Function Private Functional Encryption and Property Preserving Encryption : New Definitions and Positive Results

Shashank Agrawal [*] Shweta Agrawal [†] Saikrishna Badrinarayanan [‡]
Abishek Kumarasubramanian [§] Manoj Prabhakaran [¶] Amit Sahai [‖]

Abstract

This work furthers the exploration of meaningful definitions for security of Functional Encryption. We propose new simulation based definitions for function privacy in addition to data privacy and study their achievability. In addition, we improve efficiency/ underlying assumptions/ security achieved by existing inner product Functional Encryption and Property Preserving Encryption schemes, in both the private and public key setting. Our results can be summarized as follows:

- We present a new simulation based definition, which we call Relax-AD-SIM, that lies between simulation based (SIM) and indistinguishability based (IND) definitions for data privacy, and implies the function privacy definition of [BRS13a]. Our definition relaxes the requirements on the simulator to bypass impossibility of SIM in the standard model. We show that the inner product FE scheme of [KSW08] enjoys Relax-AD-SIM security for function hiding and the inner product FE scheme of [LOS+10] enjoys Relax-AD-SIM security for data hiding.
- We study whether known impossibilities for achieving strong SIM based security imply actual real world attacks. For this, we present a new UC-style SIM based definition of security that captures both data and function hiding, both public key and symmetric key settings and represents the "dream" security of FE. While known impossibilities rule out its achievability in the standard model, we show, surprisingly, that it can be achieved in the generic group model for Inner Product FE ([KSW08]). This provides evidence that FE implementations may enjoy extremely strong security against a large class of real world attacks, namely generic attacks.
- We provide several improvements to known constructions of Inner Product FE. In the private key setting, the construction by Shen et al. was based on non-standard assumptions, used composite order groups, and only achieved selective security. We give the first construction of a symmetric key inner product FE which is built using prime order groups, and is *fully* secure under the standard DLIN assumption. Our scheme is more efficient in the size of key and ciphertext than [SSW09], when the latter is converted to prime-order groups. We also port the public key inner product scheme of [KSW08] to prime order groups.
- We give the first construction of a property preserving encryption (PPE) scheme [PR12] for inner-products. Our scheme is secure under the DLIN assumption and satisfies the strongest definition of security – Left-or-Right security in the standard model. Note that the only previously known construction for PPE [PR12] (which was claimed to be secure in the generic group model) was recently attacked [CD13], making our construction the first candidate for PPE.

[*]UIUC. Email: sagrawl2@illinois.edu.
[†]I.I.T, Delhi. Email: shweta.a@gmail.com.
[‡]UIUC. Email: bsaikrishna7393@gmail.com.
[§]UCLA Email: abishekk@cs.ucla.edu.
[¶]UIUC. Email: manojmp@gmail.com.
[‖]UCLA Email: sahai@cs.ucla.edu.

1 Introduction

In recent times, the area of Functional encryption (FE) [SW05, SW] has enjoyed immense importance in the field of cryptography. Functional encryption is a generalization of public key encryption which allows tremendous flexibility and control in learning information from encrypted data. In functional encryption, a user can be provided with a secret key corresponding to a function f, denoted by SK_f. Given SK_f and ciphertext $\mathsf{CT}_x = \mathsf{Encrypt}(x)$, the user may run the decryption procedure to learn $f(x)$. Security of the system guarantees that nothing beyond $f(x)$ can be learned from CT_x and SK_f.

Classic results in the area focused on constructing FE for restricted classes of functions. Identity based encryption (IBE) had already been constructed [Sha84, BF01, Coc01, BW06, GPV08, CHKP10, ABB10a, ABB10b] although it was not viewed as a special case of FE until later. Sahai and Waters provided the first construction for threshold functions [SW05] which was followed by rapid succession of ever-more powerful classes of functions: membership checking [BW07], boolean formulas [GPSW06, BSW07, LOS+10], inner product functions [KSW08, LOS+10, AFV11] and more recently, even regular languages [Wat12]. More contemporary constructions of Functional Encryption 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 and also constructed FE for Turing machines [GKP+13a]. In a new breakthrough result, Garg et al. [GGH+13a] have constructed indistinguishability-secure multi-key FE schemes for all circuits.

Despite massive progress in constructing functional encryption systems for advanced classes of functions, it has remained unclear as to what is the "right" definition of security for FE systems. Security of FE comprises two orthogonal aspects: hiding the message or data embedded in the ciphertext, and hiding the function embedded in the key. A definitional framework studying data privacy for general FE was first proposed by Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10]. These and subsequent works elucidated the subtleties involved in defining a security model that captures meaningful real world data privacy. Despite considerable research [BSW11, O'N10, BO12, BF13, AGVW13, CIJ+13], what is the strongest, achievable notion for data privacy remains an actively debated open question. Unlike its counterpart, function privacy for FE has received comparatively less attention, primarily because some information about the function is inherently leaked by FE functionality in the public key setting. To see this, note that a holder of SK_f can always encrypt messages x_i of her choice and learn $f(x_i)$, thus learning information about the function. Recently, Boneh et. al. proposed a candidate security definition for function privacy [BRS13a] in the public key setting. In the symmetric key setting, function privacy was studied by Shen et al. [SSW09].

In this work, we revisit the security definitions for both data and function privacy of FE.

Data Privacy. Known definitions of data security for FE may be divided into two broad classes: Indistinguishability (IND) based or Simulation (SIM) based. Indistinguishability based security stipulates that it is in feasible 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. 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 e.g., 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 it has been shown that SIM based data security is impossible to achieve [BSW11, BO12], even when the adversary is restricted to pre-challenge queries [AGVW13]. This state of affairs raises the following questions:

1. *Is there a relaxation* [1] *of SIM security for data privacy that is stronger than IND based security but bypasses impossibilities?*

2. *What do impossibilities for SIM security mean in practice? Does inability to simulate lead to actual real world attacks?*

Function Privacy. In the symmetric key setting, function privacy was first studied by Shen et al. [SSW09], who define strong IND based security that captures both data and function privacy. In the public key setting, the first definition for function privacy was suggested very recently by Boneh et al. [BRS13a]. Their definition stems from the observation that if an adversary has some a-priori knowledge that the function comes from a small set, then he can encrypt messages that decrypt to different values for each candidate function, potentially leading to full recovery of the function embedded inside the key. They propose an IND style "real-or-random" definition of function privacy, which we call $\mathsf{ReOrRand}^{\mathsf{Fn}}$, that stipulates that as long as the function f was chosen from a sufficiently high min-entropy distribution, the adversary should not be able to distinguish SK_f from a uniformly random secret key.

Boneh et al. [BRS13a] also suggest that function privacy for FE can be formalized in a framework inspired by program obfuscation. However, whereas obfuscation security is usually defined in simulation style, the definition in [BRS13a] is indistinguishability style. Therefore, the real-or-random definition for function privacy, while elegant and strong, does not compose well with the main accepted definition (SIM based) of data security. Thus, it is natural to ask:

Is there a natural simulation based definition for function privacy in FE that is at least as strong as the real-or-random definition of [BRS13a]? Is such a definition achievable?

1.1 Our Results

In this work, we further explore meaningful definitions for FE security, and study relations with existing definitions. In addition, we improve efficiency/ underlying assumptions/ security achieved by existing inner product FE schemes, in both the private and public key setting. Our results can be summarized as follows:

- We present a new simulation based definition, which we call Relax-AD-SIM which captures data privacy as well as function privacy for FE. For data privacy, it interpolates simulation based (SIM) and indistinguishability based (IND) definitions, while for function privacy, it is at least as strong as the real-or-random definition of [BRS13a]. Our definition relaxes the requirement on the simulator to bypass impossibility of SIM based data security in the standard model. We show that the inner product FE scheme of [KSW08] enjoys Relax-AD-SIM security for function hiding and the inner product FE scheme of [LOS+10] enjoys Relax-AD-SIM security for data hiding as long as the predicate and data vectors are of constant dimension.

- We study whether known impossibilities [BSW11, BO12, AGVW13] for achieving strong SIM based security imply actual real world attacks. For this, we examine the unified definition of data and function privacy above, *without* the relaxation on the simulator, capturing the "dream" security of FE. While known impossibilities rule out its achievability in the standard model, we show, surprisingly, that it can be achieved in the generic group model for Inner Product FE ([KSW08]). This provides evidence that FE implementations enjoy extremely strong security against a large class of real world attacks, namely generic attacks.

- We provide several improvements to known constructions of Inner Product FE. In the private key setting, the construction by Shen et al. [SSW09] was based on non-standard assumptions, used

[1] Non-adaptive SIM security has been considered where the attacker is only allowed to request keys before he sees the challenge. The focus of our work will be full adaptive SIM security

composite order groups, and only achieved selective security. We give the first construction of a symmetric key inner product FE which is built using prime order groups, and is *fully* secure under the standard DLIN assumption. Our scheme is more efficient in the size of key and ciphertext than [SSW09], when the latter is converted to prime-order groups. We also port the public key inner product scheme of [KSW08] to prime order groups.

- We give the first standard model construction of a property preserving encryption (PPE) scheme [PR12] for inner-products. Our scheme is secure under the DLIN assumption and satisfies the strongest definition of security – Left-or-Right security in the standard model. Note that the only previously known construction for PPE [PR12] (which was claimed to be secure in the generic group model) was recently attacked [CD13], making our construction the first candidate for PPE.

1.2 Related Work and Comparison

In this section, we provide an overview of related work. Since our work touches upon exploring meaningful security definitions, analyzing their achievability and improving existing constructions of FE and PPE, we summarize related work in the same three categories.

Security Models. As stated above, the two existing notions of data privacy for FE are IND and SIM, both of which 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 $\{1, \mathsf{many}\} \times \{\mathsf{NA}, \mathsf{AD}\} \times \{\mathsf{IND}, \mathsf{SIM}\}$.

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 $\mathsf{many\text{-}AD\text{-}SIM}^{\mathsf{msg}}$ security in the non programmable random oracle model. Bellare and O'Neill [BO12] put forward simulation-based definitions for Functional Encryption with non-black-box simulators. They also extended the lower bound for IBE [BSW11] to the setting of efficient, non-black-box simulators, assuming the existence of collision-resistant hash functions. Agrawal et al. [AGVW13] ruled out general functional encryption for $\mathsf{1\text{-}NA\text{-}SIM}^{\mathsf{msg}}$ security. Barbosa and Farshim [BF13] extended 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. Intuitively, restricted adaptive simulation security restricts the key queries an adversary can make after he sees the challenge ciphertext to functions that are *constant* over the support of the message distribution, so as to circumvent the "non-committing encryption style" impossibility exhibited by [BSW11].

Achievability. While there has been fantastic progress for constructing FE schemes for advanced classes of functions secure under weaker definitions [GVW13, GGH+13b, GGH+13a], the situation is much less optimistic for achieving stronger definitions of security even for schemes supporting restricted classes of functions. The only positive results currently known for adaptive simulation based data privacy for FE are by Boneh et al.[BSW11] for the IBE functionality in the Random Oracle Model. Note that the situation is better for *non-adaptive* simulation based data privacy – O'Neill [O'N10] showed that for certain classes of functions called *preimage samplable functions*, $\mathsf{many\text{-}NA\text{-}IND}^{\mathsf{msg}}$ and $\mathsf{many\text{-}NA\text{-}SIM}^{\mathsf{msg}}$ are equivalent. As mentioned above, Barbosa and Farshim [BF13] extended this equivalence to the restricted-adaptive setting, which is stronger than non-adaptive. However, even with this weakening of adaptivity, they are unable to achieve simulation based data privacy for inner product predicate FE; indeed they 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 adaptive simulation based data privacy of inner product

FE is explicitly left open both by [BSW11] and [BF13]. Recently, Caro et al [CIJ+13] provided a "compiler" that converts IND based data security to SIM based data security for the Circuit-FE functionality.

The question of function privacy for FE has received even lesser attention. In the private key setting, the notion of function privacy was considered by Shen et al. [SSW09] who provided an indistinguishability style definition for data and function privacy. In the public key setting, Boneh et al [BRS13a] very recently provided the first security definition for function privacy of FE and achieved it for IBE [BRS13a] and subspace membership FE [BRS13b]. Their techniques however do not apply to the case of inner product FE and they leave open the question of achieving function privacy for the same. As described above, the function privacy definition of [BRS13a] is IND style, and does not compose well with SIM style data privacy. Our relaxation of the SIM based definition (which we call Relax-AD-SIM) is inspired from "weak simulation" definitions in the obfuscation literature [Wee05, CRV10], which allow the simulator to have an inverse polynomial dependency on the distinguishing probability and also permit the simulator to accept non black box adversary specific advice.

Constructions. Katz et al. [KSW08] provide a construction of inner product FE in the public key setting achieving data privacy under an IND based definition. Shen et al. [SSW09] achieved a weaker *selective* IND based data and function privacy for the inner product functionality, from non-standard assumptions. Both [SSW09] and [KSW08] schemes for inner product FE use composite order groups. The conversion of composite order FE schemes to prime order has been studied by [GKSW10, Fre10, OT08, OT09, LOS+10, Lew12]. The primitive of property preserving encryption was defined by Pandey and Rouselakis [PR12] who gave security definitions and also provided constructions. Their construction however is only secure in the generic group model, and they explicitly left open the problem of providing a standard model construction.

1.3 Bypassing Impossibility.

We briefly examine how our techniques can be used to bypass the impossibility of many-AD-SIM^{msg} secure inner product FE [BSW11]. Recall the main idea behind the impossibility of many-AD-SIM^{msg} secure IBE in the standard model provided by [BSW11]. At a high level, the argument forced the simulator to commit to a challenge ciphertext before seeing the adaptive (or more precisely post-challenge) key queries of the adversary. Thus, the challenge ciphertext could not have been programmed to satisfy the requisite relations with the post-challenge key queries. By choosing the challenge CT to encrypt random bits and setting the number of challenge ciphertexts to be greater than the length of the secret key, the successful simulator is forced to achieve an information theoretic compression of random bits, which is impossible. For more details we refer the reader to [BSW11].

Our definition for relaxed simulation Relax-AD-SIM^{msg} bypasses the above problem by allowing the simulator to make more queries than the adversary. For the case of data hiding, the simulator is allowed to make not just the queries that the adversary makes in the current run of the experiment, but also all the extra queries that the adversary *could* have made with noticeable probability, i.e. queries it might make for a different outcome of its random coin flips. Due to this, the simulator can program the ciphertext not just for queries asked by the adversary so far, but also for queries that the adversary is likely to ask in the future. While this seems like a very weak definition at first, we show, surprisingly, that it implies the well accepted AD-IND^{msg} definition for data privacy. This is because any successful AD-IND^{msg} adversary can be converted to a *self censoring* Relax-AD-SIM^{msg} adversary, who for any sequence of random coins, can only make queries with nonzero probability that do not allow distinguishing between the IND challenge messages. Thus, the simulator is allowed to make extra queries to the oracle, but none that allow it to distinguish between the challenge messages. The formal proof is provided in Section 3.

In the generic group model, impossibility is bypassed because even though the simulator makes exactly the same queries in the same sequence as the adversary, the challenge CT that he returns is not a group element but rather handle for a group element. Thus, the simulator retains the power to program dependencies between the challenge CT and keys even *after* the CT has been committed to. We remark that

the generic group proof needs to handle some tricky issues specific to the generic group setting. An example is that the adversary who owns some secret keys (via their group handles) can encrypt his own messages and learn dependencies between these and his secret keys. These dependencies must be programmed by the simulator, but the simulator does not know what the adversary is doing and must carefully extract this information from the adversary.

We note that our definition does not bypass the 1-NA-SIM$^{\mathsf{msg}}$ impossibility exhibited by Agrawal et al. [AGVW13]. Indeed, we show that our definition is strictly stronger than 1-NA-SIM$^{\mathsf{msg}}$ (and strictly weaker than 1-AD-SIM$^{\mathsf{msg}}$).

Organization of Paper. In Section 2, we introduce functional encryption and provide known definitions for data and function privacy, in both the symmetric and public key setting. In Section 3, we introduce our new notion of relaxed adaptive simulation based security for both data and function hiding, and analyze its relationship with known definitions. In Section 4, we provide a UC style definition of adaptive simulation based security that combines data and function privacy, public and private key, and represents the best possible security of FE, albeit impossible to instantiate in the standard model. We believe that this definition alone captures all possible usage scenarios in which FE may be deployed, and study the class of adversaries against which this definition can be achieved. In Section 5 we provide constructions of inner product FE over prime order groups, in both the public and private key setting. In Section 6, we show that our public key construction achieves both data and function hiding according to our relaxed adaptive simulation definition, for both data and function privacy. We also show that our public key construction achieves the UC style best possible simulation based security, for both data and function privacy in the generic group model. In Section 7, we show that our private key construction achieves IND based security in the standard model from the standard DLIN assumption, thus improving the result of [SSW09]. In Section 8, we provide the first construction for property preserving encryption for the inner product functionality.

2 Preliminaries: Functional Encryption

We begin by defining some standard notation. We say that a function $f : \mathbb{Z}^+ \to \mathbb{R}^+$ is negligible if $f(\kappa) \in \kappa^{-\omega(1)}$. 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

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

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

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

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

2.1 Definition: Functional Encryption

Let $\mathcal{X} = \{\mathcal{X}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ denote ensembles where each $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ is a finite set. Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ denote an ensemble where each $\mathcal{C}_\kappa$ is a finite collection of circuits, and each circuit $C \in \mathcal{C}_\kappa$ takes as input a string $x \in \mathcal{X}_\kappa$ and outputs $C(x) \in \mathcal{Y}_\kappa$.

A functional encryption scheme $\mathcal{FE}$ for $\mathcal{C}$ 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 takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.

- $\mathsf{KeyGen}(\mathsf{MSK}, C)$ is a p.p.t. algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\kappa$ and outputs a corresponding secret key SK_C.
- $\mathsf{Encrypt}(\mathsf{PK}, x)$ is a p.p.t. algorithm that takes as input the master public key PK and an input message $x \in \mathcal{X}_\kappa$ and outputs a ciphertext CT.
- $\mathsf{Decrypt}(\mathsf{SK}_C, \mathsf{CT})$ is a deterministic algorithm that takes as input the secret key SK_C and a ciphertext CT and outputs $C(x)$.

Definition 2.1 (Correctness)**.** *A functional encryption scheme $\mathcal{FE}$ is correct if for all $C \in \mathcal{C}_\kappa$ and all $x \in \mathcal{X}_\kappa$,*

$$\Pr\left[\begin{array}{l}(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa); \\ \quad \mathsf{Decrypt}(\mathsf{KeyGen}(\mathsf{MSK}, C), \mathsf{Encrypt}(\mathsf{PK}, x)) \neq C(x)\end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is taken over the coins of Setup*,* KeyGen*, and* $\mathsf{Encrypt}$*.*

Remark 1. A functional encryption scheme $\mathcal{FE}$ may permit some *intentional leakage of information* such as the length of the message $|\vec{x}|$ or function $|C|$, 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 Data Privacy

In this section we recap known definitions of data privacy for FE.

Indistinguishability Based Data Privacy. In this section we define the standard IND based definition for data privacy in FE.

Definition 2.2 ($\mathsf{NA\text{-}IND}^{\mathsf{msg}}$- and $\mathsf{AD\text{-}IND}^{\mathsf{msg}}$-Security)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a family of circuits $\mathcal{C}$. For every p.p.t. adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following two experiments:*

$\underline{\exp^{(0)}_{\mathcal{FE},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2: $(x_0, x_1, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$
3: $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_0)$
4: $b \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{PK}, \mathsf{CT}, st)$
5: *Output* b

$\underline{\exp^{(1)}_{\mathcal{FE},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2: $(x_0, x_1, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$
3: $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, x_1)$
4: $b \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{PK}, \mathsf{CT}, st)$
5: *Output* b

Define an *admissible adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as one such that for each oracle query C of $\mathcal{A}$, $C(x_0) = C(x_1)$. *We distinguish between two cases of the above experiment:*

1. The adaptive experiment*, where the oracle $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{KeyGen}(\mathsf{MSK}, \cdot)$: the functional encryption scheme $\mathcal{FE}$ is said to be* indistinguishable-secure for one message against adaptive adversaries (1-$\mathsf{AD\text{-}IND}^{\mathsf{msg}}$-secure, for short) *if for every* admissible *p.p.t. admissible adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ defined as below is negligible in the security parameter κ:*

$$\mathsf{Adv}_{\mathcal{FE},\mathcal{A}}(\kappa) \doteq \left|\Pr[\exp^{(0)}_{\mathcal{FE},\mathcal{A}}(1^\kappa) = 1] - \Pr[\exp^{(1)}_{\mathcal{FE},A}(1^\kappa) = 1]\right|$$

where the probability is over the random coins of the algorithms of the scheme $\mathcal{FE}$ and that of $\mathcal{A}$.

2. The non-adaptive experiment, *where the oracle* $\mathcal{O}(\mathsf{MSK}, \cdot)$ *is the "empty oracle" that returns nothing: the functional encryption scheme* $\mathcal{FE}$ *is said to be* indistinguishable-secure for one message against non-adaptive adversaries (1-NA-IND$^{\mathsf{msg}}$-secure, for short) *if for every* admissible *p.p.t. adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the advantage of* $\mathcal{A}$ *defined as above is negligible in the security parameter* κ*.*

We do not distinguish between one and many message security since this definition composes [GVW12].

Simulation Based Data Privacy. In this section we define the standard notions 1-AD-SIM$^{\mathsf{msg}}$, 1-NA-SIM$^{\mathsf{msg}}$) for simulation based security for data privacy.

Definition 2.3 (Real and Ideal experiments.)**.** *Let* $\mathcal{FE}$ *be a functional encryption scheme for a circuit family* $\mathcal{C}$*. Consider a p.p.t. adversary* $A = (A_1, A_2)$ *and a stateful p.p.t. simulator* $\mathcal{S}$*.*[2] *Let* $U_x(\cdot)$ *denote a universal oracle, such that* $U_x(C) = C(x)$*. Consider the following two experiments:*

$\exp^{\mathsf{real\ msg}}_{\mathcal{FE},A}(1^\kappa)$**:**	$\exp^{\mathsf{ideal\ msg}}_{\mathcal{FE},\mathcal{S}}(1^\kappa)$**:**
1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$	1: $\mathsf{PK} \leftarrow \mathcal{S}_1(1^\kappa)$
2: $(\vec{x}, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$	2: $(\vec{x}, st) \leftarrow \mathcal{A}_1^{\mathcal{S}_1(\cdot)}(\mathsf{PK})$
3: $\vec{\mathsf{CT}} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{x})$	3: $\vec{\mathsf{CT}} \leftarrow \mathcal{S}_2^{U_x(\cdot)}(1^\kappa, 1^{\lvert x\rvert})$
4: $\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{PK}, \vec{\mathsf{CT}}, st)$	4: $\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}'(\cdot)}(\mathsf{PK}, \vec{\mathsf{CT}}, st)$
5: *Output* $(\vec{x}, \alpha)$	5: *Output* $(\vec{x}, \alpha)$

Definition 2.4 (Admissible simulator)**.** *We call a stateful simulator algorithm* $\mathcal{S}$ admissible *if it makes exactly the same circuit queries to its oracle* $U_x(\cdot)$ *as the real world adversary* $\mathcal{A}$ *makes to the key derivation oracle, and runs in time poly*(κ)*.*

Simulation security can be classified into the following two types:

1. *The adaptive case* which allows both pre-challenge and post-challenge queries, i.e. where :
   - the oracle $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{KeyGen}(\mathsf{MSK}, \cdot)$ and
   - the oracle $\mathcal{O}'(\cdot)$ is the simulator, namely $\mathcal{S}^{U_x(\cdot)}(\cdot)$
2. *The non-adaptive case*, which only allows pre-challenge queries, i.e. where the oracles $\mathcal{O}(\mathsf{MSK}, \cdot)$ and $\mathcal{O}'(\cdot)$ are both the "empty oracles" that return nothing.

Definition 2.5 (1-NA-SIM$^{\mathsf{msg}}$- and 1-AD-SIM$^{\mathsf{msg}}$- Security)**.** *The functional encryption scheme* $\mathcal{FE}$ *is then said to be* simulation-secure for one message against adaptive (resp. non-adaptive) adversaries (1-AD-SIM$^{\mathsf{msg}}$(resp. 1-NA-SIM) secure, for short) *if there is an* admissible *stateful p.p.t. simulator* $\mathcal{S}$ *such that for every p.p.t. adversary* $A = (A_1, A_2)$*, the following two distributions above are computationally indistinguishable.*

$$\left\{ \exp^{\mathsf{real\ msg}}_{\mathcal{FE},A}(1^\kappa) \right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \left\{ \exp^{\mathsf{ideal\ msg}}_{\mathcal{FE},S}(1^\kappa) \right\}_{\kappa\in\mathbb{N}}$$

Remark 2. Our definition is identical to that provided in [AGVW13] and *stronger* than that provided in [BSW11], because in [BSW11], the simulator is allowed to "rewind" the adversary. In the above definition however, the simulator is forced to commit to the ciphertext before $\mathcal{A}_2$ is invoked, thus forcing the simulator to be "straight-line". Note that for the adaptive case (where the adversary is allowed to make key queries post-challenge), composition does not hold, i.e. 1-AD-SIM$^{\mathsf{msg}}$ does not imply many-AD-SIM$^{\mathsf{msg}}$ [GVW12].

[2] One can replace a stateful simulator by a regular (stateless) simulator that outputs a state st_s upon each invocation which is carried over to its next invocation.

2.3 Function Privacy

Symmetric Key Setting. Function hiding was first considered by Shen et. al. in the symmetric key setting [SSW09]. They formalized indistinguishability based security notions for achieving function (and data) hiding as follows.

Single challenge security: Let $\mathcal{FE}_{\mathsf{PrvSC}}$ be a private key functional encryption scheme for a family of circuits $\mathcal{C} : \mathcal{X} \to \mathcal{Y}$. For a p.p.t. adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following two experiments:

$\underline{\exp^{(0)}_{\mathsf{PrvSC},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa)$
2: $(t, Y_0^t, Y_1^t, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{SK},\cdot),\mathsf{Encrypt}(\mathsf{SK},\cdot)}$
3: If t=0, $\mathsf{chal} \leftarrow \mathsf{Encrypt}(\mathsf{SK}, Y_0^0)$
4: If t=1, $\mathsf{chal} \leftarrow \mathsf{KeyGen}(\mathsf{SK}, Y_0^1)$
5: $b \leftarrow \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{SK},\cdot),\mathsf{Encrypt}(\mathsf{SK},\cdot)}(\mathsf{chal}, st)$
6: Output b

$\underline{\exp^{(1)}_{\mathsf{PrvSC},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa)$
2: $(t, Y_0^t, Y_1^t, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{SK},\cdot),\mathsf{Encrypt}(\mathsf{SK},\cdot)}$
3: If t=0, $\mathsf{chal} \leftarrow \mathsf{Encrypt}(\mathsf{SK}, Y_1^0)$
4: If t=1, $\mathsf{chal} \leftarrow \mathsf{KeyGen}(\mathsf{SK}, Y_1^1)$
5: $b \leftarrow \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{SK},\cdot),\mathsf{Encrypt}(\mathsf{SK},\cdot)}(\mathsf{chal}, st)$
6: Output b

Here $t \in \{0,1\}$ represents ciphertext challenge when set to zero, and key challenge when set to one. We call an adversary *admissible* in this setting if for every circuit query C it makes to the oracle, it holds that $C(Y_0^0) = C(Y_1^0)$, and for every ciphertext query x, it holds that $Y_0^1(x) = Y_1^1(x)$.

Definition 2.6. *The private key functional encryption scheme $\mathcal{FE}_{\mathsf{PrvSC}}$ is said to be* single challenge IND secure *if for every* admissible *p.p.t. adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ defined as below is negligible in the security parameter κ:*

$$\mathsf{Adv}_{\mathsf{PrvSC},\mathcal{A}}(\kappa) \doteq \left| \Pr[\exp^{(0)}_{\mathsf{PrvSC},\mathcal{A}}(1^\kappa) = 1] - \Pr[\exp^{(1)}_{\mathsf{PrvSC},A}(1^\kappa) = 1] \right|,$$

where the probability is over the random coins of the algorithms of $\mathcal{FE}_{\mathsf{PrvSC}}$ and that of $\mathcal{A}$.

Full security: We now define a stronger notion of security called full security. Let $\mathcal{FE}_{\mathsf{PrvFS}}$ be a private key functional encryption scheme for a family of circuits $\mathcal{C} : \mathcal{X} \to \mathcal{Y}$. For a p.p.t. adversary $\mathcal{A}$, consider the following two experiments:

$\underline{\exp^{(0)}_{\mathsf{PrvFS},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa)$
2: $b \leftarrow \mathcal{A}^{\mathsf{KeyGen}_0(\mathsf{SK},\cdot,\cdot),\mathsf{Encrypt}_0(\mathsf{SK},\cdot,\cdot)}$
3: Output b

$\underline{\exp^{(1)}_{\mathsf{PrvFS},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $\mathsf{SK} \leftarrow \mathsf{Setup}(1^\kappa)$
2: $b \leftarrow \mathcal{A}^{\mathsf{KeyGen}_1(\mathsf{SK},\cdot,\cdot),\mathsf{Encrypt}_1(\mathsf{SK},\cdot,\cdot)}$
3: Output b

---

where $\mathsf{KeyGen}_b(\mathsf{SK}, C_0, C_1) = \mathsf{KeyGen}(\mathsf{SK}, C_b)$ and $\mathsf{Encrypt}_b(\mathsf{SK}, x_0, x_1) = \mathsf{Encrypt}(\mathsf{SK}, x_b)$ for all $C_0, C_1 \in \mathcal{C}$, $x_0, x_1 \in \mathcal{X}$ and $b \in \{0,1\}$. In the above experiment, an adversary is said to be admissible if for every pair of key query (C_0, C_1) and every pair of ciphertext query (x_0, x_1), it holds that $C_0(x_0) = C_1(x_1)$.

Definition 2.7. *The private key functional encryption scheme $\mathcal{FE}_{\mathsf{PrvFS}}$ is said to be* fully secure *if for every* admissible *p.p.t. adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as below is negligible in the security parameter κ:*

$$\mathsf{Adv}_{\mathsf{PrvFS},\mathcal{A}}(\kappa) \doteq \left| \Pr[\exp^{(0)}_{\mathsf{PrvFS},\mathcal{A}}(1^\kappa) = 1] - \Pr[\exp^{(1)}_{\mathsf{PrvFS},A}(1^\kappa) = 1] \right|,$$

where the probability is over the random coins of the algorithms of $\mathcal{FE}_{\mathsf{PrvFS}}$ and that of $\mathcal{A}$.

It is easy to see that full security implies single challenge security. For the case of inner product predicates, [SSW09] have shown that a single challenge secure scheme can be used to obtain a scheme that is fully secure (but less efficient). We use their transformation in our construction of a fully secure inner-product predicate encryption scheme in Section 7.

Public Key Setting. In the public key setting, function privacy is much trickier to formulate since an adversary possessing a secret key SK_C can encrypt any message x of her choice and run the legitimate decryption procedure to learn $C(x)$.

Recently Boneh et al. put forth a definition for function privacy in the public key setting [BRS13a]. This notion considers adversaries that are given public parameters of the system and can interact with a *real or random* function privacy oracle. This oracle takes as input any adversarially-chosen distribution $\mathcal{D}$ with "sufficient" entropy (formalized below) over vectors of functions, and outputs secret keys either for functions sampled from $\mathcal{D}$ or sampled uniformly. Adversaries are allowed to adaptively interact with the real-or-random oracle, for any polynomial number of queries, as long as the distributions have a certain amount of min-entropy. At the end of the interaction, adversaries should have only a negligible probability of distinguishing between the real or random modes of the oracle.

The only requirement on $\mathcal{D}$ is that it have "sufficient" entropy, where the notion of sufficient depends on the underlying functionality of the FE scheme. For example, as observed by [BRS13a], for the case of IBE, it is sufficient if identities are picked from a distribution with $\omega(\log \kappa)$. For inner product FE schemes however, the vectors need to come from a block source so that every element has entropy given the previous elements. We will call such distributions *feasible entropy* distributions.

Definition 2.8 (Real or Random function privacy oracle)**.** *The real-or-random function-privacy oracle* $\mathsf{ReOrRand}^{\mathsf{Fn}}$ *takes as input triplets of the form* $(\mathsf{mode}, \mathsf{MSK}, \mathcal{D})$*, where* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$*,* MSK *is the master secret key, and* $\mathcal{D}$ *is a circuit representing a feasible entropy distribution over* $\mathcal{F}$*. If* $\mathsf{mode} = \mathsf{real}$ *then the oracle samples* $f \leftarrow \mathcal{D}$ *and if* $\mathsf{mode} = \mathsf{rand}$ *then it samples* $f \leftarrow \mathcal{F}$ *uniformly. It then invokes the algorithm* $\mathsf{KeyGen}(\mathsf{MSK}, \cdot)$ *on* f *for outputting a secret key* SK_f*.*

Definition 2.9 (Function Privacy Adversary)**.** *A legitimate function privacy adversary* $\mathcal{A}$ *is an algorithm that is given as input a pair* $(1^\kappa, \mathsf{PK})$ *and oracle access to* $\mathsf{ReOrRand}^{\mathsf{Fn}}(\mathsf{mode}, \mathsf{MSK}, \cdot)$ *for some* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ *and to* $\mathsf{KeyGen}(\mathsf{MSK}, \cdot)$*. It is required that all of* $\mathcal{A}$*'s queries to* $\mathsf{ReOrRand}^{\mathsf{Fn}}$ *come from a feasible entropy distribution.*

We may view $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ makes key queries to the oracle $\mathsf{ReOrRand}^{\mathsf{Fn}}$ and outputs a feasible entropy distribution $\mathcal{D}$ to the oracle and some state st, and $\mathcal{A}_2$ receives st as input, and may make additional key queries to the oracle before it finally outputs its guess bit b. Note that though $\mathcal{A}$ can make several key queries, it only outputs a single challenge distribution in the above definition.

Definition 2.10 (Function Private Encryption)**.** *A public key Functional Encryption scheme* $\mathcal{FE}$ *is* $\mathsf{ReOrRand}^{\mathsf{Fn}}$ *function private if for any PPT legitimate adversary* $\mathcal{A}$*, it holds that the advantage of* $\mathcal{A}$ *defined as below is negligible in the security parameter* κ*:*

$$\mathsf{Adv}_{\mathcal{FE},\mathcal{A}}(\kappa) \doteq \left| \Pr\left[\exp^{\mathsf{real\ fn}}_{\mathcal{FE},\mathcal{A}}(1^\kappa) = 1\right] - \Pr\left[\exp^{\mathsf{rand\ fn}}_{\mathcal{FE},\mathcal{A}}(1^\kappa) = 1\right]\right| \leq \mathrm{negl}$$

where the probability is over the random coins of the algorithms of the scheme $\mathcal{FE}$ *and that of* $\mathcal{A}$ *and for each* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ *and* $\kappa \in \mathbb{N}$ *the experiment* $\exp^{\mathsf{real\ fn}}_{\mathcal{FE},\mathcal{A}}(1^\kappa)$ *is defined as follows:*

1. $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2. $b \leftarrow \mathcal{A}^{\mathsf{ReOrRand}^{\mathsf{Fn}}(\mathsf{mode},\mathsf{MSK},\cdot),\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(1^\kappa, \mathsf{PK})$
3. *Output* b

If the above holds for a computationally bounded (resp., unbounded) function privacy adversary making a polynomial number of queries to the $\mathsf{ReOrRand}^{\mathsf{Fn}}$ *oracle, then the scheme is computationally (resp. statistically)* $\mathsf{ReOrRand}^{\mathsf{Fn}}$ *function private.*

3 Relaxed Simulation Security for Functional Encryption

In light of the strong impossibility results surrounding simulation based security definitions, we propose a relaxation of simulation security which is *achievable* in the standard model, for both data and function privacy. Although our relaxation seems counter-intuitive at first, we will show that this relaxation is meaningful and implies the IND based definition of data hiding (Section 2.2) as well as real-or-random definition for function hiding [BRS13a] (Section 2.3).

3.1 Definition of Data Hiding

The simulation based definition of data hiding is similar to the one presented in Section 2.2 except that we relax the definition of admissible simulator. We allow the size of the simulator to have an inverse polynomial dependency on the distinguishing probability. We also allow the simulator to make more queries than the adversary, as long as it restricts itself to queries made by the adversary with noticeable probability. More details follow.

The experiments $\exp^{\mathsf{real\ msg}}_{\mathcal{FE},A}(1^\kappa)$ and $\exp^{\mathsf{ideal\ msg}}_{\mathcal{FE},\mathcal{S}}(1^\kappa)$ are defined exactly as in Definition 2.3. To define our admissible simulator, we will need some notation. Note that the real world adversaries $\mathcal{A}_1, \mathcal{A}_2$ are randomized, and can make different queries in different runs of the experiment. Also note that $\mathcal{A}_2$'s queries may depend on $\mathcal{A}_1$'s queries; in particular, $\mathcal{A}_2$ may be restricted from making certain queries that leak information about x. Such restrictions are encoded in the state variable st that is passed from $\mathcal{A}_1$ to $\mathcal{A}_2$. Let $\mathcal{Q}_1$ be the distribution over the set queries made by adversary $\mathcal{A}_1$ in the experiment $\exp^{\mathsf{ideal\ msg}}_{\mathcal{FE},S}(1^\kappa)$. Let $\mathcal{Q}_2^\epsilon$ be some distribution over queries that are *likely* to be made by adversary $\mathcal{A}_2$, i.e. the queries that are made by $\mathcal{A}_2$ with probability greater than ϵ, where probability is taken over the random coins of $\mathcal{A}_2$, conditioned on st.

Definition 3.1 (Admissible simulator)**.** *For fixed $\epsilon > 0$, an admissible* Relax-AD-SIM$^{\mathsf{msg}}$ *simulator $\mathcal{S}_\epsilon = (\mathcal{S}_1, \mathcal{S}_2)$ is such that:*

- *$\mathcal{S}_1$ runs in time poly(κ).*
- *$\mathcal{S}_2$ runs in time poly$(\kappa, 1/\epsilon)$, may depend on adversary $\mathcal{A}_2(st)$ where st is output by $\mathcal{A}_1$, and has the query distribution $\mathcal{Q}_\mathcal{S}$ over sets of size poly$(\kappa, 1/\epsilon)$ and $\mathcal{Q}_\mathcal{S} \overset{c}{\approx} \mathcal{Q}_1 \cup \mathcal{Q}_2^\epsilon$ (for some $\mathcal{Q}_2^\epsilon$).*

It is important to note in particular that if the adversary $\mathcal{A}_2$ makes some query with probability less than ϵ, then the admissible simulator $\mathcal{S}_\epsilon$ is disallowed from making that query unless it was made by $\mathcal{A}_1$.

Definition 3.2. *The functional encryption scheme $\mathcal{FE}$ is then said to be* Relax-AD-SIM$^{\mathsf{msg}}$ *secure, if for every p.p.t. adversary $\mathcal{A}_1$, there exists an admissible simulator $\mathcal{S}_1$ and for all $\mathcal{A}_2$ and state st output by $\mathcal{A}_1$ there exists an admissible simulator $\mathcal{S}_2$ such that for all p.p.t distinguishers $\mathcal{D}$,*

$$|\Pr\left\{\mathcal{D}\left[\exp^{\mathsf{real\ msg}}_{\mathcal{FE},A}(1^\kappa)\right] \Rightarrow 1\right\} - \Pr\left\{\mathcal{D}\left[\exp^{\mathsf{ideal\ msg}}_{\mathcal{FE},S}(1^\kappa)\right] \Rightarrow 1\right\}| \leq \epsilon$$

3.2 Definition of Function Hiding

We provide a similar simulation based definition for function hiding. The goal of function hiding is to quantify the amount of information leaked by SK_C about the circuit C. The strongest possible goal in this aspect would be to demand that the algorithm KeyGen essentially provide an obfuscator for the circuit family $\mathcal{C}_\kappa$. Due to known impossibility results for obfuscation we propose our approximate notion of simulation identical in spirit to related works on obfuscation [Wee05, CRV10].

Definition 3.3 (Real and Ideal experiments for function hiding)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a circuit family $\mathcal{C}$. Consider a p.p.t. adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a stateful p.p.t. simulator $\mathcal{S}$. Let $W_C(\cdot)$ denote a universal oracle, such that $W_C(x) = C(x)$. Consider the following two experiments:*

$\underline{\exp^{\mathsf{real\ fn}}_{\mathcal{FE},\mathcal{A}}(1^\kappa)\textbf{:}}$

1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\kappa)$
2: $(C, st) \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{PK})$
3: $\mathsf{SK}_C \leftarrow \mathsf{KeyGen}(\mathsf{PK}, C)$
4: $\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{PK}, \mathsf{SK}_C, st)$
5: *Output* α

$\underline{\exp^{\mathsf{ideal\ fn}}_{\mathcal{FE},\mathcal{S}}(1^\kappa)\textbf{:}}$

1: $\mathsf{PK} \leftarrow \mathcal{S}(1^\kappa)$
2: $(C, st) \leftarrow \mathcal{A}_1^{\mathcal{S}(\cdot)}(\mathsf{PK})$
3: $\mathsf{SK}_C \leftarrow \mathcal{S}^{W_C(\cdot)}(1^\kappa, 1^{|x|})$
4: $\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}'(\cdot)}(\mathsf{PK}, \mathsf{SK}_C, st)$
5: *Output* α

Definition 3.4 (Admissible simulator)**.** *For any $\epsilon > 0$, the admissible* Relax-AD-SIM$^{\mathsf{Fn}}$ *simulator $\mathcal{S}_\epsilon$ for function hiding runs in time poly$(\kappa, 1/\epsilon)$, makes poly$(\kappa, 1/\epsilon)$ queries to the function oracle W_C (without any restriction on the nature of queries) and may accept non black box advice about adversary $\mathcal{A}_2$.*

Note that unlike the case of data hiding, the queries of the admissible function hiding simulator are unrestricted. This is because oracle queries in this case correspond to encryption queries not secret key queries: since any adversary given SK_C can encrypt messages x_i of her choice and run the decrypt algorithm to learn $C(x_i)$, the corresponding simulator must be allowed to request function values $C(x_i)$ for any x_i of its choice. As in the case of data hiding, Relax-AD-SIM$^{\mathsf{Fn}}$ security can be classified into the following two types:

1. *The adaptive case* which allows both pre-challenge and post-challenge queries, i.e. where :
   - the oracle $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{KeyGen}(\mathsf{MSK}, \cdot)$ and
   - the oracle $\mathcal{O}'(\cdot)$ is the simulator, namely $\mathcal{S}^{W_C(\cdot)}(\cdot)$
2. *The non-adaptive case*, which only allows pre-challenge queries, i.e. where the oracles $\mathcal{O}(\mathsf{MSK}, \cdot)$ and $\mathcal{O}'(\cdot)$ are both the "empty oracles" that return nothing.

Definition 3.5. *The functional encryption scheme $\mathcal{FE}$ is then said to be* Relax-AD-SIM$^{\mathsf{Fn}}$ *secure, if for every p.p.t. adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, every $\epsilon > 0$, there exists an* admissible *stateful p.p.t. simulator $\mathcal{S}_\epsilon$ such that for all p.p.t distinguishers $\mathcal{D}$,*

$$|\Pr\left\{\mathcal{D}\left[\exp^{\mathsf{real\ fn}}_{\mathcal{FE},\mathcal{A}}(1^\kappa)\right] \Rightarrow 1\right\} - \Pr\left\{\mathcal{D}\left[\exp^{\mathsf{ideal\ fn}}_{\mathcal{FE},\mathcal{S}}(1^\kappa)\right] \Rightarrow 1\right\}| \leq \epsilon$$

3.3 What does Relax-AD-SIM mean?

Our relaxed simulation based definition Relax-AD-SIM for data and function privacy is somewhat counter-intuitive. Allowing the simulator to make extra queries seems to imply a very weak notion of security. However, we show that it is stronger than the well accepted notions of security for data as well as function hiding. Thus, it sheds light on the meaning of adaptive IND based data privacy as well as real-or-random function privacy. We do not claim that Relax-AD-SIM is the *right* notion of simulation based security, but we do believe it is an important step in that direction. Our motivation for considering this definition was twofold – for data privacy, we wished to find a weakening of *adaptive* simulation based security that is stronger than adaptive IND based security (AD-IND$^{\mathsf{msg}}$), but nevertheless bypasses the impossibilities exhibited by [BSW11, AGVW13]. For function privacy we wished to find a simulation based definition analogous to the accepted SIM based definition of data privacy, which nonetheless implies the real-or-random function hiding definition of [BRS13a].

We establish that Relax-AD-SIM$^{\mathsf{msg}}$ implies the widely accepted AD-IND$^{\mathsf{msg}}$. We also show that Relax-1-AD-SIM$^{\mathsf{msg}}$ implies 1-NA-SIM$^{\mathsf{msg}}$.

Claim 3.6. Relax-AD-SIM$^{\mathsf{msg}} \Rightarrow$ AD-IND$^{\mathsf{msg}}$

Proof. We will show that $\neg$ AD-IND$^{\mathsf{msg}}$ $\Rightarrow$ $\neg$ Relax-AD-SIM. Given a successful AD-IND$^{\mathsf{msg}}$ adversary $(\mathcal{A}_1, \mathcal{A}_2)$ (refer Section 2.2 for definition of AD-IND$^{\mathsf{msg}}$), we will construct a real world Relax-AD-SIM adversary $(\mathcal{B}_1, \mathcal{B}_2)$ as follows: $\mathcal{B}_1$ runs $\mathcal{A}_1$, receives (x_0, x_1), flips a bit b and outputs $\mathsf{CT}(x_b)$ as the challenge CT. Now $\mathcal{B}_2$ receives $\mathsf{CT}(x_b)$ and runs $\mathcal{A}_2$ to learn b. It answers $\mathcal{A}_2$'s key queries using its keygen oracle.

$\mathcal{B}_2$ makes exactly the same key queries as $\mathcal{A}_2$ in order to simulate it. In particular,

$$\Pr\left(\mathcal{B}_2 \text{ queries } v^* \text{ s.t.} f(x_0, v^*) \neq f(x_1, v^*)\right) = 0$$

since an admissible $\mathcal{A}_2$ cannot make such a query.

Now consider an admissible simulator $\mathcal{S}_\epsilon$ who must simulate $(\mathcal{B}_1, \mathcal{B}_2)$. $\mathcal{S}_\epsilon$ has oracle access to $f(x_b, \cdot)$ and can query the oracle for any key v^* which could have been queried by $(\mathcal{B}_1, \mathcal{B}_2)$ with probability ϵ. Since $\Pr(\mathcal{B}_2 \text{ queries } \mathsf{v}^* \text{ s.t. } f(x_0, v^*) \neq f(x_1, v^*)) = 0$, $\mathcal{S}_\epsilon$ cannot make queries that distinguish x_0 from x_1. Thus, the simulator does not learn anything about b from the function oracle, hence cannot simulate $\mathsf{CT}(x_b)$ with non-negligible advantage. □

We also show that Relax-1-AD-SIM$^{\mathsf{msg}}$ implies 1-NA-SIM$^{\mathsf{msg}}$.

Claim 3.7. Relax-1-AD-SIM$^{\mathsf{msg}}$ $\Rightarrow$ 1-NA-SIM$^{\mathsf{msg}}$

Proof. Again, we show that $\neg$ 1-NA-SIM$^{\mathsf{msg}}$ $\Rightarrow$ $\neg$ Relax-1-AD-SIM$^{\mathsf{msg}}$. Thus, we convert a successful 1-NA-SIM$^{\mathsf{msg}}$ adversary into a successful Relax-1-AD-SIM$^{\mathsf{msg}}$ adversary. The Relax-1-AD-SIM$^{\mathsf{msg}}$ adversary is exactly equal to the 1-NA-SIM$^{\mathsf{msg}}$ adversary. The simulator in the Relax-1-AD-SIM$^{\mathsf{msg}}$ experiment is more powerful than in the 1-NA-SIM$^{\mathsf{msg}}$ case, and can make queries to the function oracle that $\mathcal{A}_2$ makes with probability at least ϵ. However, the 1-NA-SIM$^{\mathsf{msg}}$ adversary $\mathcal{A}_2$ makes no key queries by definition, hence probability that it queries any value is 0. Hence, the Relax-1-AD-SIM$^{\mathsf{msg}}$ simulator does not have any power over the 1-NA-SIM$^{\mathsf{msg}}$ simulator, and cannot simulate the Relax-1-AD-SIM$^{\mathsf{msg}}$ adversary. □

Next, we establish that Relax-AD-SIM$^{\mathsf{Fn}}$ implies the Real-or-Random notion of function privacy defined by [BRS13a]. At a high level, the function privacy definition of [BRS13a, BRS13b] stipulates that as long as a circuit C_{real} is chosen with "sufficient" unpredictability, an adversary should not be able to tell the difference beween $\mathsf{SK}(C_{\mathsf{real}})$ and $\mathsf{SK}(C_{\mathsf{rand}})$. Their notion of sufficiently unpredictable, depends on the underlying circuit family.

We begin by defining the notion of *feasible entropy distributions* which abstracts the "sufficient unpredictability" property required by [BRS13a, BRS13b]. A distribution $\mathcal{D}$ is called a feasible entropy distribution, if a circuit sampled from $\mathcal{D}$ cannot be differentiated from a uniformly sampled circuit, given just oracle access to the circuit. Next we show that Relax-AD-SIM$^{\mathsf{Fn}}$ $\Rightarrow$ ReOrRand$^{\mathsf{Fn}}$ for any functional encryption scheme where the functions corresponding to the secret keys are sampled from a feasible entropy distribution $\mathcal{D}$.

Definition 3.8 (Feasible Entropy Distributions)**.** *Let $W_C(\cdot)$ denote a universal functional oracle defined as $W_C(x) = C(x)$, let $\mathcal{C}_\kappa$ denote a family of circuits and let $\mathcal{D}$ any distribution on $\mathcal{C}_\kappa$. Then, we say that $\mathcal{D}$ is a feasible entropy distribution, if for all non-uniform polynomial time algorithms $\mathcal{S}$, it holds that:*

$$\left| \Pr_{C \xleftarrow{\$} \mathcal{C}_\kappa} \left(\mathcal{S}^{W_C}(1^\kappa) \Rightarrow 1\right) - \Pr_{C \xleftarrow{\mathcal{D}} \mathcal{C}_\kappa} \left(\mathcal{S}^{W_C}(1^\kappa) \Rightarrow 1\right) \right| \leq \mathrm{negl}(\kappa)$$

We define feasible entropy distribution for a single challenge function but this can be generalized to multiple challenge functions, as in [BRS13a]. We observe that the input distributions defined for the IBE functionality and the subspace product (or inner product) functionality defined by [BRS13a, BRS13b] satisfy Definition 3.8.

Claim 3.9. Relax-AD-SIM$^{\mathsf{Fn}}$ $\Rightarrow$ ReOrRand$^{\mathsf{Fn}}$ [3] *for any functional encryption scheme where the functions corresponding to the secret keys are sampled from a* feasible entropy distribution $\mathcal{D}$.

[3] see Definitions 2.10 and 3.3

Proof. Assume that there is an admissible $\mathsf{ReOrRand}^{\mathsf{Fn}}$ adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ who distinguishes between real and random with non-negligible advantage α (see Definition 3.3). We will build a real world $\mathsf{Relax\text{-}AD\text{-}SIM}^{\mathsf{Fn}}$ adversary $(\mathcal{B}_1, \mathcal{B}_2)$ as follows: $\mathcal{B}_1$ runs $\mathcal{A}_1$ and receives $\mathcal{D}$. $\mathcal{B}_1$ samples a circuit C_{real} from $\mathcal{D}$ and a circuit C_{rand} uniformly. Next, it flips a bit $b \in \{\mathsf{real}, \mathsf{rand}\}$ and outputs C_b. Now $\mathcal{B}_2$ receives $\mathsf{SK}(C_b)$, runs $\mathcal{A}_2$, learns b and outputs b as his view. We assume that $(\mathcal{B}_1, \mathcal{B}_2)$ can be simulated by admissible simulator $\mathcal{S}_\epsilon$, where ϵ is a non-negligible quantity to be set later and arrive at a contradiction.

Consider the following four hybrids.

1. In this game, $\mathcal{B}_2$ is given the output of the honest keygen algorithm on a circuit C_{real} drawn from the feasible entropy distribution, i.e. $\mathsf{SK}_{C_{\mathsf{real}}} = \mathsf{KeyGen}(C_{\mathsf{real}})$. $\mathcal{B}_2$ outputs some View_1.

2. In this game, we replace the challenge secret key to be the output of ideal world simulator $\mathcal{S}$ who, given access to the function oracle $W_{C_{\mathsf{real}}}(\cdot)$, runs in time $\mathrm{poly}(\kappa, 1/\epsilon)$ and outputs some $\mathsf{SK}_{C_{\mathsf{real}}}$. Now $\mathcal{B}_2$ outputs some View_2.

3. In this game we replace the function oracle of $\mathcal{S}$ with $W_{C_{\mathsf{rand}}}(\cdot)$, the rest of the game remains the same. Say that $\mathcal{B}_2$ outputs some View_3.

4. In this game, $\mathcal{B}_2$ is given the output of the honest keygen algorithm on a circuit C_{rand} drawn from the uniform distribution, i.e. $\mathsf{SK}_{C_{\mathsf{rand}}} = \mathsf{KeyGen}(C_{\mathsf{rand}})$. $\mathcal{B}_2$ outputs some View_4.

We merely observe that hybrids 1, 2 are at most ϵ distance away because of the scheme $\mathcal{FE}$ satisfies Definition 3.3. Hybrids 2, 3 are negligibly close to each other because of the fact that the distribution $\mathcal{D}$ over the circuit family $\mathcal{C}_\kappa$ satisfies the property defined in Definition 3.8. Hybrid 3, 4 are at most ϵ distance away by Definition 3.3.

Thus we have that for any ϵ, Hybrid 1 and Hybrid 4 are separated by at most 2ϵ (plus some negligible quantity) distance. However since $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ guesses b correctly with non-negligible advantage α, by definition we have that the distance between Hybrid 1 and Hybrid 4 is $> \alpha$. Since ϵ can be arbitrary, by setting $\epsilon = \frac{\alpha}{100}$ we then arrive at a contradiction. □

4 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.

Security of Functional Encryption in practice. 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. Since simulation security for data as well as function hiding are known to be impossible to achieve, an approach is to restrict the usage scenarios by relaxing the definition of security. Another approach is to allow arbitrary usage scenarios (see Appendix A for examples) and understand what class of real world attacks are resisted by the scheme. This is the approach we take in this section. As we shall see later, this "wishful-thinking" version of FE security is achievable against a large class of real world attacks, namely generic attacks, for inner product FE.

4.1 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. 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 $\mathcal{Z}$ which is used to model all the parties external to the adversary. The adversary $\mathcal{S}$ and the environment $\mathcal{Z}$ 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[4]. The environment $\mathcal{Z}$ interacts arbitrarily with the adversary $\mathcal{S}$. The interaction between the players is described below:

- **External ciphertexts and keys:**
  - **Ciphertexts:** $\mathcal{Z}$ 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:** $\mathcal{Z}$ 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 $\mathcal{S}$.
- **Switch to public key mode:** Upon receiving a command ($\mathsf{PK\ mode}$) from $\mathcal{Z}$, $\mathcal{O}$ 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 1) 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 $\mathcal{Z}$ 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 $\mathcal{Z}$ in the ideal world.

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

- **External ciphertexts and keys:**
  - **Ciphertexts:** $\mathcal{Z}$ 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:** $\mathcal{Z}$ 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 $\mathcal{Z}$, Sys sends EK to $\mathcal{A}$.

We define $\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa)$ to be the view of $\mathcal{Z}$ 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 $\mathcal{Z}$:

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

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

5 Constructions of Inner Product FE

In this section, we provide constructions for Inner Product FE over prime order groups, in both the public key and private key setting. To begin we define some notation that will be useful to us.

5.1 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[5] 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 (Section 5.3 or Section 5.2) 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:

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

[5] Here, note that in the vein of Section 5.1, we are referring to vectors of *group elements*

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

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

5.2 Public Key FE for Inner Products over Prime Order Groups

In this section, we present a new public key 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 show that it enjoys "wishful" security against generic attacks (see Section 4), as well as the relaxed simulation security (Section 3) for *both* data and function privacy.

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. 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 $\mathsf{length}(p) = \kappa$.

- $\mathsf{Setup}(1^\kappa)$**:** 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,

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

- $\mathsf{Encrypt}(\mathsf{EK}, \vec{x})$**:** 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 1 for the above scheme is n, the length of the message and key space. 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.

5.3 Private Key FE scheme for Inner Products over Prime order groups

In this section, we construct a private-key inner-product functional encryption scheme $\mathcal{FE}_{\mathsf{Prv}}$ for attributes and predicates of length n (see Definition 2.6). Without loss of generality, we assume that the first component of an attribute or predicate is non-zero. Once again, we describe our scheme "in the exponent" for ease of notation.

- $\mathsf{Setup}(1^\kappa)$**:** Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Pick $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^{5n})$. Set,

$$\mathsf{SK} = (\mathbb{B}, \mathbb{B}^*).$$

- $\mathsf{Encrypt}(\vec{x}, \mathsf{SK})$**:** Let $\vec{x} = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{Z}_p$. Also, let $\mathbb{B} = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_{5n})$, where each $\vec{b}_j \in \mathbb{Z}_p^{5n}$. Choose $\omega, \varphi_1, \varphi_2, \ldots, \varphi_n$ uniformly and independently at random from $\mathbb{Z}_p$. The encryption of $\vec{x}$ is given by

$$\mathsf{CT}_{\vec{x}} = \omega \sum_{i=1}^{n} x_i \vec{b}_i + \sum_{i=1}^{n} \varphi_i \vec{b}_{i+4n}.$$

- $\mathsf{KeyGen}(\vec{v}, \mathsf{SK})$**:** Let $\vec{v} = (v_1, \ldots, v_n)$, where $v_i \in \mathbb{Z}_p$. Also, let $\mathbb{B}^* = (\vec{b}_1^*, \vec{b}_2^*, \ldots, \vec{b}_{5n}^*)$, where each $\vec{b}_j^* \in \mathbb{Z}_p^{5n}$. Choose $\sigma, \eta_1, \eta_2, \ldots, \eta_n$ uniformly and independently at random from $\mathbb{Z}_p$. The key for the predicate $\vec{v}$ is given by

$$\mathsf{SK}_{\vec{v}} = \sigma \sum_{i=1}^{n} v_i \vec{b}_i^* + \sum_{i=1}^{n} \eta_i \vec{b}_{i+3n}^*.$$

- $\mathsf{Decrypt}(\mathsf{CT}_{\vec{x}}, \mathsf{SK}_{\vec{v}})$**:** Compute $b = \vec{e}(\mathsf{CT}_{\vec{x}}, \mathsf{SK}_{\vec{v}})$ and output 1 if $b = e(g,g)^0$ and 0 otherwise.

It is easy to see that the above scheme is correct with all but negligible probability. If $\langle \vec{x} \cdot \vec{v} \rangle = 0 \mod p$, then $b = \vec{e}(\mathsf{CT}_{\vec{x}}, \mathsf{SK}_{\vec{v}}) = e(g,g)^{\psi\omega\sigma\langle \vec{x}\cdot\vec{v}\rangle} = e(g,g)^0$ (since $\mathbb{B}^T\mathbb{B}^* = \psi \cdot \mathbf{I}$); otherwise b is a random group element. We prove that this scheme is single challenge secure under the DLIN assumption in Section 7.

6 Security of Public Key Inner Product FE

In this section, we analyze the security of the public key inner product FE scheme provided in Section 5.2. We show that the inner product FE achieves the definition of Relax-AD-SIM security for data and function hiding in the standard model, as well as the strong "wishful" UC security defined in Section 4 in the generic group model.

6.1 Relax-AD-SIM Security in the Standard Model.

In this section, we provide a proof that the inner product FE scheme presented in Section 5.2 satisfies the Relax-AD-SIM definition of data and function hiding (Definitions 3.2, 3.3). We begin with the case of function hiding. Our proof directly uses the construction and proof of security of the inner product obfuscation scheme presented by Canetti et al. [CRV10]. Canetti et al. [CRV10] construct an obfuscator CRV-Obf for the inner product functionality under the following assumption: given a tuple of group elements $(g^{a_1}, g^{a_2}, \ldots, g^{a_d})$ where the a_i's are chosen from some joint distribution, $g^{p(a_1,\ldots,a_d)}$ is indistinguishable from uniform for all polynomials p except those which are linear when restricted to the support of the joint distribution. For the precise formulation, we refer the reader to [CRV10], Assumption 5.

Let $\mathsf{CRV\text{-}Obf}_C$ be the obfuscation of C as output by the scheme of [CRV10]. We present our proof in two steps. First, we define a hybrid world which is identical to the real world except that the output of the KeyGen algorithm is replaced by the output of adversary $\mathcal{A}_2$ given $\mathsf{CRV\text{-}Obf}_C$ as input instead of C.

We claim that the algorithm KeyGen of Section 5.2 can be rewritten in a way that it produces the correct output when given input $\mathsf{CRV\text{-}Obf}_C$ instead of C. To see this, recall from Section 5.2, that the keygen algorithm KeyGen is,

$\mathsf{KeyGen}(\mathsf{MSK}, \vec{v})$**:** 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,

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

We would also like to observe that the construction of the obfuscator $\mathsf{CRV\text{-}Obf}$ in our notation is,

$\mathsf{CRV\text{-}Obf}(\vec{v})$**:** Let $\vec{v} = (v_1, \ldots, v_n), v_i \in \mathbb{Z}_p$. Choose $\zeta \xleftarrow{\$} Z_p$ and construct $\mathsf{CRV\text{-}Obf}_{\vec{v}} = (\zeta \cdot v_1, \cdots, \zeta \cdot v_n)$.

$\mathcal{A}_2$ upon receiving n components from $\mathsf{CRV\text{-}Obf}_C$ directly plugs these into the construction KeyGen as described above and computes SK_C that is identical to the real world.

The transition from the hybrid world to the ideal world is accomplished by invoking the proof of security of the CRV-obfuscator [CRV10]. By Assumption 5 in [CRV10], for fixed $\epsilon > 0$ and obfuscator $\mathsf{CRV\text{-}Obf}_C$, Canetti et al. [CRV10] construct a simulator $\mathcal{S}'_\epsilon$ which accepts non black box advice V about adversary $\mathcal{A}_2$, runs in time $\text{poly}(\kappa, 1/\epsilon)$ and makes $\text{poly}(\kappa, 1/\epsilon)$ oracle queries for points in the set V and outputs a "dummy" obfuscator $\mathsf{CRV\text{-}Obf}'_C$. This dummy obfuscator is indistinguishable by adversary $\mathcal{A}_2$ from the real obfuscator $\mathsf{CRV\text{-}Obf}_C$ except with probability ϵ.

Formally, algorithm $\mathcal{S}_\epsilon$, given non black box advice V about adversary $\mathcal{A}_2$ works as follows:

1. $\mathcal{S}_\epsilon$ runs the honest Setup algorithm ensuring that the prime order group G used is identical to the one instantiated by the obfuscation algorithm $\mathsf{CRV\text{-}Obf}$.
2. $\mathcal{S}_\epsilon$ uses the honest KeyGen algorithm to answer key requests of adversary A_1.
3. Run the CRV simulator $\mathcal{S}'_\epsilon(V)$ and obtain the dummy obfuscator $\mathsf{CRV\text{-}Obf}'_C$
4. Run the KeyGen algorithm using $\mathsf{CRV\text{-}Obf}'_C$ instead of C as input (as outlined above) to compute the challenge secret key SK_C.
5. Run the honest KeyGen algorithm to answer key requests of adversary A_2.

By the construction of the CRV simulator, the dummy obfuscator $\mathsf{CRV\text{-}Obf}'_C$ is indistinguishable *to adversary* $\mathcal{A}_2$ from the real obfuscator obfuscator $\mathsf{CRV\text{-}Obf}_C$ except with probability ϵ. Thus, we get that that our scheme from Section 5.2 achieves $\mathsf{Relax\text{-}1\text{-}AD\text{-}SIM}^{\mathsf{Fn}}$ function privacy.

6.2 Achieving relaxed simulation security: Data Hiding.

Achieving relax sim security for the case of data hiding (definition 3.2) is different from the case of function hiding in a subtle but crucial sense. For the case of function hiding, the simulator did not have any restrictions on the queries it could make to the function oracle; that is, given access to oracle $W_C(\cdot)$, the simulator could query it on any points x_i of its choice, and in particular on all points $x_i \in V$. As mentioned above, this is because for the case of function hiding, oracle queries correspond to encryption queries made by an adversary which are unlimited and unrestricted in the public key setting. However, in the case of data privacy, oracle queries by the simulator correspond to secret key queries by the adversary and must be restricted in nature. We will construct an admissible $\mathsf{Relax\text{-}AD\text{-}SIM}^{\mathsf{msg}}$ simulator $\mathcal{S}_\epsilon$ for the inner product

FE scheme of [LOS+10] and show that the simulator is successful as long as the scheme is 1-AD-IND$^{\mathsf{msg}}$ secure. Note that 1-AD-IND$^{\mathsf{msg}}$ security for inner product FE was shown by Lewko et al. [LOS+10].

Recall that our admissible simulator $\mathcal{S}_\epsilon$ is only permitted to make queries in the set $\mathcal{Q}_\mathcal{S}$ where $|\mathcal{Q}_\mathcal{S}| = \text{poly}(\kappa, 1/\epsilon)$ and $\mathcal{Q}_\mathcal{S} \subseteq \mathcal{Q}_1 \cup \mathcal{Q}_2^\epsilon$. We construct $\mathcal{S}_\epsilon$ as follows:

1. $\mathcal{S}_\epsilon$ runs the honest Setup algorithm and obtains $(\mathsf{PK}, \mathsf{MSK})$.
2. $\mathcal{S}_\epsilon$ uses the honest KeyGen algorithm to answer the requests of adversary $\mathcal{A}_1$. $\mathcal{A}_1$ eventually outputs some x unknown to $\mathcal{S}_\epsilon$. $\mathcal{S}_\epsilon$ gets access to function oracle $U_x(\cdot)$ where $U_x(C) \doteq C(x)$.
3. $\mathcal{S}_\epsilon$ queries the function oracle U_x for all points $C_i \in \mathcal{Q}_\mathcal{S}$ and samples a message x' such that $C_i(x') = C_i(x)\ \forall\ C_i \in \mathcal{Q}_\mathcal{S}$. Next, it computes $\mathsf{CT}_{x'} = \mathsf{Encrypt}(x')$ and outputs this as the challenge CT.
4. $\mathcal{S}_\epsilon$ runs the honest KeyGen algorithm to answer key requests of adversary $\mathcal{A}_2$.
5. When $\mathcal{A}_2$ outputs some bit α[6], $\mathcal{S}_\epsilon$ outputs the same.

We show that the above simulator succeeds except with probability ϵ. We will show that if the above simulation fails then we violate 1-AD-IND$^{\mathsf{msg}}$ of the inner product FE scheme of [LOS+10]. Assume for the sake of contradiction that $\exists(\mathcal{A}_1, \mathcal{A}_2)$ that cannot be simulated by $\mathcal{S}_\epsilon$ above. We will build an IND adversary $(\mathcal{B}_1, \mathcal{B}_2)$ from $(\mathcal{A}_1, \mathcal{A}_2)$ and $\mathcal{S}_\epsilon$ as described below. Note that a description of the set $\mathcal{Q}_\mathcal{S}$ is provided alongwith $\mathcal{S}_\epsilon$.

1. Initializing $\mathcal{B}_1$: The public key PK is provided to $\mathcal{B}_1$ by the Setup algorithm.
2. Constructing $\mathcal{B}_1$: $\mathcal{B}_1$ runs $\mathcal{A}_1$, answering $\mathcal{A}_1$'s queries using its own oracle. $\mathcal{A}_1$ eventually outputs some x. $\mathcal{B}_1$ runs $\mathcal{S}_\epsilon$, answering its oracle queries C_i by simply computing $C_i(x)$ (since it knows x). $\mathcal{S}_\epsilon$ chooses x' so that $C_i(x') = C_i(x)\ \forall\ C_i \in \mathcal{Q}_\mathcal{S}$, which is learnt by $\mathcal{B}_1$. $\mathcal{B}_1$ outputs (x, x') as the IND challenge messages.
3. Constructing $\mathcal{B}_2$: Given $\mathsf{CT}^* \in \{\mathsf{CT}_x, \mathsf{CT}_{x'}\}$, $\mathcal{B}_2$ runs $\mathcal{A}_2(\mathsf{CT}^*)$, answering $\mathcal{A}_2$'s queries using its own oracle. Note that for all queries C_i of $\mathcal{A}_2$, it holds that $C_i(x) = C_i(x')$ hence $\mathcal{B}_2$ is an admissible adversary. $\mathcal{A}_2$ eventually outputs a bit α which is output by $\mathcal{B}_2$.

By assumption, $\mathcal{A}_2$ and hence $\mathcal{B}_2$ can distinguish between CT_x and $\mathsf{CT}_{x'}$ with non-negligible advantage. Thus we built an admissible IND adversary $(\mathcal{B}_1, \mathcal{B}_2)$ contradicting the 1-AD-IND$^{\mathsf{msg}}$ security of the inner product FE scheme.

Remark 3. We remark that for the inner product FE scheme of [LOS+10], 1-AD-IND$^{\mathsf{msg}}$ security and Relax-AD-SIM$^{\mathsf{msg}}$ security imply one another by the result above and Theorem 3.6.

Achieving many-challenge Relax-AD-SIM security. Since our simulator simply runs the honest Setup algorithm to generate the public parameters and the honest KeyGen algorithm to answer key queries, a simple hybrid argument shows that our simulation composes, and we achieve many-challenge key/CT, *adaptive* relax sim security.

6.3 Wishful Security in the Generic Group Model.

In this section we show that the scheme presented in Section 5.2 is secure in the generic group model as per the definition presented in Section 4. Broadly speaking, our simulator will run the adversary and provide secret keys and ciphertexts to him, as well as simulate the generic group 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

[6]Note that $\mathcal{A}_2$ is a predicate adversary not a general adversary

group handles that it obtains. 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. The detailed proof is provided in Appendix D.

7 Security for Private Key Inner Product FE

In this section, we prove that the private key inner product scheme $\mathcal{FE}_{\mathsf{Prv}}$ constructed in Section 5.3 is single challenge secure under the DLIN assumption. Towards this, we construct two *intermediate* schemes – $\mathcal{FE}_0$ and $\mathcal{FE}_1$ – in the *public-key* setting, and show how the security of these schemes implies that the scheme $\mathcal{FE}_{\mathsf{Prv}}$ is secure as well.

Consider a public-key inner-product predicate encryption scheme $\mathcal{FE}_0$ obtained by making a small modification to $\mathcal{FE}_{\mathsf{Prv}}$. In the setup phase of $\mathcal{FE}_0$, after obtaining $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathit{Dual}(\mathbb{Z}_p^{5n})$, we set the public key and (master) secret key as follows:

$$\mathcal{FE}_0.\mathsf{PK} = \widehat{\mathbb{B}} = (\vec{b}_1, \ldots, \vec{b}_n, \vec{b}_{4n+1}, \ldots, \vec{b}_{5n}) \quad \text{and} \quad \mathcal{FE}_0.\mathsf{MSK} = \widehat{\mathbb{B}}^* = (\vec{b}^*_1, \ldots, \vec{b}^*_n, \vec{b}^*_{3n+1}, \ldots, \vec{b}^*_{4n}).$$

$\mathcal{FE}_0.\mathsf{Encrypt}$ is same as the $\mathsf{Encrypt}$ algorithm described above, except that it now receives $\mathcal{FE}_0.\mathsf{PK}$ instead of SK, which is all it needs anyway. Similarly, $\mathcal{FE}_0.\mathsf{KeyGen}$ is same as KeyGen, except that it now takes $\mathcal{FE}_0.\mathsf{MSK}$ instead of SK as input. The decryption procedure stays the same and correctness is immediate. The next lemma shows that $\mathcal{FE}_0$ is a secure predicate scheme according to Definition 2.2.

Lemma 7.1 (Security of $\mathcal{FE}_0$)**.** *The scheme $\mathcal{FE}_0$ described above is a* 1-AD-IND$^{\mathsf{msg}}$ *attribute-hiding public-key inner-product predicate encryption scheme under the DLIN assumption.*

Okamoto and Takashima [OT12] recently proposed a 1-AD-IND$^{\mathsf{msg}}$ attribute-hiding public-key inner-product predicate encryption scheme under the DLIN assumption. Our scheme $\mathcal{FE}_0$ is similar to the one they have. The main difference is that our ciphertext and keys have been extended in length so that they are symmetric to each other (as our primary goal is to construct a scheme that hides both attributes and predicates in the private key setting). In Appendix C.1, we show how we can modify the security proof of Okamoto and Takashima to show that $\mathcal{FE}_0$ inherits the security properties of their scheme. This proves Lemma 7.1.

Consider another public-key inner-product encryption scheme $\mathcal{FE}_1$ obtained by swapping the public and private key in $\mathcal{FE}_0$, and thus also swapping the encryption and key-generation algorithms. This is possible because in the case of inner-product, attributes and predicates come from the same space. Specifically, in the setup phase of $\mathcal{FE}_1$, after obtaining $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathit{Dual}(\mathbb{Z}_p^{5n})$, we set the public key and (master) private key as follows:

$$\mathcal{FE}_1.\mathsf{PK} = \widehat{\mathbb{B}}^* = (\vec{b}^*_1, \ldots, \vec{b}^*_n, \vec{b}^*_{3n+1}, \ldots, \vec{b}^*_{4n}) \quad \text{and} \quad \mathcal{FE}_1.\mathsf{MSK} = \widehat{\mathbb{B}} = (\vec{b}_1, \ldots, \vec{b}_n, \vec{b}_{4n+1}, \ldots, \vec{b}_{5n}).$$

Further, $\mathcal{FE}_1.\mathsf{Encrypt} = \mathcal{FE}_0.\mathsf{KeyGen}$ and $\mathcal{FE}_1.\mathsf{KeyGen} = \mathcal{FE}_0.\mathsf{Encrypt}$. The decryption procedure stays the same and correctness is straightforward. The security of $\mathcal{FE}_1$ follows from the security of $\mathcal{FE}_0$ due to the symmetric nature of $\mathcal{FE}_{\mathsf{Prv}}$.

Lemma 7.2 (Security of $\mathcal{FE}_1$)**.** *The scheme $\mathcal{FE}_1$ described above is a* 1-AD-IND$^{\mathsf{msg}}$ *attribute-hiding public-key inner-product predicate encryption scheme under the DLIN assumption.*

We are now ready to prove the security of $\mathcal{FE}_{\mathsf{Prv}}$ according to Definition 2.6.

Theorem 7.3. *$\mathcal{FE}_{\mathsf{Prv}}$ is a single challenge IND secure private-key inner-product predicate encryption scheme for attributes and predicates of length n under the DLIN assumption.*

We prove the above theorem in Appendix C.2. In [SSW09] (Appendix A), Shen et al. describe how to construct a fully secure predicate encryption scheme (see Definition 2.7) supporting inner-products over vectors of length n from a single challenge secure scheme supporting inner-products over vectors of length $2n$. The transformation is simple: an attribute $\vec{x}$ in the former scheme is encrypted using the encryption algorithm of the latter on $\vec{x}$ concatenated with itself (keys are also generated in a similar fashion). We adopt the same approach to obtain a fully secure scheme $\mathcal{FE}^{\mathsf{FS}}_{\mathsf{Prv}}$ from the scheme $\mathcal{FE}_{\mathsf{Prv}}$. Thus we get:

Theorem 7.4. *$\mathcal{FE}^{\mathsf{FS}}_{\mathsf{Prv}}$ is a fully secure private-key inner-product predicate encryption scheme for attributes and predicates of length n under the DLIN assumption.*

8 Property Preserving Encryption

The idea of property preserving encryption (PPE) was introduced in a very recent work by Pandey and Rouselakis [PR12]. In the following, we formally define PPE schemes for binary properties, and LoR security for them.

8.1 Definition

Definition 8.1 (PPE scheme)**.** *A property preserving encryption scheme for a binary property $P : \mathcal{M} \times \mathcal{M} \to \{0,1\}$ is a tuple of four p.p.t. algorithms defined as follows:*

- $\mathsf{Setup}(1^\kappa)$ *takes as input the security parameter κ and outputs a secret key* SK *(and some public parameters).*
- $\mathsf{Encrypt}(m, \mathsf{SK})$ *takes as input a message $m \in \mathcal{M}$ and outputs a ciphertext* CT*.*
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})$ *takes as input a ciphertext* CT *and outputs a message $m \in \mathcal{M}$.*
- $\mathsf{Test}(\mathsf{CT}_1, \mathsf{CT}_2)$ *takes as input two ciphertexts CT_1 and CT_2 and outputs a bit b.*

We require that for all possible secret keys SK *output by the* Setup *algorithm, and all messages $m, m_1, m_2 \in \mathcal{M}$, the following two conditions hold:*

- *Decryption:* $\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(m, \mathsf{SK}), \mathsf{SK}) = m] \geq 1 - \mathsf{negl}(\kappa)$,
- *Property testing:* $\Pr[\mathsf{Test}(\mathsf{Encrypt}(m_1, \mathsf{SK}), \mathsf{Encrypt}(m_2, \mathsf{SK})) = P(m_1, m_2)] \geq 1 - \mathsf{negl}(\kappa)$,

where the probability is taken over the random choices of the four algorithms.

8.2 Security

In [PR12], the authors show that there exists a hierarchy of meaningful indistinguishability based security notions for PPE, which does not collapse unlike other familiar settings. At the top of the hierarchy lies *Left-or-Right* (LoR) security, a notion that is similar to full security in symmetric key functional encryption.

LoR security. Let $\Pi_P = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Test})$ be a PPE scheme for a binary property P. Consider an adversary $\mathcal{A}$ in the security game $\exp^{(b)}_{\mathsf{LoR},\mathcal{A}}(1^\kappa)$ described below, for $b \in \{0,1\}$. The Setup algorithm is run to obtain a secret key SK and some public parameters. $\mathcal{A}$ is given the parameters, and access to an oracle $\mathcal{O}_b(\mathsf{SK}, \cdot, \cdot)$, such that $\mathcal{O}_b(\mathsf{SK}, m_0, m_1) = \mathsf{Encrypt}(m_b, \mathsf{SK})$. At the end of the experiment, $\mathcal{A}$ produces an output bit. We call the adversary admissible if for every two (not necessarily distinct) pairs of messages (m_1, m_2) and (m'_1, m'_2), $P(m_1, m'_1) = P(m_2, m'_2)$.

Definition 8.2 (LoR security)**.** *The scheme Π_P is an* LoR *secure* PPE *scheme for a property P if for all p.p.t. admissible adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as below is negligible in the security parameter κ:*

$$\mathsf{Adv}_{\mathsf{LoR},\mathcal{A}}(\kappa) \doteq \left| \Pr[\exp^{(0)}_{\mathsf{LoR},\mathcal{A}}(1^\kappa) = 1] - \Pr[\exp^{(1)}_{\mathsf{LoR},A}(1^\kappa) = 1] \right|,$$

where the probability is over the random coins of the algorithms of Π_P and that of $\mathcal{A}$.

8.3 Construction

Our goal in this section is to construct a property preserving encryption (PPE) scheme for inner-product predicates. Towards this, we describe a general procedure to obtain a PPE scheme for a binary property from a private key scheme for a related class of predicates. Suppose we would like to construct a PPE scheme Π_P for a property $P : \mathcal{M} \times \mathcal{M} \to \{0,1\}$. Let $\mathcal{FE}_\mathcal{F}$ be a private key predicate encryption scheme over the class of predicates $\mathcal{F} = \{f_a \mid a \in \mathcal{M}\}$, where $f_a : \mathcal{M} \to \{0,1\}$ is defined as $f_a(b) = P(a,b)$ for all $a, b \in \mathcal{M}$. Using $\mathcal{FE}_\mathcal{F}$, we can construct Π_P as follows. The basic idea is to combine ciphertext and key of the former scheme in order to obtain a ciphertext for the latter, which allows us to operate on two ciphertexts. (This idea has been hinted at in [PR12], but not explored.)

Suppose we would like to construct a PPE scheme Π_P for a property $P : \mathcal{M} \times \mathcal{M} \to \{0,1\}$. Let $\mathcal{FE}_\mathcal{F}$ be a private key predicate encryption scheme over the class of predicates $\mathcal{F} = \{f_a \mid a \in \mathcal{M}\}$, where $f_a : \mathcal{M} \to \{0,1\}$ is defined as $f_a(b) = P(a,b)$ for all $a, b \in \mathcal{M}$. Using $\mathcal{FE}_\mathcal{F}$, we can construct Π_P as described below. The basic idea is to combine ciphertext and key of the latter scheme in order to obtain a ciphertext for the former, which allows us to operate on two ciphertexts.

- $\Pi_P.\mathsf{Setup}(1^\kappa)$**:** Run $\mathcal{FE}_\mathcal{F}.\mathsf{Setup}(1^\kappa)$ to obtain a secret key SK.
- $\Pi_P.\mathsf{Encrypt}(m, \mathsf{SK})$**:** Output $(\mathcal{FE}_\mathcal{F}.\mathsf{Encrypt}(m, \mathsf{SK}), \mathcal{FE}_\mathcal{F}.\mathsf{KeyGen}(m, \mathsf{SK}))$. Let the output be denoted by $\mathsf{CT}^*_m = (\mathsf{CT}_m, \mathsf{SK}_m)$.
- $\Pi_P.\mathsf{Test}(\mathsf{CT}^*_{m_1}, \mathsf{CT}^*_{m_2})$**:** Output $\mathcal{FE}_\mathcal{F}.\mathsf{Decrypt}(\mathsf{CT}_{m_2}, \mathsf{SK}_{m_1})$.

where $m, m_1, m_2 \in \mathcal{M}$. It is easy to see that Π_P is correct:

$$\begin{aligned}
\Pi_P.\mathsf{Test}(\mathsf{CT}^*_{m_1}, \mathsf{CT}^*_{m_2}) &= \mathcal{FE}_\mathcal{F}.\mathsf{Decrypt}(\mathsf{CT}_{m_2}, \mathsf{SK}_{m_1}) \\
&= \mathcal{FE}_\mathcal{F}.\mathsf{Decrypt}(\mathcal{FE}_\mathcal{F}.\mathsf{Encrypt}(m_2, \mathsf{SK}), \mathcal{FE}_\mathcal{F}.\mathsf{KeyGen}(m_1, \mathsf{SK}))) \\
&= f_{m_1}(m_2) \\
&= P(m_1, m_2).
\end{aligned}$$

Next, we show that if we start with a fully secure predicate scheme, we obtain a strongly secure PPE scheme.

Theorem 8.3. *A* PPE *scheme Π_P for a property P constructed in the manner described above is* LoR *secure if $\mathcal{FE}_\mathcal{F}$ is fully secure over the class of predicates $\mathcal{F}$.*

Proof. Let us say that there exists an adversary $\mathcal{A}$ attacking the scheme Π_P who obtains a non-negligible advantage in the LoR security game. We construct an adversary $\mathcal{B}$ for the fully scheme $\mathcal{FE}_\mathcal{F}$ which runs $\mathcal{A}$ as a black-box. The challenger picks up a random bit $b \in \{0,1\}$. Whenever $\mathcal{A}$ queries with two messages m_1 and m_2, $\mathcal{B}$ asks two kinds of queries to the challenger: an (m_1, m_2) ciphertext query and an (m_1, m_2) key query. $\mathcal{B}$ receives $\mathcal{FE}_\mathcal{F}.\mathsf{Encrypt}(m_b, \mathsf{SK})$ and $\mathcal{FE}_\mathcal{F}.\mathsf{KeyGen}(m_b, \mathsf{SK})$ from the challenger, which it passes on to $\mathcal{A}$ as the encryption of m_b. When $\mathcal{A}$ produces an output bit β, $\mathcal{B}$ outputs the same bit and halts.

We know that $\mathcal{A}$ is a valid adversary if for every two (not necessarily distinct) pairs of messages (m_1, m_2) and (m'_1, m'_2), $P(m_1, m'_1) = P(m_2, m'_2)$. This implies that $\mathcal{B}$ is an admissible adversary, i.e., for all ciphertext queries (x_1, x_2) and key queries (v_1, v_2), $f_{v_1}(x_1) = f_{v_2}(x_2)$. Further, it is easy to see that $\mathcal{A}$'s view in this game is indistinguishable from its view in the LoR security game. Therefore, if $\mathcal{A}$ guesses b with non-negligible probability over $1/2$, so does $\mathcal{B}$. This leads to a contradiction regarding the security of $\mathcal{FE}_\mathcal{F}$. Hence, Π_P is LoR secure. □

Note that LoR (short for left or right) security is the strongest indistinguishability based security notion proposed by Pandey and Rouselakis [PR12].

In Section 5.3, we constructed a scheme $\mathcal{FE}^{\mathsf{FS}}_{\mathsf{Prv}}$ for inner-product predicate encryption. We proved it to be fully secure under the DLIN assumption. Now, we can let $\mathcal{FE}_{\mathcal{F}} = \mathcal{FE}^{\mathsf{FS}}_{\mathsf{Prv}}$ and apply the transformation described above to obtain a PPE scheme Π_{IP} for the inner-product property ($P(\vec{a}, \vec{b}) = 1$ iff $\langle \vec{a} \cdot \vec{b} \rangle = 0$). To end this section, we have the following corollary whose proof is immediate from Theorem 7.4 and 8.3.

Corollary 8.4. Π_{IP} *is an* LoR *secure* PPE *scheme for the inner-product property under the DLIN assumption.*

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010. 1

[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. 1

[AFCK+13] DiegoF. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. 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. 41

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011. 1

[AGVW13] Shweta Agrawal, Sergey Gurbanov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013. 1, 2, 3, 5, 6, 7, 11, 26

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001. 1, 41

[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. 1, 3, 4, 6, 13, 26

[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. 1, 2, 3, 26

[BRS13a] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, 2013. 1, 2, 4, 9, 10, 11, 12

[BRS13b] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *Asiacrypt*, 2013. 4, 12

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007. 1

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011. 1, 2, 3, 4, 6, 7, 11, 13, 26

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006. 1

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007. 1

[CD13] Sanjit Chatterjee and M. Prem Laxman Das. Property preserving symmetric encryption: Revisited. *IACR Cryptology ePrint Archive*, 2013:830, 2013. 1, 3

[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. 1

[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. 1, 4

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001. 1

[Cos12] Craig Costello. Particularly friendly members of family trees. *IACR Cryptology ePrint Archive*, 2012:72, 2012. 41

[CRV10] Ran Canetti, Guy Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, 2010. 4, 10, 17, 18

[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. 41

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010. 4, 16

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010. 41

[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/. 1, 3

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013. 1, 3

[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. 1

[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. 1

[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. 4, 16

[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. 1

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008. 1

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012. 3, 7

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013. To appear. 1, 3

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008. 1, 2, 3, 4, 16, 34

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012. 4, 15, 16

[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. 1, 2, 4, 19

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. In *MATHEMATICAL NOTES*, volume 55, 1994. 31

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/. 1, 3, 26

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008. 4, 15, 16

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *Asiacrypt*, 2009. 4, 15, 16

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, number 6223 in Lecture Notes in Computer Science, pages 191–208. Springer Berlin Heidelberg, January 2010. Full version available at http://eprint.iacr.org/2010/563. 28, 29

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, number 7237 in Lecture Notes in Computer Science, pages 591–608. Springer Berlin Heidelberg, January 2012. Full version available at http://eprint.iacr.org/2011/543. 20, 27, 28

[PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, number 7237 in Lecture Notes in Computer Science, pages 375–391. Springer Berlin Heidelberg, January 2012. 1, 3, 4, 21, 22, 23

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984. 1

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt*, pages 256–266. Springer-Verlag, 1997. 31

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009. 1, 2, 3, 4, 5, 8, 9, 21

[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. 1

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005. 1

[Wat12] Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012. 1

[Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005. 4, 10

A Usage Scenarios not captured by previous definitions

In this section, we give examples of some usage scenarios that may arise in practice where FE schemes are deployed.

- *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 ciphertexts 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.

B Correctness of Public Key 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:

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

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 Security of Private Key Inner Product Predicate Encryption

C.1 Proof sketch of Lemma 7.1

We first describe our scheme $\mathcal{FE}_0$ explicitly here. We do it in a slightly different way so that it is easier to see the similarity of our scheme to the one proposed by Okamoto and Takashima [OT12]. Once again, please note that we describe our schemes, and some *problems* afterwards, 'in the exponent'.

C.1.1 Scheme

For a square matrix $\mathbb{B} = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n)$, where each $\vec{b}_i \in \mathbb{Z}_p^n$, let $(a_1, a_2, \ldots, a_n)_{\mathbb{B}} = \sum_{i=1}^{n} a_i \vec{b}_i$, where $a_i \in \mathbb{Z}_p$. The four algorithms of $\mathcal{FE}_0$ are now described as follows.

- $\mathsf{Setup}(1^\kappa)$**:** Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Pick $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^{5n})$. Let

  $$\widehat{\mathbb{B}} = (\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n, \vec{b}_{4n+1}, \vec{b}_{4n+2}, \ldots, \vec{b}_{5n}), \quad \widehat{\mathbb{B}}^* = (\vec{b}_1^*, \vec{b}_2^*, \ldots, \vec{b}_n^*, \vec{b}_{3n+1}^*, \vec{b}_{3n+2}^*, \ldots, \vec{b}_{4n}^*).$$

  Set $\mathsf{PK} = \widehat{\mathbb{B}}$ and $\mathsf{SK} = \widehat{\mathbb{B}}^*$.

- $\mathsf{Encrypt}(\vec{x}, \mathsf{PK})$**:** Let $\vec{x} = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{Z}_p$. Choose $\omega, \varphi_1, \varphi_2, \ldots, \varphi_n$ uniformly and independently at random from $\mathbb{Z}_p$. Let $\vec{\varphi}$ denote the vector $(\varphi_1, \varphi_2, \ldots, \varphi_n)$. The ciphertext for attribute $\vec{x}$ is given by
$$\mathsf{CT}_{\vec{x}} = (\quad \omega\vec{x}, \quad 0^{2n}, \quad 0^n, \quad \vec{\varphi} \quad)_B.$$

- $\mathsf{KeyGen}(\vec{v}, \mathsf{SK})$**:** Let $\vec{v} = (v_1, \ldots, v_n)$, where $v_i \in \mathbb{Z}_p$. Choose $\sigma, \eta_1, \eta_2, \ldots, \eta_n$ uniformly and independently at random from $\mathbb{Z}_p$. Let $\vec{\eta}$ denote the vector $(\eta_1, \eta_2, \ldots, \eta_n)$. The key for predicate $\vec{v}$ is given by
$$\mathsf{SK}_{\vec{v}} = (\quad \sigma\vec{v}, \quad 0^{2n}, \quad \vec{\eta}, \quad 0^n \quad)_{B^*}.$$

- $\mathsf{Decrypt}(\mathsf{CT}_{\vec{x}}, \mathsf{SK}_{\vec{v}})$**:** Compute $b = \vec{e}(\mathsf{CT}_{\vec{x}}, \mathsf{SK}_{\vec{v}})$ and output 1 if $b = e(g,g)^0$ and 0 otherwise.

C.1.2 Proof of Security

In order to prove that their public-key inner product scheme is secure, Okamoto and Takashima construct a series of hybrids linear in the number of queries made by the adversary. They define two canonical problems: Problem 1 and Problem 2, and show that the indistinguishability of hybrids can be reduced to one of these two problems. They further show that the DLIN assumption can be reduced to both problems 1 and 2, establishing the security of their scheme. For details, see Section 4.3.3 in [OT12] (full version).

Now, to prove that our scheme $\mathcal{FE}_0$ is 1-$\mathsf{AD\text{-}IND}^{\mathsf{msg}}$ attribute-hiding, we follow Okamoto and Takashima's approach. We construct the same number of hybrids in the same way, the only difference being that our ciphertexts and keys (both normal and temporal) have some extra elements at the end. Let $\mathcal{H}'$ denote our collection of hybrids, and Problem $1'$ and Problem $2'$ our two basic problems. Then, we would like to show the following:

- The indistinguishability of hybrids in $\mathcal{H}'$ can be reduced to either Problem $1'$ or Problem $2'$, and
- The DLIN assumption reduces to both the problems.

The proof of the two parts above follows the proof of Okamoto and Takashima very closely because of the similarity the two schemes possess. We defer the proof of the first part to the full version of the paper. Here, we show how the DLIN assumption can be reduced to both Problem $1'$ and Problem $2'$ via Basic Problem 0 defined in [OT10] (Definition 18 in the full version).

Problem $1'$ Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Pick $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathit{Dual}(\mathbb{Z}_p^{5n})$. Let
$$\widehat{\mathbb{B}}^* = (b_1^*, \ldots, b_n^*, b_{3n+1}^*, \ldots, b_{5n}^*).$$
Choose $\omega, \gamma_1, \ldots, \gamma_n, z$ independently and uniformly at random from $\mathbb{Z}_p$. Let $\vec{\gamma}$ denote the vector $(\gamma_1, \ldots, \gamma_n)$. Define the following quantities:
$$\begin{aligned}
\vec{f}_{0,1} &= (\quad \omega 0^{n-1}, \quad 0^{2n}, \quad 0^n, \quad \vec{\gamma} \quad)_{\mathbb{B}}, \\
\vec{f}_{1,1} &= (\quad \omega 0^{n-1}, \quad z0^{2n-1}, \quad 0^n, \quad \vec{\gamma} \quad)_{\mathbb{B}}, \\
\vec{f}_i &= \omega\vec{b}_i, \quad \text{for} \quad i = 2, \ldots, n.
\end{aligned}$$
For a p.p.t. adversary $\mathcal{A}$, consider an experiment $\exp^{(b)}_{\mathsf{P1},\mathcal{A}}(1^\kappa)$ in which $\mathcal{A}$ is given $(\mathbb{B}, \widehat{\mathbb{B}}^*, \vec{f}_{b,1}, \{\vec{f}_i\}_{i=2,\ldots,n})$, and is supposed to guess b. The advantage of $\mathcal{A}$ in Problem $1'$ is defined as:
$$\mathsf{Adv}_{\mathsf{P1},\mathcal{A}}(\kappa) = |\mathsf{Pr}[\exp^{(0)}_{\mathsf{P1},\mathcal{A}}(1^\kappa) = 1] - \mathsf{Pr}[\exp^{(1)}_{\mathsf{P1},\mathcal{A}}(1^\kappa) = 1]|.$$

Problem $2'$ Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Pick $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} Dual(\mathbb{Z}_p^{5n})$. Let

$$\widehat{\mathbb{B}} = (b_1, \ldots, b_n, b_{2n+1}, \ldots, b_{5n}).$$

Choose $\omega, \delta, \delta_0, \tau, \sigma$ independently and uniformly at random from $\mathbb{Z}_p$. Let $\vec{e}_i = (0^{i-1}, 1, 0^{n-i})$, for $i = 1, 2, \ldots, n$. Define the following quantities for $i = 1, 2, \ldots, n$:

$$\begin{aligned}
\vec{h}^*_{0,i} &= (\quad \delta\vec{e}_i, \quad 0^n, \quad 0^n, \quad \delta_0\vec{e}_i, \quad 0^n \quad)_{\mathbb{B}^*},\\
\vec{h}^*_{1,i} &= (\quad \delta\vec{e}_i, \quad \tau\vec{e}_i, \quad 0^n, \quad \delta_0\vec{e}_i, \quad 0^n \quad)_{\mathbb{B}^*},\\
\vec{h}_i &= (\quad \omega\vec{e}_i, \quad \sigma\vec{e}_i, \quad 0^n, \quad 0^n, \quad 0^n \quad)_{\mathbb{B}}.
\end{aligned}$$

For a p.p.t. adversary $\mathcal{A}$, consider an experiment $\exp^{(b)}_{\mathsf{P2},\mathcal{A}}(1^\kappa)$ in which $\mathcal{A}$ is given $(\widehat{\mathbb{B}}, \mathbb{B}^*, \{\vec{h}^*_{b,i}, \vec{h}_i\}_{i=1,\ldots,n})$, and is supposed to guess b. The advantage of $\mathcal{A}$ in Problem $2'$ is defined as:

$$\mathsf{Adv}_{\mathsf{P2},\mathcal{A}}(\kappa) = |\mathsf{Pr}[\exp^{(0)}_{\mathsf{P2},\mathcal{A}}(1^\kappa) = 1] - \mathsf{Pr}[\exp^{(1)}_{\mathsf{P2},\mathcal{A}}(1^\kappa) = 1]|.$$

Basic Problem 0**.** The definition of this problem can be found in the full version of [OT10] (Definition 18). They also show how the DLIN assumption reduces to this problem. Though the problem has been cast in an additive group, one can view it in a multiplicative group too. In the following, we show how basic problem 0 reduces to problems $1'$ and $2'$ defined above, thus proving the security of our scheme under the DLIN assumption.

For the sake of completeness, and consistency with our notation, we provide a description of Basic Problem 0 here. Let $(p, \mathcal{G}, \mathcal{G}_T, e) = \mathsf{GroupGen}(1^\kappa)$. Let X be a 3×3 matrix whose every entry is chosen independently and uniformly at random from $\mathbb{Z}_p$, such that the inverse of X exists. Let $\chi_{i,j}$ denote the entry in ith row and jth column of X. Also, let $\nu_{i,j}$ denote the entry in ith row and jth column of $(X^T)^{-1}$. For $i \in \{1, 2, 3\}$, let $\vec{b}_i = (\kappa\chi_{i,1}, \kappa\chi_{i,2}, \kappa\chi_{i,3})$ and $\vec{b}^*_i = (\xi\nu_{i,1}, \xi\nu_{i,2}, \xi\nu_{i,3})$, where $\kappa, \xi \in \mathbb{Z}_p \setminus \{0\}$. Further, pick δ, σ, ω at random from $\mathbb{Z}_p$, and ρ, τ at random from $\mathbb{Z}_p \setminus \{0\}$, and set the following:

$$\vec{y}_0 = (\delta, 0, \sigma)_{\mathbb{B}^*} \qquad \vec{y}_1 = (\delta, \rho, \sigma)_{\mathbb{B}^*} \qquad \vec{f} = (\omega, \tau, 0)_{\mathbb{B}}.$$

Finally, let $\widehat{\mathbb{B}} = (\vec{b}_1, \vec{b}_3)$ and $\mathbb{B}^* = (\vec{b}^*_1, \vec{b}^*_2, \vec{b}^*_3)$. For a p.p.t. adversary $\mathcal{A}$, consider an experiment $\exp^{(b)}_{\mathsf{BP0},\mathcal{A}}(1^\kappa)$ in which $\mathcal{A}$ is given $(\widehat{\mathbb{B}}, \mathbb{B}^*, \vec{y}^*_b, \vec{f}, \kappa, \xi, \delta\xi)$, and is supposed to guess b. The advantage of $\mathcal{A}$ in Basic Problem 0 is defined as:

$$\mathsf{Adv}_{\mathsf{BP0},\mathcal{A}}(\kappa) = |\mathsf{Pr}[\exp^{(0)}_{\mathsf{BP0},\mathcal{A}}(1^\kappa) = 1] - \mathsf{Pr}[\exp^{(1)}_{\mathsf{BP0},\mathcal{A}}(1^\kappa) = 1]|.$$

Reducing Basic Problem 0 **to Problem** $1'$**.** We show how to produce an instance of Problem $1'$ using an instance of Basic Problem 0. Suppose we have the following instance of problem 0: $(\widehat{\mathbb{B}}, \mathbb{B}^*, \vec{y}^*_b, \vec{f}, \kappa, \xi, \delta\xi)$. Let W be a $5n \times 5n$ matrix whose every entry is chosen independently and uniformly at random from $\mathbb{Z}_p$, such that the inverse of W exists. Define the matrices $\mathbb{D} = (\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_{5n})$ and $\mathbb{D}^* = (\vec{d}^*_1, \vec{d}^*_2, \ldots, \vec{d}^*_{5n})$ as follows:

$$\vec{d}_1 = W(\vec{b}^*_1, 0^{5n-3}), \qquad \vec{d}_{n+1} = W(\vec{b}^*_2, 0^{5n-3}) \qquad \vec{d}_{4n+1} = W(\vec{b}^*_3, 0^{5n-3}),$$

$$\begin{aligned}
\vec{d}_i &= W(0^{i+1}, \xi, 0^{5n-i-2}) \text{ for } i = 2, \ldots, n,\\
\vec{d}_i &= W(0^{i}, \xi, 0^{5n-i-1}) \text{ for } i = n+2, \ldots, 4n,\\
\vec{d}_i &= W(0^{i-1}, \xi, 0^{5n-i}) \text{ for } i = 4n+2, \ldots, 5n,
\end{aligned}$$

$$\vec{d}^*_1 = (W^{-1})^T(\vec{b}_1, 0^{5n-3}), \qquad \vec{d}^*_{n+1} = (W^{-1})^T(\vec{b}_2, 0^{5n-3}) \qquad \vec{d}^*_{4n+1} = (W^{-1})^T(\vec{b}_3, 0^{5n-3}),$$

$$\vec{d}_i^* = (W^{-1})^T(0^{i+1}, \kappa, 0^{5n-i-2}) \text{ for } i = 2, \ldots, n,$$
$$\vec{d}_i^* = (W^{-1})^T(0^{i}, \kappa, 0^{5n-i-1}) \text{ for } i = n+2, \ldots, 4n,$$
$$\vec{d}_i^* = (W^{-1})^T(0^{i-1}, \kappa, 0^{5n-i}) \text{ for } i = 4n+2, \ldots, 5n.$$

One can verify that $\mathbb{D}$ and $\mathbb{D}^*$ are dual orthonormal bases. Let $\widehat{\mathbb{D}}^* = (\vec{d}_1^*, \ldots, \vec{d}_n^*, \vec{d}_{3n+1}^*, \ldots, \vec{d}_{5n}^*)$. Observe that $\mathbb{D}$ and $\widehat{\mathbb{D}}^*$ can be computed from the knowledge of $\widehat{\mathbb{B}}$ and $\mathbb{B}^*$, which are part of the given instance. Further, choose $n-1$ numbers $\gamma_2, \ldots, \gamma_n$ uniformly at random from $\mathbb{Z}_p$ and set the following:

$$\vec{f}_{b,1} = W(\vec{y}_b^*, 0^{4n-2}, \gamma_2\xi, \ldots, \gamma_n\xi),$$
$$\vec{f}_i = W(0^{i+1}, \delta\xi, 0^{5n-i-2}) \text{ for } i = 2, \ldots, n.$$

Finally, output an instance of Problem 1′: $(\mathbb{D}, \widehat{\mathbb{D}}^*, \vec{f}_{b,1}, \{\vec{f}_i\}_{i=2,\ldots,n})$. We can see that:

$$\vec{f}_{0,1} = (\quad \delta 0^{n-1}, \quad 0^{2n}, \quad 0^n, \quad \vec{\gamma}' \quad)_{\mathbb{B}},$$
$$\vec{f}_{1,1} = (\quad \delta 0^{n-1}, \quad \rho 0^{2n-1}, \quad 0^n, \quad \vec{\gamma}' \quad)_{\mathbb{B}},$$
$$\vec{f}_i = \delta \vec{b}_i, \quad \text{for} \quad i = 2, \ldots, n,$$

where $\vec{\gamma}' = (\sigma, \gamma_2, \ldots, \gamma_n)$.

Reducing Basic Problem 0 to Problem 2′. We show how to produce an instance of Problem 2′ using an instance of Basic Problem 0. Once again, assume we have the following instance of problem 0: $(\widehat{\mathbb{B}}, \mathbb{B}^*, \vec{y}_b^*, \vec{f}, \kappa, \xi, \delta\xi)$. Pick W as described above. Define the matrices $\mathbb{D} = (\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_{5n})$ and $\mathbb{D}^* = (\vec{d}_1^*, \vec{d}_2^*, \ldots, \vec{d}_{5n}^*)$ as follows:

$$\vec{d}_{(j-1)n+i} = W(0^{3(i-1)}, \vec{b}_j, 0^{3(n-i)}, 0^{2n}) \text{ for } i = 1, \ldots, n; j = 1, 2,$$
$$\vec{d}_i = W(0^{n+i-1}, \kappa, 0^{4n-i}) \text{ for } i = 2n+1, \ldots, 3n$$
$$\vec{d}_{3n+i} = W(0^{3(i-1)}, \vec{b}_3, 0^{3(n-i)}, 0^{2n}) \text{ for } i = 1, \ldots, n$$
$$\vec{d}_i = W(0^{i-1}, \kappa, 0^{5n-i}) \text{ for } i = 4n+1, \ldots, 5n$$

$$\vec{d}_{(j-1)n+i}^* = (W^{-1})^T(0^{3(i-1)}, \vec{b}_j^*, 0^{3(n-i)}, 0^{2n}) \text{ for } i = 1, \ldots, n; j = 1, 2$$
$$\vec{d}_i^* = (W^{-1})^T(0^{n+i-1}, \xi, 0^{4n-i}) \text{ for } i = 2n+1, \ldots, 3n$$
$$\vec{d}_{3n+i}^* = (W^{-1})^T(0^{3(i-1)}, \vec{b}_3^*, 0^{3(n-i)}, 0^{2n}) \text{ for } i = 1, \ldots, n$$
$$\vec{d}_i^* = (W^{-1})^T(0^{i-1}, \xi, 0^{5n-i}) \text{ for } i = 4n+1, \ldots, 5n$$

One can verify that $\mathbb{D}$ and $\mathbb{D}^*$ are dual orthonormal bases. Let $\widehat{\mathbb{D}} = (\vec{d}_1, \ldots, \vec{d}_n, \vec{d}_{2n+1}, \ldots, \vec{d}_{5n})$. Now, for $i = 1, 2, \ldots, n$ set the following:

$$\vec{h}_{b,i}^* = (W^{-1})^T(0^{3(i-1)}, \vec{y}_b^*, 0^{3(n-i)}, 0^{2n}),$$
$$\vec{h}_i = W(0^{3(i-1)}, \vec{f}, 0^{3(n-i)}, 0^{2n}).$$

Finally, output an instance of Problem 2′: $(\widehat{\mathbb{D}}, \mathbb{D}^*, \{\vec{h}_{b,i}^*, \vec{h}_i\}_{i=1,\ldots,n})$. For $i = 1, 2, \ldots, n$, we can see that:

$$\vec{h}_{0,i}^* = (\quad \delta\vec{e}_i, \quad 0^n, \quad 0^n, \quad \sigma\vec{e}_i, \quad 0^n \quad)_{\mathbb{D}^*},$$
$$\vec{h}_{1,i}^* = (\quad \delta\vec{e}_i, \quad \rho\vec{e}_i, \quad 0^n, \quad \sigma\vec{e}_i, \quad 0^n \quad)_{\mathbb{D}^*},$$
$$\vec{h}_i = (\quad \omega\vec{e}_i, \quad \tau\vec{e}_i, \quad 0^n, \quad 0^n, \quad 0^n \quad)_{\mathbb{D}}.$$

where $\vec{e}_i = (0^{i-1}, 1, 0^{n-i})$.

C.2 Proof of Theorem 7.3

For the sake of contradiction, assume $\mathcal{FE}_{\mathsf{Prv}}$ is not single challenge secure. Suppose there exists an adversary $\mathcal{A}$ who gets a non-negligible advantage in the security game described in Definition 2.6. Further, assume $\mathcal{A}$ gets such an advantage in the case of $t = 1$, i.e., for key challenge. We construct an adversary $\mathcal{B}$ for $\mathcal{FE}_1$ which gets a non-negligible advantage in the security game described in Definition 2.2.

Adversary $\mathcal{B}$ runs $\mathcal{A}$ as a black-box. At the start of the game, $\mathcal{B}$ receives $\mathcal{FE}_1.\mathsf{PK} = \widehat{\mathbb{B}}^*$ from the challenger. Whenever $\mathcal{A}$ requests an encryption on an attribute $\vec{x}$, $\mathcal{B}$ asks the challenger for a key for the predicate $\vec{x}$. $\mathcal{B}$ receives $\mathcal{FE}_1.\mathsf{KeyGen}(\vec{x}, \widehat{\mathbb{B}}) = \mathcal{FE}_0.\mathsf{Encrypt}(\vec{x}, \widehat{\mathbb{B}}) = \mathcal{FE}_{\mathsf{Prv}}.\mathsf{Encrypt}(\vec{x}, \widehat{\mathbb{B}})$ from the challenger and passes it on to $\mathcal{A}$. On the other hand, whenever $\mathcal{A}$ requests for a key for a predicate $\vec{v}$, $\mathcal{B}$ encrypts $\vec{v}$ using its public key, obtaining $\mathcal{FE}_1.\mathsf{Encrypt}(\vec{v}, \widehat{\mathbb{B}}^*) = \mathcal{FE}_0.\mathsf{KeyGen}(\vec{v}, \widehat{\mathbb{B}}^*) = \mathcal{FE}_{\mathsf{Prv}}.\mathsf{KeyGen}(\vec{v}, \widehat{\mathbb{B}}^*)$, and passes it on to $\mathcal{A}$. When $\mathcal{A}$ asks the key challenge $(1, \vec{v}_0, \vec{v}_1)$, $\mathcal{B}$ sends $\vec{v}_0$ and $\vec{v}_1$ as challenge attributes to the challenger. As we have seen before, $\mathcal{B}$ obtains $\mathcal{FE}_1.\mathsf{Encrypt}\ (\vec{v}_b, \widehat{\mathbb{B}}^*) = \mathcal{FE}_{\mathsf{Prv}}.\mathsf{KeyGen}(\vec{v}_b, \widehat{\mathbb{B}}^*)$ from the challenger, and sends it to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ does. (If $\mathcal{A}$ asks for a ciphertext challenge, $\mathcal{B}$ simply aborts.)

It is easy to see that the view of $\mathcal{A}$ is same as in the security game for Definition 2.6. Therefore, if $\mathcal{A}$ succeeds with non-negligible probability in the case of $t = 1$, so does $\mathcal{B}$, which contradicts the fact that $\mathcal{FE}_1$ is secure. We can similarly show that if an adversary succeeds with non-negligible probability when asking for a ciphertext challenge, the scheme $\mathcal{FE}_0$ is not secure – again a contradiction.

D Proof of dream version data and function hiding in the generic group model

Remark 4. A subtle point that we choose not to model in the definition above is that a real world adversary given SK_f and CT_x may choose never to decrypt the two together to learn $f(x)$. In such a case the simulator need never program this dependency, and indeed does not even need to learn the value $f(x)$ from the oracle. This can be handled by providing the simulator handles for each ciphertext and key, and having it query the function oracle using these handles only if the real world adversary chooses to decrypt that ciphertext, key pair. For ease of notation we do not model this, and assume that an adversary will decrypt all ciphertext, key pairs that it receives.

D.1 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 D.1 (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 D.2 (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,*

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

D.2 Proof of Security

Simulator 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 labeled 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 $P, R, Q, R_0, \{H_i, R_i\}_{i=1}^{i=n}$[7]. He sets up the encryption key and master secret key by generating new group handles to represent the formal polynomials: $\mathsf{EK} = \left\{(P,0,0), (0,0,R), (0,Q,R_0), \{(H_i,0,R_i)\}_{i=1}^{n}\right\}$ and $\mathsf{MSK} = \left\{(0,Q,0), \{(H_i,0,0)\}_{i=1}^{i=n}, \vec{b}_1, \vec{b}_2, \vec{b}_3, \vec{b}^*_1, \vec{b}^*_2, \vec{b}^*_3\right\}$. 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 $\mathsf{PK} = (p, \mathcal{G}, \mathcal{G}_T, e)$. 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:

[7] Recall that n is part of intentionally leaked information (Remark 1) in our scheme

$$C = \left\{C_0 = (sP,\ 0,\ 0), \{C_i = (sH_i,\ Q\alpha x_i,\ r_i)\}_{i=1}^{n}\right\} \tag{2}$$

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.
- $\mathcal{S}$ 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.3.

  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.

Next, we show that the real and ideal worlds are indistinguishable to $\mathcal{Z}$. Formally, we prove the following theorem:

Theorem D.3. *The simulator $\mathcal{S}$ constructed in Section D.2 is such that for all adversaries $\mathcal{A}$, for all $\mathcal{Z}$ 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 $\mathcal{Z}$ 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.

D.3 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 5.2 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 D.4. *(***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,

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

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 $\vec{a}^T = (x \quad y \quad z)$ 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 D.5. *(**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:

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

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:

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

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 $Ps^k s^j H_i$. 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 $s^j s^k$ 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 $\mathsf{PK\ mode}$ 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 $R_0 r_i^j$ 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 $(*, Q^2 f^j v_i^j, 0)$. 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. □

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 $\mathcal{A}$.

Simplifying expressions occurring in generic group computations We describe here the Simplify step.

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 monomials of the form θf_i^j for any i and any $\theta \in \mathbb{Z}_p$. If at least one m_i is not zero, then $\mathcal{S}$ 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 $\Omega + D$.

We are now ready to prove our main theorem. Recall that our goal was to prove that the Simulator constructed in Section D.2 is secure. We now proceed to do this formally.

Theorem D.6. *The simulator $\mathcal{S}$ constructed in Section D.2 is such that for all adversaries $\mathcal{A}$, for all $\mathcal{Z}$ 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 $\mathcal{Z}$ 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 $\mathcal{Z}$ merely transmitting messages back and forth, hence the interaction of $\mathcal{Z}$ and $\mathcal{A}$ in the Real-World is identical to that between $\mathcal{Z}$ and $\mathcal{S}$ in the Ideal-World. The interactions of $\mathcal{Z}$ 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 D.5 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 $\leq q^2 t/p$ by Theorem D.2 where t is the maximum degree of a polynomial.

3. Hybrid $2+i$ for $i \in [q]$: $\mathcal{O}$ sets up the PK, 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 D.5 that the only possible relations that occur in the game are those corresponding to decryption operations. However by construction of the simulator D.2, 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. □

D.4 Concrete parameters and analysis of our scheme

In the full proof from Appendix D.3 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 $\mathcal{G}, \mathcal{G}_T$, 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.

D.5 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 $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ 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 $\mathcal{G}_1$ and the key elements from $\mathcal{G}_2$. 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.