On the Practical Security of Inner Product Functional Encryption

Shashank Agrawal Shweta Agrawal Saikrishna Badrinarayanan
Abishek Kumarasubramanian Manoj Prabhakaran Amit Sahai

Abstract

Functional Encryption (FE) is an exciting new paradigm that extends the notion of public key encryption. In this work we explore the security of Inner Product Functional Encryption schemes with the goal of achieving the highest security against practically feasible attacks. While there has been substantial research effort in defining meaningful security models for FE, known definitions run into one of the following difficulties – if general and strong, the definition can be shown impossible to achieve, whereas achievable definitions necessarily restrict the usage scenarios in which FE schemes can be deployed.

We argue that it is extremely hard to control the nature of usage scenarios that may arise in practice. Any cryptographic scheme may be deployed in an arbitrarily complex environment and it is vital to have meaningful security guarantees for general scenarios. Hence, in this work, we examine whether it is possible to analyze the security of FE in a wider variety of usage scenarios, but with respect to a meaningful class of adversarial attacks known to be possible in practice. Note that known impossibilities necessitate that we must either restrict the usage scenarios (as done in previous works), or the class of attacks (this work). We study real world loss-of-secrecy attacks against Functional Encryption for Inner Product predicates constructed over elliptic curve groups. Our main contributions are as follows:

- We capture a large variety of possible usage scenarios that may arise in practice by providing a *stronger*, more general, intuitive framework that supports *function privacy* in addition to data privacy, and a separate *encryption key* in addition to public key and master secret key. These generalizations allow our framework to capture program obfuscation as a special case of functional encryption, and allows for a separation between users that encrypt data, access data and produce secret keys.
- We note that the landscape of attacks over pairing-friendly elliptic curves have been the subject of extensive research and there now exist constructions of pairing friendly elliptic curves where the complexity of all known non-generic attacks is (far) greater than the complexity of generic attacks. Thus, by appropriate choice of the underlying elliptic curve, we can capture all known practically feasible attacks on secrecy by restricting our attention to generic attacks.
- We construct a new inner product FE scheme using prime order groups and show it secure under our new, hitherto strongest known framework in the generic group model, thus ruling out all generic attacks in arbitrarily complex real world environments. Since our construction is over prime order groups, we rule out factoring attacks that typically force higher security parameters. Our concrete-analysis proofs provide guidance on the size of elliptic curve groups that are needed for explicit complexity bounds on the attacker.

Keywords: Functional Encryption, Practical Security, Pairing Based Cryptography, Inner Product Encryption, Generic Attacks, Simulation Based Security.

1 Introduction

Functional Encryption [SW05, SW] (FE) is an exciting new paradigm that generalizes public key encryption. In functional encryption, each decryption key corresponds to a specific function. When the holder of a decryption key for the function f gets an encryption of a message m, the only thing his key allows him to learn is $f(m)$, but nothing more.

Classic results in the area focused on constructing FE for restricted classes of functions – point functions or identity based encryption (IBE) [Sha84, BF01, Coc01, BW06, GPV08, CHKP10, ABB10a, ABB10b] threshold functions [SW05], membership checking [BW07], boolean formulas [GPSW06, BSW07, LOS$^+$10], inner product functions [KSW08, LOS$^+$10, AFV11] and more recently, even regular languages [Wat12]. Recent constructions of FE support general functions: Gurabov et al. [GVW13] and Garg et al. [GGH$^+$13b] provided the first constructions for an important subclass of FE called "public index FE" (also known as "attribute based encryption") for all circuits, Goldwasser et al. [GKP$^+$13b] constructed succinct simulation-secure single-key FE scheme for all circuits. In a breakthrough result, Garg et al. [GGH$^+$13a] constructed indistinguishability-secure multi-key FE schemes for all circuits. Goldwasser et al. and Ananth et al. [GKP$^+$13a, ABG$^+$13] constructed FE for Turing machines. Recently, Functional Encryption has even been generalized to multi-input functional encryption [GGG$^+$14].

Alongside ever-more-sophisticated constructions, there has been significant work in defining the right security model for FE. Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10] proposed definitional frameworks to study Functional Encryption in its general form. These works discussed the subtleties involved in defining a security model for FE that captures meaningful real world security. Since then there has been considerable research focus on understanding what security means for FE and whether it can be achieved [BSW11, O'N10, BO12, BF13, AGVW13, CIJ$^+$13]. The strongest, most intuitive notions of security turned out to be impossible to realize theoretically, while weaker notions restricted the usage scenarios in which FE schemes could be deployed (more on this below).

Security of Functional Encryption in practice. In this work we explore the security of Functional Encryption schemes from a practical standpoint, with the goal of trying to achieve maximum security against all practically feasible attacks. While there has been considerable progress in defining meaningful security models for FE, existing definitions do not capture a number of real world usage scenarios that will likely arise in practice. However, it is essential to understand how Functional Encryption systems behave in complex real world environments, since this is inevitable in the future of FE. Towards this end, we examine security features that we believe are desirable in practice, and discuss whether these can be achieved.

- *Can we hide the function?* Consider the application of keyword searching on encrypted data, where the keywords being searched for are sensitive and must remain hidden. This scenario is well motivated in practice; for example the FBI might recruit untrusted server farms to perform searches on confidential encrypted data, but desire not to reveal the words being searched. Can FE schemes achieve this?

- *Can we limit what the adversary learns to* only *the function's output?* Intuitively, a functional encryption scheme should only reveal to a decryptor the function output, and nothing more. For example, if the function has some computational hiding properties, can we guarantee that the FE scheme does not leak any additional information beyond the function output?

- *Can an adversary break FE schemes where it can ask for keys after receiving ciphertexts?* In real world applications, it is very likely that an adversary can receive authorized decryption keys even after it obtains the ciphertext that it is trying to break. For example, in searchable encryption, the decryption key corresponding to a search would only be given out after the encrypted database is publicly available. Similarly in Identity Based Encryption, a user may receive an email encrypted with his identity before he obtains the corresponding secret key. Can one guarantee that an attacker who obtains an arbitrary interleaving of cihertexts and keys, can learn nothing beyond the legitimate function values?

None of the existing security definitions for FE [BSW11, O'N10, BO12, BF13, AGVW13] provide comprehensive guarantees against all the above usage scenarios. Below, we discuss why this is the case, and examine alternate approaches to providing meaningful security guarantees against a wide range of practical attacks, in all the above scenarios.

Recap of Security Definitions. Before we discuss our approach, it will be useful to recap existing definitions of security and discuss their restrictions. Known definitions of security for FE may be divided into two broad classes: Indistinguishability (IND) based or Simulation (SIM) based. Indistinguishability based security stipulates that it is infeasible to distinguish encryptions of any two messages, without getting a secret key that decrypts the ciphertexts to distinct values; simulation-based security stipulates that there exists an efficient simulator that can simulate the view of the adversary, given only the function evaluated on messages and keys. Both of these notions can be further classified as follows: [O'N10] described the divide between *adaptive* (AD) versus *non-adaptive* (NA) which captures whether the adversary's queries to the key derivation oracle may or may not depend on the challenge ciphertext; and [GVW12] described the divide between *one* versus *many*, which depends on whether the adversary receives a single or multiple challenge ciphertexts. Thus, existing definitions of security belong to the class $\{1, \mathsf{many}\} \times \{\mathsf{NA}, \mathsf{AD}\} \times \{\mathsf{IND}, \mathsf{SIM}\}$.

Standard Model woes. Unfortunately, none of the above definitions capture security in all the usage scenarios discussed above. For example, Boneh et al and O'Neill [BSW11, O'N10] showed that IND based definitions do not capture scenarios where it is required that the user learn *only* the output of the FE function, for eg., when the function hides something computationally. To get around this, [BSW11, O'N10] proposed SIM based definitions that study FE in the "ideal world-real world" paradigm. However, the world of SIM security for FE has been plagued with impossibilities of efficient simulation. Also, even the strongest known SIM based definitions (many-AD-SIM) do not capture function hiding. Even disregarding function hiding, [BSW11] showed that many-AD-SIM is impossible even for very simple functionalities. A weakening of AD-SIM, namely NA-SIM [O'N10] does not capture scenarios where users may obtain keys *after* obtaining new third-party-generated ciphertexts. Despite this severe restriction on usage, NA-SIM was also shown to also be impossible [AGVW13, BO12], seemingly ruling out security for even those usage scenarios that *are* captured.

Does this mean nothing can be said about real world security of FE in scenarios not captured by definitions or ruled out by impossibilities for simulation? Given that strong, intuitive definitions capturing real world scenarios are unachievable, are practitioners doomed to make do with the restricted usage scenarios offered by IND based security?

There seem to be two complementary directions forward. The first is is to seek notions of security "in-between" IND and SIM that are achievable, thus providing guarantees for a restricted (but larger than IND) class of usage scenarios against all efficient attackers. Indeed, there is already research effort pursuing this agenda [AGVW13, BF13]. However, it is extremely hard (if not impossible) to control the nature of usage scenarios that arise in practice. A second direction is to examine whether it is possible to address as many usage scenarios as we can, but restrict ourselves to analyzing security only against classes of attacks that are known to be practically feasible. This is the approach we take in this work.

In this work we study the practical security of Functional Encryption for Inner Product predicates, which is the state of the art for general FE [KSW08, LOS$^+$10, AFV11]. However, we believe that the ideas developed in this work will be applicable to all FE schemes that are built from pairings on elliptic curves, which captures the majority of known FE constructions [BF01, SW05, GPSW06, BW06, BSW07, KSW08, LOS$^+$10, Wat12].

Real World attacks on Elliptic Curve based FE. The impossibilities exhibited by [BSW11, AGVW13] work by arguing that there exist scenarios which preclude existence of a simulator by information theoretic arguments. However, non-existence of a simulator does not imply real world attacks in the sense of distinguishing between ciphertexts or recovering any useful information about the message or the key. Arguably, attacks that cause actual loss of secrecy are the attacks that we care about in practice, and this is the class of attacks we consider in this work.

For pairing friendly elliptic curves that are used for FE constructions, there has been extensive research effort studying practically feasible attacks. Attacks can be of two kinds: those that respect the algebraic structure of the underlying groups, which are called *generic* attacks, and those that do not, or *non-generic* attacks. Generic attacks are described as algorithms that act oblivious of particular group representations. Due to its importance and wide applicability, much research effort has been focused on studying the complexity of generic and non-generic attacks on pairing-friendly elliptic curves. By now, there is a long line of work [FST10, Fre06, AFCK$^+$13, Cos12] focused on constructing pairing friendly elliptic curves where the complexity of all known non-generic attacks is extremely high. If such elliptic curves are used to build cryptographic schemes, there is strong heuristic evidence that the only successful practically feasible attacks will be generic in nature. We stress that we will work with elliptic curve groups of prime order, and so factoring-based attacks will not be relevant.

A well known mathematical model to study generic attacks is the *Generic Group Model* (GGM) [Nec94, Sho97].

In the GGM, all algorithms obtain access to elements of the group via random "handles" (of sufficient length) and remain unaware of their actual representations. The GGM has a strong track record of usefulness; indeed, even notable critics of provable security, Koblitz and Menezes, despite their criticisms, admit that the generic group model has been unreasonably successful at resisting attack [KM06].

Our Results. We investigate the security of inner product FE in the generic group model under a new strong framework for security, that captures *all* the usage scenarios discussed above simultaneously. This rules out a large class of attacks – namely arbitrary generic attacks – against the scheme deployed in an arbitrary usage environment. We construct a new inner product FE scheme based on prime order elliptic curve groups. Our results may be summarized as follows.

- *Capturing arbitrary usage scenarios:* We begin by providing a strong, simple and intuitive framework for security which captures all usage scenarios discussed above. Our framework captures function hiding in addition to data hiding; thus it guarantees that $\mathsf{CT}_{\vec{x}}$ and SK_f reveal no information about *either* $\vec{x}$ or f beyond what is revealed by $f(\vec{x})$. Generalized this way, our framework can be seen to subsume program obfuscation. We also introduce the idea of having a separate encryption key in the context of Functional Encryption. This setting lies between public and symmetric key functional encryption, in that while the encryption key is not publicly known to all users, it is also not the same as the master secret key used for generating secret keys for users in the system. This allows for a division between the people that create encryptions and the people that issue secret keys. We believe this setting is well motivated in the real world, since it is often the case that there is a hierarchy that separates the people that create encrypted data and people that access it. A real-world example would be an FBI encrypted database where police officers can be granted access to parts of the database, but only FBI personnel can add to the database.
- *Resisting generic attacks:* We show that our inner product FE scheme is secure under our strong framework in the Generic Group Model, resolving the problem left open by [BSW11] and [BF13]. We obtain *unconditional statistical security* for our scheme under our framework in the GGM. Our positive results also translate to the setting of obfuscation, achieving obfuscation for the inner product functionality secure against generic attacks.
- *Concrete security analysis:* Our security analysis is concrete, and as a result we can show exactly what parameters are needed to (provably) achieve security against attackers with different computational resources. For example, we show that with a pairing-friendly elliptic curve group whose order is a 222-bit prime, an attacker who is restricted to 2^{80} generic computations, breaks our scheme with at most 2^{-60} probability of success. Some additional security calculations are provided in Table 1.

Adversary Runtime	Attack Probability	Required Prime Group Order (bitlength)
2^{80}	2^{-60}	222 bits
2^{80}	2^{-80}	242 bits
2^{100}	2^{-80}	282 bits
2^{128}	2^{-80}	338 bits
2^{128}	2^{-128}	386 bits

Table 1: The table entries contain the bit length of security parameter to achieve the corresponding level of security.

Our Perspective. By showing that our strong security framework is realizable against all generic attacks, we are providing strong evidence of real-world security even when the generic model is instantiated in a heuristic manner – in our case with a suitably chosen pairing-friendly elliptic curve group. Much care and study is required for how, what, and when security is preserved in such instantiations – indeed this is a very active and important area of research in our community for the Random Oracle Model. We believe that guarantees obtained by such analysis are extremely useful in practice. For example, consider the example of an IBE used in practice, say in a large organization [vol]. Suppose the public parameters are published, and some user creates and publishes $2n$ encryptions for users who have yet to obtain their secret keys. Now, if n out of $2n$ users are chosen in some arbitrary, ciphertext-dependent way, and these users obtain their keys, are the remaining n encryptions secure? Simulation based definitions are the only definitions we know that capture security of the IBE in such scenarios, but it was shown by [BSW11] that there cannot exist a simulator for $\mathsf{many\text{-}AD\text{-}SIM}$ security of IBE. On the positive side, [BSW11] also showed that IBE does satisfy $\mathsf{many\text{-}AD\text{-}SIM}$ in the

Random Oracle Model. We believe that this is evidence that IBEs indeed provide *practical security* in scenarios such as the above, *even despite* the impossibility of simulation in this scenario.

We do caution that care needs to be exercised in understanding the requirements of any application of FE, and there may be applications for which our guarantees of security against generic attacks do not suffice. Intuitively these are applications where the main threat is not leaking secret information but in *not* being able to actually *simulate* some view. The only example of such a security property that we know of is *deniability*, where only the existence of a simulator would give plausible deniability to a participant. We stress that our analysis of generic attacks should not be taken to imply any kind of deniability.

Related Work and Comparison The study of simulation (SIM) based security for FE was initiated independently by Boneh et al [BSW11] and O'Neill [O'N10]. Boneh, et al. [BSW11] showed an impossibility even for the "simple" IBE functionality under many-AD-SIM security in the non programmable random oracle model. Bellare and O'Neill [BO12] strengthened this to show that many-AD-SIM security is impossible to achieve even in the standard model. They also put forward simulation-based definitions for Functional Encryption with non-black-box simulators. [AGVW13] ruled out general functional encryption for 1-NA-SIM security. On the positive side, O'Neill showed that for certain classes of functions called *preimage samplable functions*, many-NA-IND and many-NA-SIM are equivalent. In recent work, Barbosa and Farshim [BF13], extend O'Neill's equivalence between indistinguishability and semantic security to *restricted adaptive key extraction* attacks and show that this equivalence holds for a large class of functionalities.

The only positive results currently known for many-AD-SIM for FE are by Boneh et al.[BSW11] for very few functionalities in the Random Oracle Model. [BF13] show that inner product functionality of [KSW08] can be used to encode a one way function under the Small Integer Solution (SIS) problem, and hence natural approaches to prove its restricted adaptive simulation security fail in the standard model. The problem of proving semantic security of inner product FE is explicitly left open both by [BSW11] and [BF13].

The question of function privacy (or key hiding) was considered by Shen et al [SSW09], in the symmetric key setting and more recently by Boneh et al [BRS13] in the public key setting under IND based definitions. [SSW09] provide a construction of FE for inner product predicates in the standard model, under the IND based notion of security, using composite order groups and assuming hardness of factoring (even when viewed in the GGM). Our result on the other hand, is unconditionally statistically secure in the generic group model, under a strong simulation based definition of security, using prime order groups. Our construction for inner product FE is inspired by the scheme of [KSW08] and the works of [GKSW10, Fre10, OT08, OT09, LOS$^+$10, Lew12]. Our result implies a program obfuscator for the inner product functionality in the generic group model, for details see Appendix F. Our obfuscator is related to the hyperplane-membership obfuscator of [CRV10].

Our Techniques Prior to our work, the only techniques to achieve positive results for many-AD-SIM security of FE were in the programmable ROM, for the anonymous IBE and public-index functionalities, based on techniques to build non-committing encryption in the ROM [BSW11]. We develop new and entirely different techniques to achieve positive results for inner product FE in the GGM under a definition stronger than many-AD-SIM.

As an illustrative example, consider the scenario where the adversary has the encryption key. In this setting, the adversary may encrypt any vector of his choice, and run the decrypt operation with the secret key he is given and the messages he encrypted to learn relations between them. The simulator needs to learn what vectors the adversary is encrypting so as to query the function oracle and program the requisite relations to hold. However, this strategy is complicated by the fact that the adversary need not generate ciphertexts honestly and attempt to decrypt them honestly; instead he can carry out an arbitrarily obfuscated sequence of group operations, which may implicitly be encrypting and decrypting values. Our proof handles this issue by deploying a novel algebraic message extraction technique – the simulator keeps track of all algebraic relations that the adversary is developing, and is able to test if the algebraic relation that the adversary is developing depends on some property of an unknown vector $\vec{v}$ corresponding to a decryption key. We prove by algebraic means that if this happens, the adversary *can only* be checking whether $\vec{v}$ is orthogonal to some other vector $\vec{u}$. No other algebraic relations about $\vec{v}$ can be checked by the adversary because of the randomization present in our inner product FE scheme, except with negligible probability. Furthermore, in this case we prove that the vector $\vec{u}$ can only be either a vector corresponding to some challenge (honestly generated by the system, not the adversary) ciphertext, or a vector $\vec{u}$ that the simulator can fully extract from the adversary's algebraic queries.

The generic group model allows us to bypass impossibility because the adversary is forced to perform computations via the generic group oracle which the simulator can control. At a high level, the simulator keeps track of the queries requested by the adversary, uses these queries to learn what the adversary is doing, and carefully programming the oracle to maintain the requisite relations between group elements to behave like the ideal world in the view of the adversary. For further technical details, please see the proof in Section 5.

2 Preliminaries

In Appendix A, we define some standard notation that is used throughout the paper. We emphasize that all our groups are multiplicative, and any additive notation refers to computations in the exponent.

2.1 Functional Encryption

A functional encryption scheme $\mathcal{FE}$ for consists of four algorithms $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined as follows.

- $\mathsf{Setup}(1^\kappa)$ is a p.p.t. algorithm that takes as input the unary representation of the security parameter and outputs the public parameters, encryption key and master secret key $(\mathsf{PP}, \mathsf{EK}, \mathsf{MSK})$. Implicit in the public parameters PP are the security parameter and a function class $\mathcal{F}_{\mathsf{PP}} = \{f : \mathcal{X}_{\mathsf{PP}} \to \mathcal{Y}_{\mathsf{PP}}\}$.
- $\mathsf{KeyGen}(\mathsf{PP}, \mathsf{MSK}, f)$ is a p.p.t. algorithm that takes as input the public parameters PP, the master secret key MSK and a function $f \in \mathcal{F}_{\mathsf{PP}}$ and outputs a corresponding secret key SK_f.
- $\mathsf{Encrypt}(\mathsf{PP}, \mathsf{EK}, \vec{x})$ is a p.p.t. algorithm that takes as input the public parameters PP, the encryption key EK and an input message $\vec{x} \in \mathcal{X}_{\mathsf{PP}}$ and outputs a ciphertext $\mathsf{CT}_{\vec{x}}$.
- $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_f, \mathsf{CT}_{\vec{x}})$ is a deterministic algorithm that takes as input the public parameters PP, the secret key SK_f and a ciphertext $\mathsf{CT}_{\vec{x}}$ and outputs $f(\vec{x})$.

Definition 1 (Correctness)**.** A functional encryption scheme $\mathcal{FE}$ is correct if

$$\Pr\left[\begin{array}{l}(\mathsf{PP}, \mathsf{MSK}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^\kappa);\\ \qquad \forall f \in \mathcal{F}_{\mathsf{PP}}, \forall \vec{x} \in \mathcal{X}_{\mathsf{PP}},\ \mathsf{Decrypt}\big(\mathsf{KeyGen}(\mathsf{PP}, \mathsf{MSK}, f), \mathsf{Encrypt}(\mathsf{PP}, \mathsf{EK}, \vec{x})\big) \neq f(x)\end{array}\right]$$

is a negligible function of κ, where the probability is taken over the coins of Setup, KeyGen, and $\mathsf{Encrypt}$.

Remark 2. A functional encryption scheme $\mathcal{FE}$ may permit some *intentional leakage of information*. In this case, the secret SK_f or the ciphertext $\mathsf{CT}_{\vec{x}}$ may leak some legitimate information about the function f or the message $\vec{x}$ respectively. A common example of this type information is the length of the message $|\vec{x}|$ that is leaked in any public key encryption scheme. This is captured by [BSW11] via the "empty" key, by [AGVW13] by giving this information to the simulator directly and by [BF13] by restricting to adversaries who do not trivially break the system by issuing challenges that differ in such leakage. We use the approach of [AGVW13] and pass on any intentionally leaked information directly to the simulator.

2.2 Generic Group (GG) Model Overview

The generic group model [Nec94, Sho97] provides a method by which to study the security of algorithms that act oblivious of particular group representations. All algorithms obtain access to elements of the group via random "handles" (of sufficient length) and remain unaware of their actual representations. In our work we will require two groups $\mathcal{G}, \mathcal{G}_T$ (called the source and target group respectively) where $\mathcal{G}$ is equipped with a bilinear map $e : \mathcal{G} \times \mathcal{G} \to \mathcal{G}_T$. Algorithms with generic access to these may request group additions and inverses on either group, as well as pairings between elements in the source group.

Given group elements in $\mathcal{G}, \mathcal{G}_T$ an adversary will only be able to perform group exponentiations, multiplications, pairings and equality comparisons. Given this restricted way in which an adversary is allowed to access the groups $\mathcal{G}, \mathcal{G}_T$, he is only able to compute certain relations between elements which we call Admissible Relations, as defined below.

Definition 3 (Admissible Relations)**.** Consider a group $\mathcal{G}$ of order p, which supports a bilinear map $e : \mathcal{G} \times \mathcal{G} \to \mathcal{G}_T$. Let g and g_T be the generators of $\mathcal{G}$ and $\mathcal{G}_T$ respectively. Let $\{A_i\}_{i=1}^{\ell}$, $\{B_i\}_{i=1}^{m}$ be sets of formal variables taking values from $\mathbb{Z}_p$, representing the exponents of g and g_T respectively. Then we define *admissible relations* over the set $\{A_i\} \cup \{B_i\}$ to be all relations of the form $\sum_k \gamma_k A_k \stackrel{?}{=} 0$ or $\sum_k \gamma_k B_k + \sum_{i,j} \gamma_{i,j} A_i A_j \stackrel{?}{=} 0$ where $\gamma_k, \gamma_{i,j} \in \mathbb{Z}_p$.

Admissible relations capture the only relations an adversary can learn given only generic access to elements in the source and target group, described in the exponent for ease of exposition. Thus, exponentiation of a group element becomes multiplication in the exponent (eg. $(g^{A_k})^{\gamma_k}$ becomes $g^{\gamma_k A_k}$) , multiplication of two elements in the same group becomes addition in the exponent ($\prod_k (g^{A_k})^{\gamma_k}$ becomes $g^{\sum_k \gamma_k A_k}$) and pairing between source group elements becomes multiplication in the target group exponent ($e(g^{A_i}, g^{A_j})$ becomes $g_T^{A_i A_j}$).

We will also need the Schwartz Zippel lemma.

Theorem 4 (Schwartz Zippel Lemma)**.** *Let g_1, g_2 be any two* different *ℓ-variate polynomials with coefficients in field $\mathbb{Z}_p$. Let the degree of the polynomial $g_1 - g_2$ be t. Then,*

$$\Pr_{\{X_i\}_{i=1}^{\ell} \xleftarrow{\$} \mathbb{Z}_p} [g_1(X_1, \ldots, X_\ell) = g_2(X_1, \ldots, X_\ell)] \leq \frac{t}{p}$$

3 Wishful Security for Functional Encryption

In this section, we present the dream version security definition for Functional Encryption, which captures data hiding as well as function hiding in the strongest, most intuitive way via the ideal world-real world paradigm. This definition extends and generalizes the definition of [BSW11, BF13] to support function hiding in addition to data hiding (subsuming obfuscation), and encryption key in addition to public key. In the spirit of multiparty computation, this framework guarantees privacy for inputs of honest parties, whether messages or functions.

3.1 UC-style definition capturing Dream Security for FE

We fix the functionality of the system to be $\mathcal{F}_\kappa = \{f : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa\}$. We will refer to $\vec{x} \in \mathcal{X}$ as "message" and $f \in \mathcal{F}$ as "function" or "key". Our framework consists of an external environment Env who acts as an interactive distinguisher attempting to distinguish the real and ideal worlds, potentially in an adversarial manner.

Ideal-World. The ideal world functionality for FE is captured schematically in Figure 1. Formally, the ideal-world in a functional encryption system consists of the functional encryption oracle $\mathcal{O}$, the ideal world adversary (or simulator) $\mathcal{S}$, and an environment Env which is used to model all the parties external to the adversary. The adversary $\mathcal{S}$ and the environment Env are modeled as interactive p.p.t turing machines.

Throughout the interaction, $\mathcal{O}$ maintains a two-dimensional table $\mathcal{T}$ with rows indexed by messages $\vec{x}_1, \ldots \vec{x}_{\mathsf{rows}}$ and columns indexed by functions $f_1, \ldots, f_{\mathsf{cols}}$, and the entry corresponding to row $\vec{x}_i$ and column f_j is $f_j(\vec{x}_i)$. At a given time, the table contains all the message-key pairs seen in the interactions with $\mathcal{O}$ until then. $\mathcal{O}$ is initialized with a description of the functionality[1]. The environment Env interacts arbitrarily with the adversary $\mathcal{S}$. The interaction between the players is described below:

- **External ciphertexts and keys:**
  - **Ciphertexts:** Env may send $\mathcal{O}$ ciphertext commands $(\mathsf{CT}, \vec{x})$ upon which $\mathcal{O}$ creates a new row corresponding to $\vec{x}$, populates all the newly formed entries $f_1(\vec{x}), \ldots, f_{\mathsf{cols}}(\vec{x})$ and returns the newly populated table entires to $\mathcal{S}$.
  - **Keys:** Env may send $\mathcal{O}$ secret key commands (SK, f) upon which $\mathcal{O}$ creates a new column corresponding to f, populates all the newly formed entries $f(\vec{x}_1), \ldots, f(\vec{x}_{\mathsf{rows}})$ and returns the newly populated table entries to $\mathcal{S}$.

[1] For eg., for the inner product functionality $\mathcal{O}$ needs to be provided the modulus N

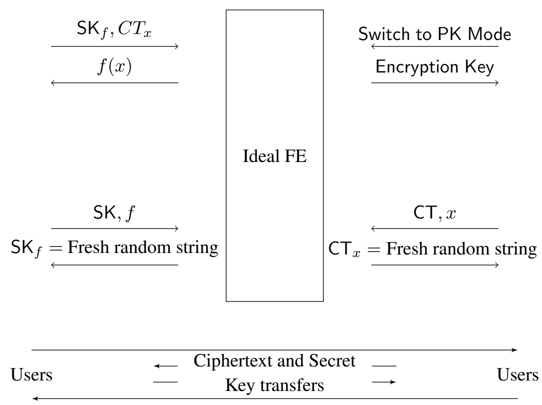

Figure 1: Users request keys for functions f and ciphertexts for inputs x, and receive independent random strings, chosen afresh in each invokation. In the public key mode, users receive the encryption key and can perform encryptions on their own for any values x. Decryption is equivalent to querying the oracle with $\mathsf{SK}_f, \mathsf{CT}_x$ and receiving the corresponding $f(x)$. Additionally, users may interact amongst themselves and exchange secret keys and ciphertexts arbitrarily.

- **Switch to public key mode:** Upon receiving a command ($\mathsf{PK\ mode}$) from Env, $\mathcal{O}$ forwards this message to $\mathcal{S}$. From this point on, $\mathcal{S}$ may query $\mathcal{O}$ for the function value corresponding to any message $\vec{x} \in \mathcal{X}$ of its choice, and any key in the system. Upon receiving command $(\vec{x}, \mathsf{keys})$, $\mathcal{O}$ updates $\mathcal{T}$ as follows: it adds a new row corresponding to $\vec{x}$, computes all the table entries for this row, and returns the newly populated row entries to $\mathcal{S}$.

At any point in time we allow $\mathcal{S}$ to obtain any intentionally leaked information (as defined in Remark 2) about all the messages and keys present in $\mathcal{T}$ from $\mathcal{O}$. Note that $\mathcal{S}$ may add any message or key of its choice to the system at any point in time through the adversarial environment Env with which it interacts arbitrarily. Hence, we omit modeling this option in our ideal world. We define $\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa)$ to be the view of Env in the ideal world.

Real-World. The real-world consists of an adversary $\mathcal{A}$, a system administrator Sys and external environment Env, which encompasses all external key holders and encryptors. The adversary $\mathcal{A}$ interacts with other players in the game through Sys. The environment Env may interact arbitrarily with $\mathcal{A}$. Sys obtains $(\mathsf{PP}, \mathsf{EK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$. PP is provided to Env and $\mathcal{A}$. The interaction between the players can be described as follows:

- **External ciphertexts and keys:**
  - **Ciphertexts:** Env may send Sys encryption commands of the form $(\mathsf{CT}, \vec{x})$ upon which, Sys obtains $\mathsf{CT}_{\vec{x}} = \mathsf{Encrypt}(\mathsf{EK}, \vec{x})$ sends $\mathsf{CT}_{\vec{x}}$ to $\mathcal{A}$.
  - **Keys:** Env may send Sys secret key commands of the form (SK, f) upon which, Sys obtains $\mathsf{SK}_f = \mathsf{KeyGen}(\mathsf{MSK}, f)$ and returns SK_f to $\mathcal{A}$.
- **Switch to public key mode:** Upon receiving a command ($\mathsf{PK\ mode}$) from Env, Sys sends EK to $\mathcal{A}$.

We define $\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa)$ to be the view of Env in the real world.

We say that a functional encryption scheme is *strongly simulation secure* in this framework, if for every real world adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for every environment Env:

$$\{\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa)\}_{\kappa \in \mathbb{N}} \overset{c}{\approx} \{\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa)\}_{\kappa \in \mathbb{N}}$$

While simulation based security has been shown impossible to achieve even for data privacy alone, we will show that the stronger definition presented above can be achieved against a large class of real world attacks, namely generic attacks. We believe that this provides evidence that FE schemes enjoy far greater security in practice.

4 Functional Encryption for Inner Products over Prime Order Groups

We present a new functional encryption scheme for inner products in the encryption key setting from prime order bilinear groups. Our scheme starts from the composite order scheme for inner product FE presented in [KSW08]. It then applies a series of transformations, as developed in [GKSW10, Fre10, OT08, OT09, Lew12], to convert it to a scheme over prime order groups. We will show our scheme to be be fully simulation secure in the generic group model.

To begin we define some notation that will be useful to us.

4.1 Group Notation required by constructions.

Notation for Linear Algebra over groups. When working over the prime order group $\mathcal{G}$, we will find it convenient to consider tuples of group elements. Let $\vec{v} = (v_1, \cdots, v_d) \in \mathbb{Z}_p^d$ for some $d \in \mathbb{Z}^+$ and $g \in \mathcal{G}$. Then we define $g^{\vec{v}} \doteq (g^{v_1}, \ldots, g^{v_d})$. For ease of notation, we will refer to $(g^{v_1}, \ldots, g^{v_d})$ by $(v_1, \cdots, v_d)$. This notation allows us to do scalar multiplication and vector addition over tuples of group elements as:

$$(g^{\vec{v}})^a = g^{(a\vec{v})} \text{ and } g^{\vec{v}} \cdot g^{\vec{w}} = g^{(\vec{v}+\vec{w})}$$

Finally we define a new function, $\vec{e}$ which deals with pairings two d-tuples of elements $\vec{v}, \vec{w}$ as:

$$\vec{e}(g^{\vec{v}}, g^{\vec{w}}) := \prod_{i=1}^{d} e(g^{v_i}, g^{w_i}) = e(g,g)^{\vec{v}\cdot\vec{w}}$$

where the vector dot product $\vec{v}\cdot\vec{w}$ in the last term is taken modulo p. We represent an element $g^a \in \mathcal{G}$ using the notation (a) and an element $e(g,g)^b \in \mathcal{G}_T$ using the notation $[b]$. Here g is assumed to be some fixed generator of $\mathcal{G}$.

Dual Pairing Vector Spaces. We will employ the concept of dual pairing vector spaces from [Lew12, OT08, OT09]. For a fixed dimension d, Let $\mathbb{B} = (\vec{b}_1, \ldots, \vec{b}_d), \mathbb{B}^* = (\vec{b}_1^*, \ldots, \vec{b}_d^*)$ be two random bases (represented as column vectors) for the vector[2] space $\mathbb{Z}_p^d$. Furthermore, they are chosen so that,

$$\begin{pmatrix} \vec{b}_1^T \\ \vdots \\ \vec{b}_d^T \end{pmatrix} \cdot \begin{pmatrix} \vec{b}_1^* & \cdots & \vec{b}_d^* \end{pmatrix} = \psi \cdot \mathbf{I}_{d\times d} \tag{1}$$

where $\mathbf{I}_{d\times d}$ is the identity matrix and $\psi \xleftarrow{\$} \mathbb{Z}_p$ is a uniformly distributed random variable. [Lew12] describes a standard procedure which allows one to pick such bases.

We use the notation $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^d)$ in the rest of this work to describe the selection of such basis vectors. Depending on the scheme we choose the value of d. We now establish further notation using $d = 3$ below. We use the formal variables $\left\{ \begin{pmatrix} b_{1i} \\ b_{2i} \\ b_{3i} \end{pmatrix} \right\}_{i=1}^{3}$ and $\left\{ \begin{pmatrix} b_{1i}^* \\ b_{2i}^* \\ b_{3i}^* \end{pmatrix} \right\}_{i=1}^{3}$ to denote the basis vectors of $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^3)$.

Furthermore, we overload vector notation (the usage will be clear from context) by associating with a three tuple of formal polynomials (a_1, a_2, a_3), the set of formal polynomials represented as:

$$\left\{ (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix}, (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{12} \\ b_{22} \\ b_{32} \end{pmatrix}, (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{13} \\ b_{23} \\ b_{33} \end{pmatrix} \right\}$$

and with the tuple $(a_1, a_2, a_3)^*$, the set of formal polynomials represented as:

$$\left\{ (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{11}^* \\ b_{21}^* \\ b_{31}^* \end{pmatrix}, (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{12}^* \\ b_{22}^* \\ b_{32}^* \end{pmatrix}, (a_1\ a_2\ a_3) \cdot \begin{pmatrix} b_{13}^* \\ b_{23}^* \\ b_{33}^* \end{pmatrix} \right\}$$

[2]Here, note that in the vein of Section 4.1, we are referring to vectors of *group elements*

The functionality $\mathcal{F} : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \{0,1\}$ is described as $\mathcal{F}(\vec{x},\vec{v}) = 1$ iff $\langle \vec{x} \cdot \vec{v} \rangle = 0 \mod p$, and 0 otherwise. The four algorithms $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ are defined as follows. Let $\mathsf{GroupGen}$ be a group generation algorithm which takes as input a security parameter κ and outputs a bilinear group of prime order p with $\mathsf{length}(p) = \kappa$.

Note that in the description of the scheme and in the proof, we will "work in the exponent" for ease of notation. Thus, g^a will be represented as a, exponentiation will become multiplication in log, i.e. $(g^a)^b$ will be represented as ab and multiplication will become addition, i.e. $g^a \cdot g^b$ will be represented as $a+b$. We recommend the reader refer to sections 4.1 and 4.1 for notational conventions.

- $\mathsf{Setup}(1^\kappa)$**:** Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Let $n \in \mathbb{Z}, n > 1$ be the dimension of the message space. Pick $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^3)$ and let $P, Q, R, R_0, H_1, R_1, H_2, R_2, \ldots, H_n, R_n \xleftarrow{\$} \mathbb{Z}_p$. Set,

$$\begin{aligned} \mathsf{PP} &= (p, \mathcal{G}, \mathcal{G}_T, e) \\ \mathsf{EK} &= \left(P \cdot \vec{b}_1, Q \cdot \vec{b}_2 + R_0 \cdot \vec{b}_3, R \cdot \vec{b}_3, \left\{H_i \cdot \vec{b}_1 + R_i \cdot \vec{b}_3\right\}_{i=1}^{i=n}\right) \\ \mathsf{MSK} &= \left(Q, \{H_i\}_{i=1}^{i=n}, \vec{b}_1, \vec{b}_2, \vec{b}_3, \vec{b}_1^*, \vec{b}_2^*, \vec{b}_3^*\right) \end{aligned}$$

- $\mathsf{Encrypt}(\mathsf{EK}, \vec{x})$**:** Let $\vec{x} = (x_1, \ldots, x_n), x_i \in \mathbb{Z}_p$. Choose random $s, \alpha \in \mathbb{Z}_p$ and random elements $\{r_i\}_{i=1}^n \in \mathbb{Z}_p$.

$$\mathsf{CT}_{\vec{x}} = \left\{C_0 = sP \cdot \vec{b}_1, C_i = \left\{s(H_i \cdot \vec{b}_1 + R_i \cdot \vec{b}_3) + \alpha \cdot x_i \cdot (Q \cdot \vec{b}_2 + R_0 \cdot \vec{b}_3) + r_i \cdot \vec{b}_3\right\}_{i=1}^{i=n}\right\}$$

- $\mathsf{KeyGen}(\mathsf{MSK}, \vec{v})$**:** Let $\vec{v} = (v_1, \ldots, v_n), v_i \in \mathbb{Z}_p$. Choose, $\{\delta_i\}_{i=1}^{i=n}, \zeta, Q_6, R_5 \xleftarrow{\$} \mathbb{Z}_p$ and construct $\mathsf{SK}_{\vec{v}} = (K_0, K_1, \ldots, K_n)$ as,

$$K_0 = \left(-\sum_{i=1}^{n} H_i \cdot \delta_i\right) \cdot \vec{b}_1^* + Q_6 \cdot \vec{b}_2^* + R_5 \cdot \vec{b}_3^*$$

  and,

$$\left\{K_i = \delta_i \cdot P \cdot \vec{b}_1^* + Q \cdot \zeta \cdot v_i \cdot \vec{b}_2^*\right\}_{i=1}^{i=n}$$

- $\mathsf{Decrypt}(\mathsf{SK}_{\vec{v}}, \mathsf{CT}_{\vec{x}})$**:** Compute $b = \vec{e}(C_0, K_0) \cdot \prod_{i=1}^{i=n} \vec{e}(C_i, K_i)$ and output 1 if $b = e(g,g)^0$ and 0 otherwise.

Intentionally leaked information as defined in Remark 2 for the above scheme is n, the length of the message and key space.

4.2 Correctness

Correctness of the scheme relies on the cancellation properties between the vectors in $\mathbb{B}$ and $\mathbb{B}^*$ as described in Eqn 1. We provide proof of correctness in Appendix B.

5 Proof of Security

We will now provide a proof that the scheme presented in Section 4 is fully simulation secure in the generic group model as per the framework presented in Section 3. We begin by describing the construction of our simulator.

5.1 Simulator Construction

Intuition. Broadly speaking, our simulator will run the adversary and provide secret keys and ciphertexts to him, as well as simulate the GG oracle. Our simulator maintains a table where it associates each group handle that it issues to the adversary with a formal polynomial. Through its interaction with the generic group oracle (played by $\mathcal{S}$), $\mathcal{A}$ may learn relations between the group handles that it obtains. Note that since we are in the GG model, $\mathcal{A}$ will only be able to learn admissible relations (Definition 3). Whatever dependencies $\mathcal{A}$ learns, $\mathcal{S}$ programs these using its table. To do this, it keeps track of what $\mathcal{A}$ is doing via its requests to the GG oracle, extracts necessary information from $\mathcal{A}$ cleverly where required and sets up these (formal polynomial) relations, thus ensuring that the real and ideal world views are indistinguishable. This is tricky in the public key mode, where the adversary may encrypt messages of its choice (using potentially bad randomness) and attempt to learn relations with existing keys using arbitrary generic group operations. In this case, the simulator needs to be able to extract the message from the adversary, obtain the relevant function values from the oracle, and program the dependencies into the generic group.

Formal Construction. Formally, the simulator $\mathcal{S}$ is specified as follows:

- **Initialization:** $\mathcal{S}$ constructs a table called *simulation table* to simulate the GG oracle $(p, \mathcal{G}, \mathcal{G}_T, e)$. A simulation table consists of two parts one each for the source group $\mathcal{G}$ and the target group $\mathcal{G}_T$ respectively. Each part is a list that contains two columns labelled formal polynomial and group handle respectively. Group handles are strings from $\{0,1\}^{2\kappa}$. A formal polynomial is a multivariate polynomial defined over $\mathbb{Z}_p$. We assume that there is a canonical ordering amongst the variables used to create the formal polynomial entries and thus each polynomial may be represented by a unique canonical representation.
- **Setup:** Upon receiving the public parameters, i.e. function description i.e p from $\mathcal{O}$, $\mathcal{S}$ executes the setup algorithm of the scheme as follows. He generates new group handles corresponding to the identity elements of $\mathcal{G}$ and $\mathcal{G}_T$. He associates these with the formal polynomials S_0 and T_0 respectively. $\mathcal{S}$ picks 18 new formal variables that represent the bases $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^3)$ as $\left\{ \left(b_{1i}, b_{2i}, b_{3i}\right)^\top \right\}_{i=1}^{3}$ and $\left\{ \left(b^*_{1i}, b^*_{2i}, b^*_{3i}\right)^\top \right\}_{i=1}^{3}$ as well as a new formal variable ψ. Next, $\mathcal{S}$ picks new formal variables $P, R, Q, R_0, \{H_i, R_i\}_{i=1}^{i=n}$[3]. He sets up the encryption key and master secret key by generating new group handles to represent the formal polynomials: $\mathsf{EK} = \big\{(P,0,0), (0,0,R), (0,Q,R_0), \{(H_i,0,R_i)\}_{i=1}^{n}\big\}$ and $\mathsf{MSK} = \big\{(0,Q,0), \{(H_i,0,0)\}_{i=1}^{i=n}, \vec{b}_1, \vec{b}_2, \vec{b}_3, \vec{b}^*_1, \vec{b}^*_2, \vec{b}^*_3\big\}$. He stores these associations in the simulation table.
- **Running the adversary:** $\mathcal{S}$ runs the adversary $\mathcal{A}(1^\kappa)$ and gives it the public parameters $\mathsf{PP} = (p, \mathcal{G}, \mathcal{G}_T, e)$. This amounts to $\mathcal{S}$ providing the adversary with oracle access to $\mathcal{G}, \mathcal{G}_T, e$ and sending him p.
- **Request for Public Key:** When $\mathcal{S}$ receives the command $\mathsf{PK\ mode}$ from $\mathcal{O}$, he sends the group handles of EK to $\mathcal{A}$.
- **External Ciphertexts and Keys:** At any time, $\mathcal{S}$ may receive a message of the form $\mathsf{MsgIdx}_{\vec{x}}, f_1(\vec{x}), \ldots, f_{\mathsf{cols}}(\vec{x})$ from $\mathcal{O}$. In response:
  - $\mathcal{S}$ follows the outline of the $\mathsf{Encrypt}$ algorithm as follows: He picks new formal variables $s, \alpha, \{x_i\}_{i=1}^n, \{r_i\}_{i=1}^n$ (all indexed by the particular index $\mathsf{MsgIdx}_{\vec{x}}$, dropped here for notational convenience). He then constructs the formal polynomials associated with the following 3-tuples:
  $$C = \Big\{C_0 = (sP,\ 0,\ 0), \big\{C_i = (sH_i,\ Q\alpha x_i,\ r_i)\big\}_{i=1}^n\Big\} \tag{2}$$
  and adds each formal polynomial thus generated in C to the simulation table along with a new group handle.
  - $\mathcal{S}$ then programs the generic group to incorporate the function values $f_1(\vec{x}), \ldots, f_{\mathsf{cols}}(\vec{x})$ that were received. To do this, $\mathcal{S}$ retrieves the formal polynomials associated with all the keys in the table $\{K_0^j, K_1^j, \ldots, K_n^j\}_{j\in[\mathsf{cols}]}$. Then, for each j, he computes the formal polynomials associated with the decrypt operation between C and K^j, i.e. $b = \vec{e}(C_0, K_0^j) \cdot \prod_{i=1}^{i=n} \vec{e}(C_i, K_i^j)$. If $f_j(\vec{x}) = 0$, he sets the resultant expression to correspond to the group handle for the identity element in the target group. Else, he generates a new group handle and stores the resultant expression to correspond to it.

[3]Recall that n is part of intentionally leaked information (Remark 2) in our scheme

– $\mathcal{S}$ then sends the group handles corresponding to C to $\mathcal{A}$.

He acts analogously in the case of a $\mathsf{KeyIdx}_j, f_j(\vec{x}_1), \ldots, f_j(\vec{x}_{\mathsf{rows}})$ message by following the KeyGen algorithm to generate formal polynomials corresponding to a new key and programming the decrypt expressions to correspond to the received function values.

- **Generic Group Operations:** At any stage, $\mathcal{A}$ may request generic group operations from $\mathcal{S}$ by providing the corresponding group handle(s) and specifying the requested operation, such as pairing, identity, inverse or group operation. In response, $\mathcal{S}$ looks up its simulation table for the formal polynomial(s) corresponding to the specified group handle(s), computes the operation between the formal polynomials, simplifies the resultant expression and does a reverse lookup in the table to find a group handle corresponding to the resultant polynomial. If it finds it, $\mathcal{S}$ will return this group handle to $\mathcal{A}$, otherwise it randomly generates a new group handle, stores it in the simulation table against the resultant formal polynomial, and returns this to $\mathcal{A}$. For more details, we refer the reader to Appendix D.

  Tracking admissible relations learnt by $\mathcal{A}$: If $\mathcal{A}$ requests generic group operations to compute a polynomial involving a term $\psi Q^2\mathsf{expr}$ where expr is an expression containing a term of the form $\sum_{i=1}^{n} c_i v_i$ for some constant $c_i \in \mathbb{Z}_p$, then $\mathcal{S}$ considers this as a function evaluation by $\mathcal{A}$ on message that he encrypted himself. He extracts the message $\vec{x} = c_1, \ldots, c_n$. $\mathcal{S}$ then sends the message $(\vec{x}, \mathsf{keys})$ to $\mathcal{O}$. Upon receiving $\mathsf{MsgIdx}_{\vec{x}}, f_1(\vec{x}), \ldots, f_{\mathsf{cols}}(\vec{x})$ from $\mathcal{O}$, $\mathcal{S}$ computes the decrypt expressions for the extracted message with all the keys and programs the linear relations in the generic group oracle as in the previous step.

In Appendix E, we show that the real and ideal worlds are indistinguishable to Env. Formally, we prove the following theorem:

Theorem 5. *The simulator $\mathcal{S}$ constructed in Section 5.1 is such that for all adversaries $\mathcal{A}$, for all* Env *with auxiliary input z,* $\{\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*} \approx \{\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*}$ *in the generic group model.*

High Level Overview of Proof: Intuitively, the only way the environment distinguishes between the real and ideal world is when he obtains group elements from the GG oracle which differ in satisfying some admissible relation between the two worlds. We argue that this cannot happen. To do so, we begin by by enumerating all admissible relations present between the elements of the real world scheme. Then we prove that our simulator accounted for all these relations. To see how the simulator accounted for all possible admissible relations, note that whenever an external party sends its ciphertext or key to the adversary, modeled by Env sending message $(\mathsf{CT}, \vec{x})$ (or (SK, f)) to $\mathcal{O}$, the simulator obtains from $\mathcal{O}$ all possible function evaluations between the newly added message, say $\vec{x}^*$ and all existing keys in the system. It creates the required formal variables representing the ciphertext CT^* of $\vec{x}^*$, and computes via polynomial arithmetic the decrypt operation corresponding to $\mathsf{CT}_{\vec{x}^*}$ and $\mathsf{SK}_{f_1}, \ldots, \mathsf{SK}_{f_{\mathsf{cols}}}$. Lets say that $f_j(\vec{x}^*) = 0$ as informed by $\mathcal{O}$. Then, $\mathcal{S}$ sets the decrypt expression $\vec{e}(C_0^*, K_0^j) \cdot \prod_{i=1}^{i=n} \vec{e}(C_i^*, K_i^j)$ equal to the identity element of the target group, and returns the group handles for CT^* to the adversary. $\mathcal{S}$'s job is much more complex in the public key mode, where $\mathcal{A}$ has the encryption key and may perform encryptions of vectors of his choice. Note that the adversary may use no randomness while performing these encryptions, and may indeed behave in an obfuscated manner – he can carry out an arbitrarily obfuscated sequence of group operations, which may implicitly be encrypting and decrypting values. Our simulator keeps track of his operations via his queries to the generic group oracle, and if $\mathcal{S}$ encounters a term of the form $\psi \cdot Q^2 \cdot \sum_{i=1}^{n} c_i v_i$ for some constant $c_i \in \mathbb{Z}_p$, then $\mathcal{S}$ extracts the message $\vec{x}^* = c_1, \ldots, c_n$, queries $\mathcal{O}$ for function values corresponding to $\vec{x}^*$ and programs them into the generic group as above. Note that $\mathcal{S}$ can also attempt to extract $\vec{x}^*$ when $\mathcal{A}$ attempts to encrypt it, however it is cleaner to describe this extraction operation during $\mathcal{A}$'s decrypt attempts. This also makes the simulator more efficient, since it need not program dependencies that $\mathcal{A}$ does not check via computing the decrypt operation. $\mathcal{S}$ can handle key updates in a similar fashion. The detailed proof is provided in Appendix E.

5.2 Concrete parameters and analysis of our scheme

In the full proof from Appendix E we observe that the only case for distinguishability between real and ideal worlds is the hybrid where we move from Generic Group elements to polynomials in formal variables.

Thus, we have that if the adversary receives q group elements in total from the groups $\mathbb{G}, \mathbb{G}_T$, then the probability that he would be able to distinguish between the real and ideal worlds is

$$q\frac{(q-1)}{2}\frac{t}{p}$$

where t is the maximum degree of any formal variable polynomial that could be constructed in our cryptosystem. It is a maximum of 3 for each element in the source group for our FE scheme and thus t = 6 considering possible pairings. p is the order of the group.

5.3 Practical considerations

We observe that every pairing in our scheme is between some element of the ciphertext and an element of the key. Thus suppose $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a set of groups with an asymmetric bilinear map. Then it is easy to see that our scheme extends to this setting by choosing the ciphertext elements from $\mathbb{G}_1$ and the key elements from $\mathbb{G}_2$. Furthermore, our security proof also extends to this setting, as a generic group adversary is now further restricted in the set of queries he could make. This allows for a scheme in the faster setting of asymmetric bilinear maps.

We also note that our scheme is shown to be secure against generic attacks and that non-generic attacks do exist in all known bilinear groups. However a long list of previous research focuses on constructing elliptic curves where the complexity of any non-generic attack is worse than generic attacks [FST10, Fre06, AFCK$^+$13, Cos12, BF01] making our work relevant and meaningful. These constructions are practical as well. Hence we believe that FE constructions over suitably chosen elliptic curve groups have the potential of being practically secure.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology Eprint Arxiv, 2013. `http://eprint.iacr.org/2013/689.pdf`.

[AFCK+13] DiegoF. Aranha, Laura Fuentes-Castaeda, Edward Knapp, Alfred Menezes, and Francisco Rodrguez-Henrquez. Implementing pairings at the 192-bit security level. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195. Springer Berlin Heidelberg, 2013.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

[AGVW13] Shweta Agrawal, Sergey Gurbanov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BF13] Manuel Barbosa and Pooya Farshim. On the semantic security of functional encryption schemes. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography, PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 143–161. Springer Berlin Heidelberg, 2013.

[BO12] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. Cryptology ePrint Archive, Report 2012/515, 2012.

[BRS13] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Crypto*, 2013.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[CIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO*, 2013.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[Cos12] Craig Costello. Particularly friendly members of family trees. *IACR Cryptology ePrint Archive*, 2012:72, 2012.

[CRV10] Ran Canetti, Guy Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, 2010.

[Fre06] David Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In *Proceedings of the 7th international conference on Algorithmic Number Theory*, ANTS'06, pages 452–465, Berlin, Heidelberg, 2006. Springer-Verlag.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[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*, pages 578–602, 2014.

[GGH$^+$13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. `http://eprint.iacr.org/`.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.

[GKP$^+$13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

[GKP$^+$13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GKSW10] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2010.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013. To appear.

[KM06] Neal Koblitz and Alfred Menezes. Another look at generic groups. In *Advances in Mathematics of Communications*, pages 13–28, 2006.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.

[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*, pages 62–91, 2010.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. In *MATHEMATICAL NOTES*, volume 55, 1994.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`.

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008.

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *Asiacrypt*, 2009.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt*, pages 256–266. Springer-Verlag, 1997.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.

[SW] Amit Sahai and Brent Waters. Functional encryption:beyond public key cryptography. Power Point Presentation, 2008. http://userweb.cs.utexas.edu/bwaters/presentations/ files/functional.ppt.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[vol] http://www.voltage.com/.

[Wat12] Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

A Notation

We say that a function $f : \mathbb{Z}^+ \to \mathbb{R}^+$ is negligible if $f(\lambda) \in \lambda^{-\omega(1)}$. For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over some set Ω we define the statistical distance $\mathrm{SD}(\mathcal{D}_1, \mathcal{D}_2)$ as

$$\mathrm{SD}(\mathcal{D}_1, \mathcal{D}_2) := \frac{1}{2} \sum_{x \in \Omega} \big| \Pr_{\mathcal{D}_1}[x] - \Pr_{\mathcal{D}_2}[x] \big|$$

We say that two distribution ensembles $\mathcal{D}_1(\lambda)$ and $\mathcal{D}_2(\lambda)$ are statistically close or statistically indistinguishable if $\mathrm{SD}\big(\mathcal{D}_1(\lambda), \mathcal{D}_2(\lambda)\big)$ is a negligible function of λ.

We say that two distribution ensembles $\mathcal{D}_1(\lambda), \mathcal{D}_2(\lambda)$ are computationally indistinguishable, denoted by $\stackrel{c}{\approx}$, if for all probabilistic polynomial time turing machines $\mathcal{A}$,

$$\Big| \Pr[\mathcal{A}(1^\lambda, \mathcal{D}_1(\lambda)) = 1] - \Pr[\mathcal{A}(1^\lambda, \mathcal{D}_2(\lambda)) = 1] \Big|$$

is a negligible function of λ.

B Correctness of Inner Product Scheme

For any, $\mathsf{SK}_{\vec{v}}, \mathsf{CT}_{\vec{x}}$ the pairing evaluations in the decryption part of our scheme proceed as follows. Terms that are marked $(\times)$ are ones that we do not care about:

$$
\begin{aligned}
\vec{e}(C_0, K) &= \vec{e}\Bigg((sP \cdot \vec{b}_1), ((-\sum_{i=1}^{n} H_i \cdot \delta_i) \cdot \vec{b}_1^* + Q_6 \cdot \vec{b}_2^* + R_5 \cdot \vec{b}_3^*)\Bigg) \\
&= (-sP\sum_{i=1}^{n} H_i\delta_i) \cdot (\vec{b}_1^T \cdot \vec{b}_1^*) + (\times)(\vec{b}_1^T \cdot \vec{b}_2^*) + (\times)(\vec{b}_1^T \cdot \vec{b}_3^*) \\
&= \psi(-sP\sum_{i=1}^{n} H_i\delta_i) \text{ (by Equation 1)} \\
\vec{e}(C_i, K_i) &= \vec{e}(\Big(s(H_i \cdot \vec{b}_1 + R_i \cdot \vec{b}_3) + \alpha \cdot x_i \cdot (Q \cdot \vec{b}_2 + R_0 \cdot \vec{b}_3) + r_i \cdot \vec{b}_3\Big), \\
&\quad \Big(\delta_i \cdot P \cdot \vec{b}_1^* + Q \cdot \zeta \cdot v_i \cdot \vec{b}_2^*\Big)) \\
&= (sH_i\delta_iP)\vec{b}_1^T\vec{b}_1^* + (\alpha x_i Q \cdot Q\zeta v_i)\vec{b}_2^T\vec{b}_2^* + (\times)(\vec{b}_1^T \cdot \vec{b}_2^* + \vec{b}_2^T \cdot \vec{b}_1^* + \vec{b}_3^T \cdot \vec{b}_1^* + \vec{b}_3^T \cdot \vec{b}_2^*) \\
&= \psi(sH_i\delta_iP + \alpha\zeta Q^2 x_i v_i) \text{ (by Equation 1)} \\
\text{Thus,} \quad & \vec{e}(C_0, K) \cdot \prod_{i=1}^{i=n} \vec{e}(C_i, K_i) \\
&= \psi(-sP\sum_{i=1}^{n} H_i\delta_i) + \sum_{i=1}^{n} \big(\psi(sH_i\delta_iP + \alpha\zeta Q^2 x_i v_i)\big) \\
&= \psi Q^2 \alpha\zeta(\sum_{i=1}^{n} x_i v_i)
\end{aligned}
$$

When $\sum_{i=1}^{n} x_i v_i$ is 0 mod p, the final answer is always the identity element of the target group and when it is not, the answer evaluates to a random element in the target group (as $\psi, Q, \alpha, \zeta \xleftarrow{\$} \mathbb{Z}_p$).

C Comparison

First, we note that no simulation based definition [BSW11, O'N10, AGVW13, BF13] considers the notion of key-hiding or function privacy. Another difference is in the way we handle the "challenge message" distribution. In [BF13, BSW11] the adversary outputs a distribution of messages $\mathcal{M}$ from which a challenge encryption is chosen. The definition of [BF13] allows for even the random coins used in this process of choosing the challenge to be provided to the adversary (in the form of a history h). An adversary may output any challenge distribution of his choice.

Our model tackles this problem along the veins of UC security. It allows for an arbitary environment to send any sequence of messages or secret-keys to the adversary at any point in time in the interaction. This encompasses the challenge phase of the models presented in [BF13, BSW11].

Certain information such as the length of the encrypted message or the identity to which a message is encrypted (in the case of IBE) may be revealed by an encryption scheme. Given that this information is harmless, it being available to the adversary must be made trivial. [BSW11] handles this case by the presence of a special key ϵ which reveals all the intentional information about an encrypted message. [BF13] handles this case by ensuring that the challenge message distribution only contains messages that reveal "equal amounts of information". They ensure this by defining a relation R on pairs of messages (called potential leakage relation in [BF13], see Section 3).

Our modeling generalizes both these definitions and clarifies what intentional information is leaked in an ideal world, i.e information about all the messages and secret keys via any arbitrary function that belongs an information leakage set R. Our model is also the *first* model to capture the fact that intentional leakage is necessary and may happen with respect to secret keys present in the system as well.

D Generic Group Operations

Whenever $\mathcal{A}$ requests the GG oracle for group operations corresponding $\mathcal{G}, \mathcal{G}_T$ or the pairing operation e, $\mathcal{S}$ does the following:

1. **Request for Identity:** When $\mathcal{A}$ requests for the identity element of the group $\mathcal{G}$, $\mathcal{S}$ looks up the simulation table for the formal polynomial 0 in the part that corresponds to $\mathcal{G}$ and returns the group handle corresponding to it to the adversary. He acts analogously with request for the identity of $\mathcal{G}_T$.

2. **Request for Inverses:** When $\mathcal{A}$ requests the inverse of a group handle h in $\mathcal{G}$, $\mathcal{S}$ looks up the formal polynomial associated with h from the simulation table, denoted by $\hat{h}$. He computes the polynomial $(-1)\hat{h}$ and looks for it in the simulation table. If he finds an associated group handle, he returns it to $\mathcal{A}$. If not, he generates a new group handle and adds the association between $(-1)\hat{h}$ and the generated handle. He returns the newly generated handle to $\mathcal{A}$. He acts analogously for requests involving handles in $\mathcal{G}_T$.

3. **Request for group operation:** When $\mathcal{A}$ requests a group operation on two group elements $h, \ell \in \mathcal{G}$, $\mathcal{S}$ looks them both up in the simulation table and obtains their corresponding formal polynomials $\hat{h}$ and $\hat{\ell}$. He computes the formal polynomial $\hat{q} = \hat{h} + \hat{\ell}$. $\mathcal{S}$ then does a look up in the simulation table for the polynomial $\hat{q}$ and if it finds an associated group handle, returns it to $\mathcal{A}$. If it doesn't find a group handle corresponding to $\hat{q}$, it generates a new group handle and adds this association to the simulation table and returns the newly generated handle to $\mathcal{A}$. It acts analogously for requests involving handles in $\mathcal{G}_T$.

4. **Request for Pairing operation:** When $\mathcal{A}$ requests a pairing operation on two group elements $h, \ell \in \mathcal{G}$, $\mathcal{S}$ looks them both up in the simulation table obtains their corresponding formal polynomials $\hat{h}$ and $\hat{\ell}$. He computes the formal polynomial $\hat{q} = \hat{h} \times \hat{\ell}$, where $\times$ denotes polynomial multiplication. $\mathcal{S}$ then does a look up in the simulation table for the polynomial $\hat{q}$ and if it finds an associated group handle, returns it to $\mathcal{A}$. If it doesn't find a group handle corresponding to $\hat{q}$, it generates a new group handle and adds this association to the simulation table and returns the newly generated handle to $\mathcal{A}$.

D.1 Simplifying expressions occuring in generic group computations

We describe here the $\mathsf{Simplify}$ step.

$\mathsf{Simplify}$: Let $\hat{\ell}$ be the formal polynomial computed by $\mathcal{S}$ in any generic group operation. We first handle the 9 constraints satisfied by $\vec{b}_1, \vec{b}_1^*, \cdots, \vec{b}_3, \vec{b}_3^*$ as per Eqn 1. Consider the relation $b_{11}b_{11}^* + b_{12}b_{12}^* + b_{13}b_{13}^* = \psi$. $\mathcal{S}$ writes $\hat{\ell} = Ab_{11}b_{11}^* + Bb_{12}b_{12}^* + Cb_{13}b_{13}^* + D$ where D has no monomials divisible by $b_{11}b_{11}^*$ or $b_{12}b_{12}^*$ or $b_{13}b_{13}^*$, breaking ties (if any) arbitrarily. Then he writes $\hat{\ell} = A\psi + (B - A)b_{12}b_{12}^* + (C - A)b_{13}b_{13}^* + D$ using the above constraint.

Next, $\mathcal{S}$ writes $\hat{\ell}$ as $\hat{\ell} = \psi Q^2 \zeta_j A_j + B$ where where $\zeta[\mathsf{KeyIdx}_j]$ (denoted hence forth by just ζ_j) is the formal variable corresponding to KeyIdx_j from $\mathcal{O}$, and B has no monomials that are divisible by $\psi Q^2 \zeta_j$. It then behaves differently in the following two cases:

1. EK **was not sent to** $\mathcal{A}$ **(Encryption Key setting):** For the ciphertext corresponding MsgIdx_k, let $\alpha[\mathsf{MsgIdx}_k]$ (henceforth denoted by α_k) be the corresponding formal variable used by $\mathcal{S}$ when generating the ciphertext. $\mathcal{S}$ writes $A_j = \alpha_k \cdot \sum_{i=1}^n \theta_i x_i^k f_i^j + D$ where D has no monomials divisible by $\alpha_k x_i^k f_i^j$ for any i. If each θ_i is not zero, then $\mathcal{S}$ rewrites $A_j = \alpha_k \theta_1 \cdot \left(\sum_{i=1}^n x_i^k f_i^j\right) + \alpha_k \left(\sum_{i=2}^n (\theta_i - \theta_1) x_i^k f_i^j\right) + D$. If $f(\mathsf{MsgIdx}_k, \mathsf{KeyIdx}_j)$. If the output is 0, he sets this portion of A_j to 0 else, he does not change the expression. He repeats this procedure for all ciphertexts that he issued to $\mathcal{A}$.

2. EK **was issued to** $\mathcal{A}$ **(Public Key setting):** In the case when EK was issued, $\mathcal{S}$ first does all of the operations mentioned in the previous case. In addition to these, $\mathcal{S}$ writes A_j as $A_j = \sum_{i=1}^{i=n} m_i f_i^j + D$ where m_i, possibly 0, is in $\mathbb{Z}_p$ and D contains no mononomials of the form θf_i^j for any i and any $\theta \in \mathbb{Z}_p$. If atleast one m_i is not zero, then simu constructs the message $m = (m_1, \ldots, m_n)$. $\mathcal{S}$ records this message in a table along with the KeyIdx_j. This table contains two columns, one with message, index pairs and the second column contains a formal variable. For each message prev_j in this list corresponding to KeyIdx_j, $\mathcal{S}$ queries the oracle for the function value of $\mathcal{T}[\mathsf{prev}_j - m, \mathsf{KeyIdx}_j]$. If any of the return values is 0, $\mathcal{S}$ replaces the expression above in A_j

by the corresponding formal variable found in this secondary table. If not, he generates a new formal variable Ω, adds this entry to the secondary table and rewrites A_j as $\Omega + D$.

E Analysis of Real and Ideal worlds

We now provide a proof that $\mathcal{S}$ constructed above works correctly. Intuitively, the only way the environment distinguishes between the real and ideal world is when he obtains group elements from the GG oracle which differ in satisfying some admissible relation between the two worlds. In this section we will prove that one cannot discover admissible relations in the prime order scheme presented in Section 4 apart from the ones that are accounted for by our simulator $\mathcal{S}$. At a high level, we do this by enumerating all admissible relations present between the elements of the real world scheme and then proving that our simulator programmed all these relations.

Recall that our scheme is based on the composite order scheme of [KSW08], but unlike in [KSW08] our scheme is based on prime order bilinear groups and thus our proof is more involved. We proceed to enumerate all possible admissible relations in two phases.

First, we show that if there are any admissible relations between individual group elements of our scheme, then certain relations are satisfied over $\mathbb{Z}_p^3$.

Theorem 6. *(***Translation of Admissible Relations***) Let $A \subseteq \mathbb{Z}_p^3$ be any set. Let $(\mathbb{B}, \mathbb{B}^*) \leftarrow Dual(\mathbb{Z}_p)$ be a random dual orthonormal basis represented by formal variables satisfying Eqn 1. Define $C \doteq \{x\vec{b}_1 + y\vec{b}_2 + z\vec{b}_3 | (x,y,z) \in A\} \cup \{x\vec{b}_1^* + y\vec{b}_2^* + z\vec{b}_3^* | (x,y,z) \in A\}$. Then any admissible relation amongst the elements of C results in an admissible relation (with the same coefficients) amongst corresponding elements of A.*

Proof. Recall from our notation that the vector of elements $x\vec{b}_1 + y\vec{b}_2 + z\vec{b}_3$ is just the three elements denoted by the row vector,

$$\left\{ \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix}, \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} b_{12} \\ b_{22} \\ b_{32} \end{pmatrix}, \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} b_{13} \\ b_{23} \\ b_{33} \end{pmatrix} \right\}$$

Denote the column vectors of the basis matrix by, $\left\{ \vec{g}_i = \begin{pmatrix} b_{1i} \\ b_{2i} \\ b_{3i} \end{pmatrix} \right\}_{i=1}^{3}$ and $\left\{ \vec{g}_i^* = \begin{pmatrix} b_{1i}^* \\ b_{2i}^* \\ b_{3i}^* \end{pmatrix} \right\}$

We have that the new group elements added to C due to the element $\vec{a}^T = \begin{pmatrix} x & y & z \end{pmatrix}$ are $a^T \cdot \vec{g}_1, a^T \cdot \vec{g}_2, a^T \cdot \vec{g}_3$ and $a^T \cdot \vec{g}_1^*, a^T \cdot \vec{g}_2^*, a^T \cdot \vec{g}_3^*$.

Relations in the source group. Consider any admissible relation in the source group of the form $\sum \alpha_i \vec{a}_i^T \vec{g}_1 + \sum \beta_i \vec{c}_i^T \vec{g}_1^* \cdots + \sum \eta_i \vec{t}_i^T \vec{g}_3 + \sum \gamma_i \vec{d}_i^T \vec{g}_3^* = 0$. Since the above equation is 0 as a formal expression and the variables $\vec{g}_1, \vec{g}_1^*, \ldots, \vec{g}_3, \vec{g}_3^*$ are different we have that each of the independent components sum up to 0. Thus, $\sum \alpha_i a_i^T = 0$, etc. yielding a corresponding relation between elements in A.

Relations in the target group. Next consider admissible relations that involve pairings and are thus in the target group. Recall that from the choice of the vectors $\vec{b}_1, \vec{b}_2, \vec{b}_3, \vec{b}_1^*, \vec{b}_2^*, \vec{b}_3^*$, the only admissible relations that they satisfy are of the form,

$$\begin{pmatrix} \vec{b}_1^T \\ \vec{b}_2^T \\ \vec{b}_3^T \end{pmatrix} \cdot \begin{pmatrix} \vec{b}_1^* & \vec{b}_2^* & \vec{b}_3^* \end{pmatrix} = \begin{pmatrix} \psi & 0 & 0 \\ 0 & \psi & 0 \\ 0 & 0 & \psi \end{pmatrix} \tag{3}$$

And rewriting the matrices with the column vectors $\vec{g}_1$ etc. we have that

$$\begin{pmatrix} \vec{g}_1 & \vec{g}_2 & \vec{g}_3 \end{pmatrix} \cdot \begin{pmatrix} \vec{g}_1^{*T} \\ \vec{g}_2^{*T} \\ \vec{g}_3^{*T} \end{pmatrix} = \sum_{i=1}^{i=3} \vec{g}_i \vec{g}_i^{*T} = \sum_{i=1}^{i=3} \vec{g}_i^* \vec{g}_i^T = \begin{pmatrix} \psi & 0 & 0 \\ 0 & \psi & 0 \\ 0 & 0 & \psi \end{pmatrix} \tag{4}$$

Consider an arbitrary admissible relationship in the target group of the form

$$\sum \alpha_i(a_i^T \cdot \vec{g}_1)(c_i^T \cdot \vec{g}_2) + \sum \beta_i(d_i^T \cdot \vec{g}_1)(t_i^T \cdot \vec{g}_1^*) \cdots + \sum \gamma_i(h_i^T \cdot \vec{g}_3)(\ell_i^T \cdot \vec{g}_3^*) = 0$$

where the relation (w.l.o.g) has a summation with every possible pair $\vec{g}_1\vec{g}_2, \vec{g}_1\vec{g}_1^*, \vec{g}_1\vec{g}_1$ etc. (a total of 36 such terms for every possible combination). Rearranging terms inside each summation and looking at some arbitrary pair, say $\vec{g}_1, \vec{g}_2$, we have that,

$$\begin{aligned}\sum \alpha_i(a_i^T \cdot \vec{g}_1)(c_i^T \cdot \vec{g}_2) &= \sum \alpha_i(a_i^T \cdot \vec{g}_1)(\vec{g}_2^T \cdot c_i) \\ &= \sum \alpha_i(a_i^T) \cdot (\vec{g}_1 \cdot \vec{g}_2^T) \cdot c_i\end{aligned}$$

and similarly for other terms as well. $\vec{g}_1\vec{g}_2^T$ is a 3×3 matrix. We focus only on the terms in its main diagonal. It consists of the following three quadratic polynomials: $b_{11}b_{12}, b_{21}b_{22}, b_{31}b_{32}$. These terms will not appear elsewhere in any other term except for $(\vec{g}_1\vec{g}_2^T)^T = \vec{g}_2\vec{g}_1^T$ and when it does appear in $\vec{g}_2\vec{g}_1^T$ it appears in the main diagonal as well. The coefficients on the diagonal are a_1c_1, a_2c_2, a_3c_3 respectively.

Since this summation is 0 as a formal polynomial, we have that each individual coefficient of $b_{11}b_{12}, b_{21}b_{22}, b_{31}b_{32}$ must be 0. Thus we have that $\sum \alpha_i(a_i^T c_i) = 0$ yielding a relation in A.

However we are not done, since some of the vectors, say $\vec{g}_1, \vec{g}_1^*$ etc., satisfy a relationship namely Eqn 4. We handle this case by the following analysis.

Suppose we have that

$$\sum \alpha_i a_i^T(\vec{g}_1\vec{g}_1^{*T})c_i + \cdots \sum \beta_i t_i^T(\vec{g}_3\vec{g}_3^{*T})d_i = 0$$

From Equation 4, we may write $\vec{g}_1\vec{g}_1^{*T} = \psi \cdot \mathbf{I}_{3\times 3} - \sum_{i=2}^{3} \vec{g}_i\vec{g}_i^{*T}$. Since the equation is identically 0 and the formal variable ψ does not appear anywhere else, we get that its coefficient $\sum \alpha_i a_i^T c_i = 0$. Similarly we obtain $\sum \beta_i t_i^T d_i = \sum \alpha_i a_i^T c_i = 0$ and so on. Thus, we get $\sum \alpha_i a_i^T c_i + \cdots + \sum \beta_i t_i^T d_i = 0$. This concludes the proof that any admissible relation over the elements of C results in an identical admissible relation over corresponding elements of A. □

Next, we have the following theorem that enumerates all admissible relations between elements in the real world.

Theorem 7. *(**Admissible Relations in Our Scheme.**) In the real world, $\mathcal{A}$ sees the following admissible relations:*

1. *Before* EK *is given: In this case, the only admissible relations present in the system are relations corresponding to* $\mathsf{Decrypt}(\mathsf{MsgIdx}_i, \mathsf{KeyIdx}_j)$ *for messages and keys such that* $\mathcal{T}[\mathsf{MsgIdx}_i, \mathsf{KeyIdx}_j] = 0$ *and any linear combinations of such expressions.*

2. *After* EK *is given: In this case, in addition to the above relations, additional admissible relations exist for any message* $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$ *such that* $\mathcal{T}[\vec{m}, \mathsf{KeyIdx}_j] = 0$ *for some key corresponding to* KeyIdx_j *and any linear combinations of such expressions.*

Proof. Our proof handles each case separately.
Case 1: Before EK is given. Suppose that $\mathcal{A}$ requests a total of m_k keys and m_c cipher texts. Then, the group elements that $\mathcal{A}$ has can be summarized using 3-tuples as follows.
The secret keys:

$$\left\{\mathsf{SK}^j = \left\{K^j = \big(\sum_{i=1}^{n}(-H_i \cdot \delta_i^j),\ Q_6^j,\ R_5^j\big), \{K_i^j = (P\delta_i^j,\ Qf^j v_i^j, 0)\}_{i=1}^n\right\}\right\}_{j=1}^{m_k} \tag{5}$$

The ciphertexts:

$$\left\{C^j = \left\{C_0^j = (s^j P,\ 0,\ 0), \{C_i^j = (s^j H_i,\ Q\alpha^j x_i^j,\ r_i^j)\}_{i=1}^n\right\}\right\}_{j=1}^{m_c} \tag{6}$$

Note that here, for ease of notation, we have collapsed the component $s^j R_i^j + r_i^j + R_0 \alpha^j x_i^j$ to just r_i^j since this is a formal polynomial of which r_i^j is a fresh new formal variable. This would suffice for us in the proof. In what follows we will use the character $*$ to denote "anything", i.e. whenever it is irrelevant what the exact value of the element is. For example $0 \times * = 0$.

Now let us look at possible admissible relations in the source group containing the 3-tuples corresponding to keys. Note that:

1. Linear relations containing an element $(\sum_{i=1}^n (-H_i \delta_i^j),\ Q_6^j,\ R_5^j)$ cannot exist because R_5^j is not present in any of the other elements in Eqns 5 and 6. Thus it would never get canceled unless the coefficient of the above element is 0.

2. Linear relations containing any of the elements $(P\delta_i^j,\ Qf^j v_i^j,\ 0)$ cannot exist because for each i the variable δ_i^j occurs only in the element $(\sum_{i=1}^n (-H_i \delta_i^j),\ Q_6^j,\ R_5^j)$ whose coefficient is 0 as seen above, and does not exist in any of the other elements in Eqns 5 and 6. Thus any linear relation cannot contain this term with nonzero coefficient.

Next consider possible admissible relations in the source group containing the 3-tuples corresponding to ciphertexts. Note that:

1. Linear relations containing any of the elements $C_i^j = (s^j H_i, Q\alpha^j x_i^j, r_i^j)$ cannot exist because r_i^j is not present in any of the other elements in Eqns 5 and 6. Thus it would never get canceled unless the coefficient of the above element is 0.

2. Linear relations containing C_0^j do not exist as the variable s^j appears only in the elements C_i^j whose coefficients are all 0.

Thus there are no dependencies in the source group involving key or ciphertext elements. Next we enumerate the list of elements in the target group and study dependencies between them.

1. First, we claim that any linear relation containing the product of the element $(\sum_{i=1}^n (-H_i \delta_i^j),\ Q_6^j,\ R_5^j)$ has to be with some element of the form $(s^k P\ ,0\ ,0)$. We proceed to rule out all other cases below. First, note that the element cannot be multiplied with itself as no other multiplication will generate $(R_5^j)^2$. Also, it cannot be multiplied with any of the other elements of SK^j since : 1) it cannot be multiplied with $(P\delta_i^j, *, 0)$ as the first component will then contain some term of the form $PH_i \cdot (\delta_i^j)^2$ and this term cannot be canceled : all other ways to generate $(\delta_i^j)^2$ involve multiplying two elements with $(P\delta_i^j, *, 0)$ would produce a P^2 term and thus be unsuitable. 2) it cannot be multiplied with $(P\delta_k^j, *, *)$ for $k \neq i,\ n > 1$, since the first term $\sum_{i=1}^n (-H_i \delta_i^j) \cdot P\delta_k^j$ cannot be constructed otherwise.Using the same reasoning as above, it cannot also be multiplied with key elements of other secret keys SK^k for $k \neq j$. It cannot be multiplied with ciphertext elements of the form $(s^k H_i,\ Q\alpha^k x_i^k,\ r_i^k)$ since the third component of the multiplication – $r_i^k R_5^j$ cannot be constructed otherwise.

2. Linear relations containing a multiplication of the element $(P\delta_i^j, *, 0)$ cannot exist unless it is multiplied with a term of the form $(s^k H_i, *, *)$ because: 1) Multiplying it with itself or other key elements of the form $(P\delta_a^b, *, 0)$ produces terms containing $(P^2 \cdot \delta_i^j \delta_a^b, *, 0)$ which cannot be canceled. 2) Multiplying it with $(\sum_{i=1}^n (-H_i \delta_i^j),\ Q_6^j,\ R_5^j)$ is ruled out by previous analysis 3) Multiplying it with any term $(s^k H_\ell, *, *)$ for $\ell \neq i$ cannot work because the term $\delta_i^j H_\ell$ only occurs again in some element $(P\delta_i^j H_\ell s^{k'}, *, *)$ with a different multiplier $s^{k'}$. Thus, multiplying it with any term which is not $(s^k H_i, *, *)$ produces terms that cannot be canceled.

   Thus, we have shown that the only feasible multiplications which contribute to linear relations are those which multiply the first key component K_0 with the first ciphertext component C_0 and the i^{th} key component with the i^{th} ciphertext component for $i = 1, \ldots n$.

   Now, next observe that if the first key component K_0^j multiplied with the first ciphertext component C_0^k we get the term $-\sum_i H_i \delta_i^j s^k P$ in the $\mathcal{G}_p$ component which can only be obtained by multiplying K_i^j with C_i^k for $i \in [n]$ and adding them up. This constitutes the legitimate decryption procedure and would also appear in the ideal world.

Suppose we ignore K_0 and C_0 terms and multiply just the i^{th} components for various ciphertext-key pairs, we get terms of the form $H_i\delta_i^j$ which do not appear anywhere except K_0 and hence cannot get canceled. Also note that every legitimate decryption expression has a unique key-ciphertext identifier $f^j\alpha^j$ the second component, and hence this cannot be combined with any other legitimate decryption expression.

Next, we consider the linear relations that involve ciphertext elements. Note that by the above analysis we have already ruled out linear relations involving key elements so it suffices to restrict our attention to linear relations that involve only ciphertext group elements. Recall what the ciphertext group elements look like:

$$\left\{C_0^j = (s^jP,\ 0,\ 0), \left\{C_i^j = (s^jH_i,\ Q\alpha^j x_i^j,\ r_i^j)\right\}_{i=1}^{i=n}\right\}_{j=1}^{m_c}$$

Next we look at possible admissible relations in the target group.

1. Consider elements of the form C_i^j for any i, j. Such elements can only be multiplied with elements whose third component is 0 in order to cancel out the formal variable r_i^j. Hence these elements can only be multiplied with an element of the form C_0^k for some k. Consider multiplications of C_i^j with elements of the form C_0^k for any k. Such a multiplication gives rise to the term $Ps^ks^jH_i$. Note that this term does appear when we multiply C_0^j with C_i^k! However such multiplications have no term in the second component and hence always hold regardless of the message. Recall that we do not care about such relations.

2. Linear dependencies involving multiplying the element C_0^j with elements C_i^k were analyzed above. Multiplying C_0^j with C_0^k yields the term s^js^k in the first component which cannot be obtained in any other way.

Case 2: After EK is given. Apart from the key and ciphertext elements listed above, we would also like to consider admissible relations between elements of EK. The reason we would like to do this is because in the PK mode the simulator $\mathcal{S}$ is constructed in a manner as to send EK to the $\mathcal{A}$.

Recall that as 3-tuples,

$$\mathsf{EK} = \left\{(P, 0, 0), (0, Q, R_0), \{(H_i, 0, R_i)\}_{i=1}^{i=n}\right\}$$

We only need to consider admissible relations in the target group. Multiplication of any element with the terms $\{(H_i, 0, R_i)\}_{i=1}^{n}, (P, 0, 0)$ are irrelevant as they are independent of the message or key vectors. Multiplication of the term $(0, Q, R_0)$ with the term C_i^j is not useful as the unique term $R_0r_i^j$ produced in such a product can never be produced by any other multiplication.

We are only left with multiplications of the form $(0, Q, R_0)\cdot K_i^j$. These multiplications are allowed and produce elements of the form $(*, Q^2f^jv_i^j, 0)$. The adversary may compute any linear combination of any number of such multiplications, and thus these are valid admissible relations. Call these relations $\mathcal{L}$. Looking ahead we observe that these relations are set correctly by the simulator $\mathcal{S}$ as they merely mimic the Decrypt operation where the adversary used bad randomness i.e set $s = 0, \alpha = 1$ which are anyway valid computations for an adversary possessing the encryption key.

Our scheme may have other admissible relations present but these do not involve any message or key vectors, but $\mathcal{S}$ is constructed to satisfy all such dependencies since it mimics the real world. □

We are now ready to prove our main theorem. Recall that our goal was to prove that the Simulator constructed in Section 5.1 is secure. We now proceed to do this formally.

Theorem 8. *The simulator $\mathcal{S}$ constructed in Section 5.1 is such that for all adversaries $\mathcal{A}$, for all* Env *with auxiliary input z,* $\{\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*}$

Proof. The proof follows the following broad outline. According to our definition Env outputs an arbitrary PPT function of its view in both the real and ideal worlds. In the Ideal-World, $\mathcal{S}$ sets up the adversary and communicates on his behalf with Env merely transmitting messages back and forth, hence the interaction of Env and $\mathcal{A}$ in the Real-World is identical to that between Env and $\mathcal{S}$ in the Ideal-World. The interactions of Env with $\mathcal{O}$ in the Ideal-World and with Sys in the Real-World are also identical by definition. In the real world, Sys sends $\mathcal{A}$ GG elements corresponding to keys and ciphertexts of external players. In the Ideal-World these are sent to $\mathcal{A}$ by $\mathcal{S}$. We now need to argue that the GG handles that $\mathcal{S}$ provides to the $\mathcal{A}$ and the GG handles that $\mathcal{A}$ receives in the Real-World are indistinguishable. We do so by constructing a sequence of hybrids. The first hybrid is the real world in which Sys sends GG elements to $\mathcal{A}$, and in subsequent hybrids, the elements returned to the adversary are one by one changed to those sent by $\mathcal{S}$. We then argue that an environment who can tell the difference was able to find an admissible relation satisfied in one game but not the other. This is a contradiction because all admissible relations identified in Theorem 7 were programmed by $\mathcal{S}$.

Denote by q the total number of GG queries made by the adversary. Let t be maximum degree of any admissible relation evaluated by $\mathcal{A}$ over key and ciphertext elements from scheme. t is a constant.

1. Hybrid 1: The first hybrid is the Real-World.

2. Hybrid 2: Replace the GG oracle and Sys by algorithms which perform the following operations. With every group element that is randomly chosen, associate a new formal variable. With every random parameter chosen in $\mathbb{Z}_p$ associate a new formal variable. All arithmetic done by the GG oracle or Sys on these parameters are now done via polynomial arithmetic. Return to the adversary, random group handles that are associated with the polynomials that he requests for.

   Hybrid 2 associates with each different formal polynomial a distinct random handle, whereas in Hybrid 1, these polynomials were evaluated by setting the formal variables to random values in $\mathbb{Z}_p$ and the resultant evaluations were assigned random group handles. The only way to distinguish between these two hybrids is if two different polynomials evaluated to the same value but were given different handles. The probability that Hybrid 1 and Hybrid 2 are distinguishable is $\leq q^2t/p$ by Theorem 4 where t is the maximum degree of a polynomial.

3. Hybrid $2+i$ for $i \in [q]$: $\mathcal{O}$ sets up the PP, MSK and EK and shares them with Sys. $\mathcal{S}$ and Sys simultaneously compute all the messages that they need to send to the adversary. However, $\mathcal{S}$ sends all the replies to $\mathcal{A}$ until the i-th GG query made by the adversary. Starting from the $i+1$-th query of the adversary Sys replies to the oracle queries. Thus, the only place where Hybrid $2+i-1$ differs from $2+i$ is that in the former, the i-th query is answered by Sys whereas in the latter it is answered by $\mathcal{S}$.

   Recall that all of the GG operations in these hybrids are still done over formal polynomials. Consider the admissible relation evaluated by $\mathcal{A}$ in Hybrid $2+(i-1)$ in the i-th query. If it involves no message or key vectors, they are satisfied automatically in Hybrid $2+i$ by the construction of $\mathcal{S}$. If they involve any message or key vectors, consider the party that issued the i-th message. If it is a part of a key or a ciphertext update, then it is constructed in an identical manner by both $\mathcal{S}$ and Sys. If it is a group operation then the only way in which the two hybrids can be distinguished is if the relations are different as formal polynomials. We noted in Theorem 7 that the only possible relations that occur in the game are those corresponding to decryption operations. However by construction of the simulator 5.1, all such relations are tracked and set correctly by $\mathcal{S}$. Hence there do not occur any relations that differ as formal polynomials. Thus Hybrid $2+i$ is identically distributed to Hybrid $2+(i-1)$.

4. Hybrid $(2+q)$: This is the ideal world.

This completes the proof. We also observe that although the theorem calls for only computational indistinguishability between the real and ideal worlds, we obtain statistical indistinguishability. □

F Obfuscation scheme from [KSW08]

In this section we present an obfuscation scheme for the inner product functionality that arises naturally from the [KSW08]. We begin by providing a definition for obfuscation schemes from [CRV10].

F.1 Formal definition of obfuscation

Let $\mathcal{C} = C_\kappa, \kappa \in \mathbb{Z}^+$ be a family of polynomial-size circuits, where C_κ denotes all circuits of input length κ. A probabilistic polynomial time (PPT) algorithm $\mathcal{O}$ is an obfuscator for the family $\mathcal{C}$ if the following three conditions are met.

- **Approximate functionality**: There exists a negligible function ϵ such that for every κ, every circuit $C \in C_\kappa$ and every x in the input space of C, $Pr[\mathcal{O}(C)(x) = C(x)] > 1 - \epsilon(\kappa)$, where the probability is over the randomness of $\mathcal{O}$. If this probability always equals 1, then we say that $\mathcal{O}$ has exact functionality.

- **Polynomial slowdown**: There exists a polynomial q such that for every κ, every circuit $C \in C_\kappa$, and every possible sequence of coin tosses for $\mathcal{O}$, the circuit $\mathcal{O}(C)$ runs in time at most $q(|C|)$.

- **Virtual black-box**: For every PPT adversary A and polynomial δ, there exists a PPT simulator S such that for all suciently large κ, and for all $C \in C_\kappa$,

$$\left|\Pr[A(\mathcal{O}(C)) = 1] - \Pr[S^C(1^\kappa) = 1]\right| < \frac{1}{\delta(\kappa)},$$

where the first probability is taken over the coin tosses of A and $\mathcal{O}$, and the second probability is taken over the coin tosses of S.

F.2 Obtaining an inner product obfuscator from [KSW08]

Let $C_{p,n} = \{\mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \{0,1\}\}$ be a function indexed by $p \in \mathsf{primes}(1^\kappa)$ and $n \in \mathbb{Z}^+, n > 1$. We now provide an obfuscator $\mathcal{O}$ for the function family $\mathcal{C} = \{C_{p,n}\}$. Our goal is to base the obfuscator directly on the functional encryption scheme from Section 4. Using the notation therein, let $C_{p,n}$ be the function $\mathcal{F}$.

An obfuscator is constructed as follows.

- **Input**: $\vec{v} \in C_{p,n}, p, n$.

- Let $(\mathsf{PP}, \mathsf{MSK}, \mathsf{EK}) = \mathsf{Setup}(1^\kappa)$. Let $\mathsf{SK}_{\vec{v}} = \mathsf{KeyGen}(\mathsf{MSK}, \vec{v})$. Publish, PP, EK and $\mathsf{SK}_{\vec{v}}$ as a part of the program $\mathcal{O}$.

- Upon input x, the computation proceeds as follows. Let $\mathsf{CT}_x = \mathsf{Encrypt}(\mathsf{EK}, x)$. Let $y = \mathsf{Decrypt}(\mathsf{SK}_{\vec{v}}, \mathsf{CT}_{\vec{x}})$.

- **Output:** y

F.3 Proof of security

We informally mention the reason why construction from F.2 is a valid obfuscation scheme.

- **Approximate functionality**: The scheme $\mathcal{O}$ achieves exact functionality from the exact correctness of the underlying functional encryption scheme.

- **Polynomial slowdown**: The scheme achieves polynomial slowdown because of the polynomial runtime of all the algorithms in the underlying functional encryption scheme.

- **Virtual black-box**: The scheme satisfies the virtual black-box property *in the generic group model* from the proof of security of the underlying FE scheme. We defer a formal proof of this last property to the full version.