_20.

[APM20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear fe. In *EUROCRYPT*, volume 12105 of *LNCS*, pages 110–140. Springer, 2020. https://doi.org/10.1007/978-3-030-45721-1_5.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_31.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from lwe and pairings in the standard model. In *TCC*, volume 12550 of *LNCS*, pages 149–178. Springer, 2020. https://doi.org/10.1007/978-3-030-64375-1_6.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and lwe. In *EUROCRYPT*, volume 12105 of *LNCS*, pages 13–43. Springer, 2020. https://doi.org/10.1007/978-3-030-45721-1_2.

[Bab85] László Babai. On Lovász' lattice reduction and the nearest lattice point problem (shortened version). In *STACS*, volume 182 of *LNCS*, pages 13–20. Springer, 1985. https://doi.org/10.1007/BFb0023990.

[BCD+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *ACM CCS*, CCS '16, pages 1006–1018. ACM, 2016. https://doi.org/10.1145/2976749.2978425.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and

Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *CRYPTO*, volume 10401 of *LNCS*, pages 67–98. Springer, 2017. https://doi.org/10.1007/978-3-319-63688-7_3.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT*, volume 11476 of *LNCS*, pages 103–128. Springer, 2019. https://doi.org/10.1007/978-3-030-17653-2_4.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, volume 5978 of *LNCS*, pages 201–218. Springer, 2010. https://doi.org/10.1007/978-3-642-11799-2_13.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, pages 10–24. ACM, 2016. https://doi.org/10.1137/1.9781611974331.ch2.

[BECE+20] Samuel Bouaziz-Ermann, Sébastien Canard, Gautier Eberhart, Guillaume Kaim, Adeline Roux-Langlois, and Jacques Traoré. Lattice-based (partially) blind signature without restart. *IACR Cryptol. ePrint Arch.*, 2020. https://eprint.iacr.org/2020/260.

[BF11] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *PKC*, volume 6571 of *LNCS*, pages 1–16. Springer, 2011. https://doi.org/10.1007/978-3-642-19379-8_1.

[BFF+14] Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In *CRYPTO*, volume 8616 of *LNCS*, pages 95–112. Springer, 2014. https://doi.org/10.1007/978-3-662-44371-2_6.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, volume 8366 of *LNCS*, pages 28–47. Springer, 2014. https://doi.org/10.1007/978-3-319-04852-9_2.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, volume 8441 of

LNCS, pages 533–556. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_30.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, volume 10991 of *LNCS*, pages 565–596. Springer, 2018. https://doi.org/10.1007/978-3-319-96884-1_19.

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001. https://doi.org/10.1007/3-540-44647-8_1.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC*, volume 9562 of *LNCS*, pages 209–224. Springer, 2016. https://doi.org/10.1007/978-3-662-49096-9_9.

[BGSS17] Olivier Blazy, Philippe Gaborit, Julien Schrek, and Nicolas Sendrier. A code-based blind signature. In *ISIT*, pages 2718–2722. IEEE, 2017. https://doi.org/10.1109/ISIT.2017.8007023.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012. http://doi.acm.org/10.1145/2090236.2090262.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT*, volume 9452 of *LNCS*, pages 470–491. Springer, 2015. https://doi.org/10.1007/978-3-662-48797-6_20.

[BJK+18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-io from standard assumptions. In *SCN*, volume 11035 of *LNCS*, pages 425–441. Springer, 2018. https://doi.org/10.1007/978-3-319-98113-0_23.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: threshold protocols for signatures and (H) IBE. In *ACNS*, volume 7954 of *LNCS*, pages 218–236. Springer, 2013. https://doi.org/10.1007/978-3-642-38980-1_14.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In *EUROCRYPT*, volume 12696 of *LNCS*, pages 33–53. Springer, 2021. https://doi.org/10.1007/978-3-030-77870-5_2.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In *CRYPTO*, volume 8042 of *LNCS*, pages 410–428. Springer, 2013. https://doi.org/10.1007/978-3-642-40041-4_23.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, STOC '13, pages 575–584. ACM, 2013. https://doi.org/10.1145/2488608.2488680.

[BLRL+18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. *J. Cryptol.*, 31:610–640, 2018. https://doi.org/10.1007/s00145-017-9265-9.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001. https://doi.org/10.1007/3-540-45682-1_30.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO*, volume 11692 of *LNCS*, pages 176–202. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_7.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *J. Cryptol.*, 16:185–215, 2003. https://doi.org/10.1007/s00145-002-0120-1.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *PKC*, volume 2567 of *LNCS*, pages 31–46. Springer, 2003. https://doi.org/10.1007/3-540-36288-6_3.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 719–737. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_42.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011. https://doi.org/10.1007/978-3-642-19571-6_16.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic

encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE Computer Society, 2011. https://doi.org/10.1109/FOCS.2011.12.

[BV15a] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, pages 171–190. IEEE Computer Society, 2015. https://doi.org/10.1109/FOCS.2015.20.

[BV15b] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, volume 9015 of *LNCS*, pages 1–30. Springer, 2015. 10.1007/978-3-662-46497-7_1.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In *CRYPTO*, volume 9816 of *LNCS*, pages 363–84. Springer, 2016. https://doi.org/10.1007/978-3-662-53015-3_13.

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext policy abe. In *ITCS*, volume 215 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. https://doi.org/10.4230/LIPIcs.ITCS.2022.28.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007. https://doi.org/10.1007/978-3-540-70936-7_29.

[BW19] Ward Beullens and Hoeteck Wee. Obfuscating simple functionalities from knowledge assumptions. In *PKC*, volume 11443 of *LNCS*, pages 254–283. Springer, 2019. https://doi.org/10.1007/978-3-030-17259-6_9.

[CCL+19] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO*, volume 11694 of *LNCS*, pages 191–221. Springer, 2019. https://doi.org/10.1007/978-3-030-26954-8_7.

[CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC*, volume 12111 of *LNCS*, pages 266–296. Springer, 2020. https://doi.org/10.1007/978-3-030-45388-6_10.

[CDG+18a] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT*, volume 11273 of *LNCS*, pages 703–732. Springer, 2018. https://doi.org/10.1007/978-3-030-03329-3_24.

[CDG+18b] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. https://eprint.iacr.org/2018/1021.

[CDSG+20] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In *CRYPTO*, volume 12170 of *LNCS*, pages 747–775. Springer, 2020. https://doi.org/10.1007/978-3-030-56784-2_25.

[CFS01] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *ASIACRYPT*, volume 2248 of *LNCS*, pages 157–174. Springer, 2001. https://doi.org/10.1007/3-540-45682-1_10.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *CCS*, CCS '20, pages 1769–1787. ACM, 2020. https://doi.org/10.1145/3372297.3423367.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33:34–91, 2020. https://doi.org/10.1007/s00145-019-09319-x.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Springer, 1982. https://doi.org/10.1007/978-1-4757-0602-4_18.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *EUROCRYPT*, volume 10820 of *LNCS*, pages 360–384. Springer, 2018. https://doi.org/10.1007/978-3-319-78381-9_14.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 523–552. Springer, 2010. https://doi.org/10.1007/978-3-642-13190-5_27.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, volume 10624 of *LNCS*, pages 409–437. Springer, 2017. https://doi.org/10.1007/978-3-319-70694-8_15.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, volume 9216 of

LNCS, pages 630–656. Springer, 2015. https://doi.org/10.1007/978-3-662-48000-7_31.

[CP92] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *CRYPTO*, volume 740 of *LNCS*, pages 89–105. Springer, 1992. https://doi.org/10.1007/3-540-48071-4_7.

[CS19] Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: threshold post-quantum signatures. In *IMACC*, volume 11929 of *LNCS*, pages 128–153. Springer, 2019. https://doi.org/10.1007/978-3-030-35199-1_7.

[CW14] Jie Chen and Hoeteck Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In *SCN*, volume 8642 of *LNCS*, pages 277–297. Springer, 2014. https://doi.org/10.1007/978-3-319-10879-7_16.

[CXW06] Xiangguo Cheng, Weidong Xu, and Xinmei Wang. A threshold blind signature from well pairing on elliptic curves. *Journal of electronics (China)*, 23(1):76–80, 2006. https://doi.org/10.1007/s11767-004-0071-9.

[CXYN07] Wei Cui, Yang Xin, Yixian Yang, and Xinxin Niu. A new blind signature and threshold blind signature scheme from pairings. In *CISW)*, pages 699–702. IEEE, 2007. https://doi.org/10.1109/CISW.2007.4425591.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC*, volume 9614 of *LNCS*, pages 164–195. Springer, 2016. https://doi.org/10.1007/978-3-662-49384-7_7.

[Des94] Yvo Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–458, 1994. https://doi.org/10.1002/ett.4460050407.

[DJN+20] Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergård. Fast threshold ECDSA with honest majority. In *SCN*, volume 12238 of *LNCS*, pages 382–400. Springer, 2020. https://doi.org/10.1007/978-3-030-57990-6_19.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018. https://doi.org/10.13154/tches.v2018.i1.238-268.

[DKLS18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure

two-party threshold ECDSA from ECDSA assumptions. In *S& P*, pages 980–997, 2018. https://doi.org/10.1109/SP.2018.00036.

[DKLS19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *S&P*, pages 1051–1066, 2019. https://doi.org/10.1109/SP.2019.00024.

[DLdW19] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate Voronoi cells. In *PQCrypto*, volume 11505 of *LNCS*, pages 3–22. Springer, 2019. https://doi.org/10.1007/978-3-030-25510-7_1.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, volume 9056 of *LNCS*, pages 617–640. Springer, 2015. https://doi.org/10.1007/978-3-662-46800-5_24.

[DOK+20] Anders Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS*, volume 12309 of *LNCS*, pages 654–673. Springer, 2020. https://doi.org/10.1007/978-3-030-59013-0_32.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In *PKC*, volume 10770 of *LNCS*, pages 245–277. Springer, 2018. https://doi.org/10.1007/978-3-319-76581-5_9.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC*, volume 12710 of *LNCS*, pages 99–130. Springer, 2021. https://doi.org/10.1007/978-3-030-75245-3_5.

[DP16] Léo Ducas and Thomas Prest. Fast Fourier orthogonalization. In *ISSAC*, ISSAC '16, pages 191–198. ACM, 2016. https://doi.org/10.1145/2930889.2930923.

[DP21] Pratish Datta and Tapas Pal. (Compact) adaptively secure FE for attribute-weighted sums from k-lin. In *ASIACRYPT*, volume 13093 of *LNCS*, pages 434–467. Springer, 2021. https://doi.org/10.1007/978-3-030-92068-5_15.

[dPK22] Rafael del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In *CRYPTO*, volume 13508 of *LNCS*, pages 306–336. Springer, 2022. https://doi.org/10.1007/978-3-031-15979-4_11.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In *TCC*, volume 13043, pages 256–287. Springer, 2021. https://doi.org/10.1007/978-3-030-90453-1_9.

[DRS18] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto*, volume 10786 of *LNCS*, pages 419–440. Springer, 2018. https://doi.org/10.1007/978-3-319-79063-3_20.

[DvW21] Léo Ducas and Wessel van Woerden. Ntru fatigue: How stretched is overstretched? In *ASIACRYPT*, volume 13093 of *LNCS*, pages 3–32. Springer, 2021. https://doi.org/10.1007/978-3-030-92068-5_1.

[EHK+17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, January 2017. 10.1007/s00145-015-9220-6.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *ASIACRYPT*, volume 12492 of *LNCS*, pages 259–288. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_9.

[ESLR22] Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and vrfs. *IACR Cryptol. ePrint Arch.*, 2022. https://eprint.iacr.org/2022/141.

[ESS+19] Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS*, volume 11464 of *LNCS*, pages 67–88. Springer, 2019. https://doi.org/10.1007/978-3-030-21568-2_4.

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Tang Yu. Shorter hash-and-sign lattice-based signatures. In *CRYPTO*, volume 13508 of *LNCS*, pages 245–275. Springer, 2022. https://doi.org/10.1007/978-3-031-15979-4_9.

[FFMV23] Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption from learning with errors. In *EUROCRYPT*, volume 14006 of *LNCS*, pages 573–604. Springer, 2023. https://doi.org/10.1007/978-3-031-30620-4_19.

[FHK+] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky,

Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Specification v1.0, available at https://falcon-sign.info/.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, volume 4117 of *LNCS*, pages 60–77. Springer, 2006. https://doi.org/10.1007/11818175_4.

[FKOS15] Tore Kasper Frederiksen, Marcel Keller, Emmanuela Orsini, and Peter Scholl. A unified approach to MPC with preprocessing using OT. In *ASIACRYPT*, volume 9452 of *LNCS*, pages 711–735. Springer, 2015. https://doi.org/10.1007/978-3-662-48797-6_29.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In *EUROCRYPT*, volume 12106 of *LNCS*, pages 63–95. Springer, 2020. https://doi.org/10.1007/978-3-030-45724-2_3.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 477–495. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_27.

[GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *CCS*, CCS '18, pages 1179–1194. ACM, 2018. https://doi.org/10.1145/3243734.3243859.

[GG20] Rosario Gennaro and Steven Goldfeder. One round threshold ECDSA with identifiable abort. *IACR Cryptol. ePrint Arch.*, 2020. https://eprint.iacr.org/2020/540.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 578–602. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_32.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013. https://doi.org/10.1109/FOCS.2013.13.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-

optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS*, volume 9696 of *LNCS*, pages 156–174. Springer, 2016. https://doi.org/10.1007/978-3-319-39555-5_9.

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, volume 12698 of *LNCS*, pages 97–126. Springer, 2021. https://doi.org/10.1007/978-3-030-77883-5_4.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010. http://hdl.handle.net/1721.1/73191.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security*, pages 519–535. USENIX Association, 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/grassi.

[GKSŚ20] Adam Gągol, Jędrzej Kula, Damian Straszak, and Michał Świętek. Threshold ecdsa for decentralized asset custody. *Cryptology ePrint Archive*, 2020. https://eprint.iacr.org/2020/498.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, pages 612–621. IEEE, 2017. https://doi.org/10.1109/FOCS.2017.62.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, volume 7428 of *LNCS*, pages 530–547. Springer, 2012. https://doi.org/10.1007/978-3-642-33027-8_31.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC*, STOC '21, pages 736–749. ACM, 2021. https://doi.org/10.1145/3406325.3451070.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, pages 89–98. Springer, 2006. https://doi.org/10.1145/1180405.1180418.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008. http://doi.acm.org/10.1145/1374376.1374407.

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In *CRYPTO*, volume 6841 of *LNCS*, pages 630–648. Springer, 2011. https://doi.org/10.1007/978-3-642-22792-9_36.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, volume 8042 of *LNCS*, pages 75–92. Springer, 2013. https://doi.org/10.1007/978-3-642-40041-4_5.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In *ASIACRYPT*, volume 9452 of *LNCS*, pages 550–574. Springer, 2015. https://doi.org/10.1007/978-3-662-48797-6_23.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, volume 62, pages 1–33. ACM, 2013. https://doi.org/10.1145/2824233.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate Encryption for Circuits from LWE. In *CRYPTO*, volume 9216 of *LNCS*, pages 503–523. Springer, 2015. https://doi.org/10.1007/978-3-662-48000-7_25.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In *EUROCRYPT*, volume 11478 of *LNCS*, pages 345–375. Springer, 2019. https://doi.org/10.1007/978-3-030-17659-4_12.

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In *CRYPTO*, volume 12171 of *LNCS*, pages 500–529. Springer, 2020. https://doi.org/10.1007/978-3-030-56880-1_18.

[HKM18] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving LWE. *Des. Codes Cryptogr.*, 86:55–83, 2018. https://doi.org/10.1007/s10623-016-0326-0.

[HLS18] Andreas Hülsing, Tanja Lange, and Kit Smeets. Rounded gaussians – fast and secure constant-time sampling for lattice-based crypto. In *PKC*, volume 10770 of *LNCS*, pages 728–757. Springer, 2018. https://doi.org/10.1007/978-3-319-76581-5_25.

[HPRR20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In *PQCrypto*, volume 12100 of *LNCS*, pages 53–71. Springer, 2020.

https://doi.org/10.1007/978-3-030-44223-1_4.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *CRYPTO*, volume 6841 of *LNCS*, pages 447–464. Springer, 2011. https://doi.org/10.1007/978-3-642-22792-9_25.

[IKSA03] Subariah Ibrahim, Maznah Kamat, Mazleena Salleh, and Sh.R. Abdul Aziz. Secure E-voting with blind signature. In *NCTT*, pages 193–197. IEEE, 2003. https://doi.org/10.1109/NCTT.2003.1188334.

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In *ICALP*, volume 8572 of *LNCS*, pages 650–662. Springer, 2014. https://doi.org/10.1007/978-3-662-43948-7_54.

[JL99] W-S Juang and C-L Lei. Partially blind threshold signatures based on discrete logarithm. *Computer Communications*, 22(1):73–86, 1999. https://doi.org/10.1016/S0140-3664(98)00214-X.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over R to build io. In *EUROCRYPT*, volume 11476 of *LNCS*, pages 251–281. Springer, 2019. https://doi.org/10.1007/978-3-030-17653-2_9.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO*, volume 1294 of *LNCS*, pages 150–164. Springer, 1997. https://doi.org/10.1007/BFb0052233.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, pages 60–73. ACM, 2021. https://doi.org/10.1145/3406325.3451093.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability Obfuscation from LPN over Large Fields, DLIN, and constant depth PRGs. In *EUROCRYPT*, volume 13275 of *LNCS*, pages 670–699. Springer, 2022. https://doi.org/10.1007/978-3-031-06944-4_23.

[KDK11] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *PETS*, volume 6794 of *LNCS*, pages 175–191. Springer, 2011. https://doi.org/10.1007/978-3-642-22263-4_10.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. ACM/SIAM, 2000. https://dl.acm.org/doi/10.5555/338219.338661.

[KLX22] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature

schemes in the algebraic group model. In *PKC*, volume 13178 of *LNCS*, pages 468–497. Springer, 2022. https://doi.org/10.1007/978-3-030-97131-1_16.

[KM15] Veronika Kuchta and Mark Manulis. Rerandomizable threshold blind signatures. In *INTRUST*, volume 9473, pages 70–89. Springer, 2015. https://doi.org/10.1007/978-3-319-27998-5_5.

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In *ASIACRYPT*, pages 375–404. Springer, 2020. https://doi.org/10.1007/978-3-030-64840-4_13.

[KS21] Kamil Kluczniak and Leonard Schild. Fdfb: Full domain functional bootstrapping towards practical fully homomorphic encryption. *arXiv preprint arXiv:2109.02731*, 2021. https://doi.org/10.48550/arXiv.2109.02731.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008. https://doi.org/10.1007/978-3-540-78967-3_9.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234. ACM, 2012. https://doi.org/10.1145/2213977.2214086.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT*, volume 9665 of *LNCS*, pages 28–57. Springer, 2016. https://doi.org/10.1007/978-3-662-49890-3_2.

[Lin17a] Huijia Lin. Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs. In *CRYPTO*, volume 10401 of *LNCS*, pages 599–629. Springer, 2017. https://doi.org/10.1007/978-3-319-63688-7_20.

[Lin17b] Yehuda Lindell. Fast secure two-party ECDSA signing. In *CRYPTO*, volume 10402 of *LNCS*, pages 613–644. Springer, 2017. https://doi.org/10.1007/978-3-319-63715-0_21.

[LL20a] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In *EUROCRYPT*, volume 12107 of *LNCS*, pages 247–277. Springer, 2020. https://doi.org/10.1007/978-3-030-45727-3_9.

[LL20b] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-lin. In *ASIACRYPT*, volume 12493 of *LNCS*, pages 437–466. Springer, 2020. https://doi.org/10.1007/978-3-030-64840-4_15.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, volume 4052 of *LNCS*, pages 144–155. Springer, 2006. https://doi.org/10.1007/11787006_13.

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS*, pages 1837–1854. ACM, 2018. https://doi.org/10.1145/3243734.3243788.

[LNP22a] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In *PKC*, volume 13178 of *LNCS*, pages 498–527. Springer, 2022. https://doi.org/10.1007/978-3-030-97131-1_17.

[LNP22b] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general. In *CRYPTO*, volume 13508, pages 71–101. Springer, 2022. https://doi.org/10.1007/978-3-031-15979-4_3.

[LNPS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plançon, and Gregor Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In *ASIACRYPT*, volume 13093 of *LNCS*, pages 218–248. Springer, 2021. https://doi.org/10.1007/978-3-030-92068-5_8.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In *PKC*, volume 12710 of *LNCS*, pages 215–241. Springer, 2021. https://doi.org/10.1007/978-3-030-75245-3_9.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC*, volume 7778 of *LNCS*, pages 107–124. Springer, 2013. https://doi.org/10.1007/978-3-642-36362-7_8.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010. https://doi.org/10.1007/978-3-642-13190-5_4.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices

and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 1–31. Springer, 2010. https://doi.org/10.1007/978-3-642-13190-5_1.

[LPS10] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. Public-key cryptographic primitives provably as secure as subset sum. In *TCC*, volume 5978 of *LNCS*, pages 382–400. Springer, 2010. https://doi.org/10.1007/978-3-642-11799-2_23.

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In *CRYPTO*, volume 8616 of *LNCS*, pages 315–334. Springer, 2014. https://doi.org/10.1007/978-3-662-44371-2_18.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75:565–599, 2015. https://doi.org/10.1007/s10623-014-9938-4.

[LSK+19] Huy Quoc Le, Willy Susilo, Thanh Xuan Khuc, Minh Kim Bui, and Dung Hoang Duong. A blind signature from module lattices. In *DSC*, pages 1–8, 2019.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 239–256. Springer, 2014. https://doi.org/10.1007/978-3-642-55220-5_14.

[LT19] Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT*, volume 11923 of *LNCS*, pages 520–551. Springer, 2019. https://doi.org/10.1007/978-3-030-34618-8_18.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, pages 11–20. IEEE, 2016. https://doi.org/10.1109/FOCS.2016.11.

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 547–567. Springer, 2011. https://doi.org/10.1007/978-3-642-20465-4_30.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, volume 7417 of *LNCS*, pages 180–198. Springer, 2012. https://doi.org/10.1007/978-3-642-32009-5_12.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009. https://doi.org/10.1007/978-3-642-10366-7_35.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_43.

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In *IMACC*, volume 3796 of *LNCS*, pages 1–12. Springer, 2005. https://doi.org/10.1007/11586821_1.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems - a cryptographic perspective*. Springer, 2002. https://doi.org/10.1007/978-1-4615-0897-7.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_41.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. https://doi.org/10.1137/S0097539705447360.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, volume 9666 of *LNCS*, pages 735–763. Springer, 2016. https://doi.org/10.1007/978-3-662-49896-5_26.

[NIS17] Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2017. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[NPP22] Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. In *ASIACRYPT*, volume 13791, pages 95–125. Springer, 2022. https://doi.org/10.1007/978-3-031-22963-3_4.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467. IEEE Computer Society, 1997. https://doi.org/10.1109/SFCS.1997.646134.

[NV08] Phong Q. Nguyên and Thomas Vidick. Sieve algorithms for the shortest

vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008. https://doi.org/10.1515/JMC.2008.009.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, volume 740 of *LNCS*, pages 31–53. Springer, 1992. https://doi.org/10.1007/3-540-48071-4_3.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010. https://doi.org/10.1007/978-3-642-14623-7_11.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 591–608. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_35.

[PHVBS19] Dimitrios Papachristoudis, Dimitrios Hristu-Varsakelis, Foteini Baldimtsi, and George Stephanides. Leakage-resilient lattice-based partially blind signatures. *IET Information Security*, 13(6):670–684, 2019. https://doi.org/10.1049/iet-ifs.2019.0156.

[PP19] Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. In *PKC*, volume 11443 of *LNCS*, pages 504–533. Springer, 2019. https://doi.org/10.1007/978-3-030-17259-6_17.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, volume 3876 of *LNCS*, pages 145–166. Springer, 2006. https://doi.org/10.1007/11681878_8.

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In *ASIACRYPT*, volume 10624 of *LNCS*, pages 347–374. Springer, 2017. https://doi.org/10.1007/978-3-319-70694-8_13.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13:361–396, 2000. https://doi.org/10.1007/s001450010003.

[PSM17] Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A practical multivariate blind signature scheme. In *Financial Cryptography and Data Security*, volume 10322 of *LNCS*, pages 437–454. Springer, 2017. https://doi.org/10.1007/978-3-319-70972-7_25.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. https://doi.org/10.1145/1568318.1568324.

[Ros20] Mélissa Rossi. *Extended Security of Lattice-Based Cryptography*. PhD thesis, Université PSL, 2020. https://www.di.ens.fr/~mrossi/docs/thesis.pdf.

[Rüc10] Markus Rückert. Lattice-based blind signatures. In *ASIACRYPT*, volume 6477 of *LNCS*, pages 413–430. Springer, 2010. https://doi.org/10.1007/978-3-642-17373-8_24.

[SBC+07] Elaine Shi, John Bethencourt, TH Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *SP*, pages 350–364. Springer, 2007. https://doi.org/10.1109/SP.2007.29.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994. https://doi.org/10.1007/BF01581144.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997. https://doi.org/10.1007/3-540-69053-0_18.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 617–635. Springer, 2009. https://doi.org/10.1007/978-3-642-10366-7_36.

[Ste96] Jacques Stern. A new paradigm for public key identification. *IEEE Trans. Inf. Theory*, 42(6):1757–1768, 1996. https://doi.org/10.1109/18.556672.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005. https://doi.org/10.1007/11426639_27.

[Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In *ASIACRYPT*, volume 11923 of *LNCS*, pages 459–488. Springer, 2019. https://doi.org/10.1007/978-3-030-34618-8_16.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In *CRYPTO*, volume 11692 of *LNCS*, pages 62–85. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_3.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In

CRYPTO, LNCS, pages 535–559. Springer, 2022. https://doi.org/10.1007/978-3-031-15802-5_19.

[TT15] Katsuyuki Takashima and Atsushi Takayasu. Tighter security for efficient lattice cryptography via the rényi divergence of optimized orders. In *ProvSec*, volume 9451 of *LNCS*, pages 412–431. Springer, 2015. https://doi.org/10.1007/978-3-319-26059-4_23.

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In *EUROCRYPT*, volume 13276 of *LNCS*, pages 782–811. Springer, 2022. https://doi.org/10.1007/978-3-031-07085-3_27.

[VWW23] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive lwe. In *ASIACRYPT*, volume 13791 of *LNCS*, pages 195–221. Springer, 2023. https://doi.org/10.1007/978-3-031-22963-3_7.

[Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, volume 7417 of *LNCS*, pages 218–235. Springer, 2012. https://doi.org/10.1007/978-3-642-32009-5_14.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014. https://doi.org/10.1007/978-3-642-54242-8_26.

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC*, volume 10677 of *LNCS*, pages 206–233. Springer, 2017. https://doi.org/10.1007/978-3-319-70500-2_8.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In *EUROCRYPT*, volume 13276 of *LNCS*, pages 217–241. Springer, 2022. 10.1007/978-3-031-07085-3_8.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *EUROCRYPT*, volume 12698, pages 127–156. Springer, 2021. https://doi.org/10.1007/978-3-030-77883-5_5.

[WWW22] Brent Waters, Hoeteck Wee, and David J Wu. Multi-authority ABE from lattices without random oracles. In *TCC*, volume 13747 of *LNCS*, pages 651–679. Springer, 2022. https://doi.org/10.1007/978-3-031-22318-1_23.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, pages 600–611. IEEE, 2017. https://doi.org/10.1109/FOCS.2017.61.

[YAZ+19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO*, volume 11692 of *LNCS*, pages 147–175. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_6.

[YL19] Xun Yi and Kwok-Yan Lam. A new blind ECDSA scheme for bitcoin transaction anonymity. In *Asia-CCS*, pages 613–620. ACM, 2019. https://doi.org/10.1145/3321705.3329816.

[ZDH20] Ruiyu Zhu, Changchang Ding, and Yan Huang. Practical MPC+ FHE with applications in secure multi-partyneural network evaluation. *IACR Cryptol. ePrint Arch.*, 2020. https://eprint.iacr.org/2020/550.

[ZSS20] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. COSAC: compact and scalable arbitrary-centered discrete Gaussian sampling over integers. In *PQCrypto*, volume 12100 of *LNCS*, pages 284–303. Springer, 2020. https://doi.org/10.1007/978-3-030-44223-1_16.

CURRICULUM VITAE

NAME Anshu Yadav

DATE OF BIRTH 01 July 1987

EDUCATION QUALIFICATIONS

2018 **MS by Research**

Institution	Indian Institute of Technology, Bombay
Specialization	Computer Science and Engineering

2012 **M.Tech.**

Institution	DAIICT, Gandhinagar
Specialization	Information and Communication Technology

2010 **B.Tech.**

Institution	BBDNITM, Lucknow
Specialization	Computer Science and Engineering

DOCTORAL COMMITTEE

Chairperson	Prof. Balaram Ravindran 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
	Dr. Nishanth Chandran Microsoft Research, India