# On the Practical Security of Inner Product Functional Encryption

Shashank Agrawal[1⋆], Shweta Agrawal[2], Saikrishna Badrinarayanan[3⋆⋆], Abishek Kumarasubramanian[4], Manoj Prabhakaran[1⋆], and Amit Sahai[3⋆⋆⋆]

[1] University of Illinois Urbana-Champaign
{sagrawl2,mmp}@illinois.edu
[2] Indian Institute of Technology Delhi
shweta@cse.iitd.ac.in
[3] University of California Los Angeles
{saikrishna,sahai}@cs.ucla.edu
[4] Google
abishekk@cs.ucla.edu

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

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

⋆ Research supported in part by NSF grant 1228856.

⋆⋆ Part of the work was done while the author was at University of Illinois Urbana-Champaign, supported by S. N. Bose scholarship.

⋆⋆⋆ Research supported in part from a DARPA/ONR PROCEED award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11- 1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

predicates constructed over elliptic curve groups. Our main contributions are as follows:

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

**Keywords:** functional encryption, practical security, pairing based cryptography, inner-product encryption, generic attacks, simulation based security.

## 1 Introduction

Functional Encryption [45,44] (FE) is an exciting new paradigm that generalizes public key encryption. In functional encryption, each decryption key corresponds to a specific function. When the holder of a decryption key for the function $f$ gets an encryption of a message $m$, the only thing his key allows him to learn is $f(m)$, but nothing more.

Classic results in the area focused on constructing FE for restricted classes of functions – point functions or identity based encryption (IBE) [46,12,21,17] [29,20,3,4] threshold functions [45], membership checking [16], boolean formulas [35,11,39], inner product functions [36,39,5] and more recently, even regular languages [49]. Recent constructions of FE support general functions: Gorbunov et al. [34] and Garg et al. [27] 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. [32] constructed succinct simulation-secure single-key FE scheme for all circuits. In a breakthrough result, Garg et al. [26] constructed indistinguishability-secure multi-key FE schemes for

all circuits. Goldwasser et al. and Ananth et al. [31,7] constructed FE for Turing machines. Recently, Functional Encryption has even been generalized to multi-input functional encryption [30].

Alongside ever-more-sophisticated constructions, there has been significant work in defining the right security model for FE. Boneh, Sahai and Waters [15] and O'Neill [43] proposed definitional frameworks to study Functional Encryption in its general form. These works discussed the subtleties involved in defining a security model for FE that captures meaningful real world security. Since then there has been considerable research focus on understanding what security means for FE and whether it can be achieved [15,43,10,9,6,19]. The strongest, most intuitive notions of security turned out to be impossible to realize theoretically, while weaker notions restricted the usage scenarios in which FE schemes could be deployed (more on this below).

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

- *Can we hide the function?* Consider the application of keyword searching on encrypted data, where the keywords being searched for are sensitive and must remain hidden. This scenario is well motivated in practice; for example the FBI might recruit untrusted server farms to perform searches on confidential encrypted data, but desire not to reveal the words being searched. Can FE schemes achieve this?
- *Can we limit what the adversary learns to* only *the function's output?* Intuitively, a functional encryption scheme should only reveal to a decryptor the function output, and nothing more. For example, if the function has some computational hiding properties, can we guarantee that the FE scheme does not leak any additional information beyond the function output?
- *Can an adversary break FE schemes where it can ask for keys after receiving ciphertexts?* In real world applications, it is very likely that an adversary can receive authorized decryption keys even after it obtains the ciphertext that it is trying to break. For example, in searchable encryption, the decryption key corresponding to a search would only be given out after the encrypted database is publicly available. Similarly in Identity Based Encryption, a user may receive an email encrypted with his identity before he obtains the corresponding secret key. Can one guarantee that an attacker who obtains an arbitrary interleaving of ciphertexts and keys, can learn nothing beyond the legitimate function values?

None of the existing security definitions for FE [15,43,10,9,6] provide comprehensive guarantees against all the above usage scenarios. Below, we discuss why this is the case, and examine alternate approaches to providing meaningful security guarantees against a wide range of practical attacks, in all the above scenarios.

*Recap of security definitions.* Before we discuss our approach, it will be useful to recap existing definitions of security and discuss their restrictions. Known definitions of security for FE may be divided into two broad classes: Indistinguishability (IND) based or Simulation (SIM) based. Indistinguishability based security stipulates that it is infeasible to distinguish encryptions of any two messages, without getting a secret key that decrypts the ciphertexts to distinct values; simulation-based security stipulates that there exists an efficient simulator that can simulate the view of the adversary, given only the function evaluated on messages and keys. Both of these notions can be further classified as follows: [43] 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 [33] described the divide between *one* versus *many*, which depends on whether the adversary receives a single or multiple challenge ciphertexts. Thus, existing definitions of security belong to the class $\{1, \mathsf{many}\} \times \{\mathsf{NA}, \mathsf{AD}\} \times \{\mathsf{IND}, \mathsf{SIM}\}$.

*Standard model woes.* Unfortunately, none of the above definitions capture security in all the usage scenarios discussed above. For example, Boneh et al. and O'Neill [15,43] showed that IND based definitions do not capture scenarios where it is required that the user learn *only* the output of the FE function, for eg., when the function hides something computationally. To get around this, [15,43] proposed SIM based definitions that study FE in the "ideal world-real world" paradigm. However, the world of SIM security for FE has been plagued with impossibilities of efficient simulation. Moreover, even the strongest known SIM based definitions (many-AD-SIM) do not capture function hiding. Even disregarding function hiding, [15] showed that many-AD-SIM is impossible even for very simple functionalities. A weakening of AD-SIM, namely NA-SIM [43] does not capture scenarios where users may obtain keys *after* obtaining new third-party-generated ciphertexts. Despite this severe restriction on usage, NA-SIM was also shown to be impossible [6], seemingly ruling out security for even those usage scenarios that *are* captured.

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

There seem to be two complementary directions forward. The first is to seek notions of security "in-between" IND and SIM that are achievable, thus providing guarantees for a restricted (but larger than IND) class of usage scenarios against all efficient attackers. Indeed, there is already research effort pursuing this agenda [6,9,2]. However, it is extremely hard (if not impossible) to control

the nature of usage scenarios that arise in practice. A second direction is to examine whether it is possible to address as many usage scenarios as we can, but restrict ourselves to analyzing security only against classes of attacks that are known to be practically feasible. This is the approach we take in this work.

In this work we study the practical security of Functional Encryption for Inner Product predicates, which is the state of the art for general FE [36,39,5]. However, we believe that the ideas developed in this work will be applicable to all FE schemes that are built from pairings on elliptic curves, which captures the majority of known FE constructions [12,45,35,17,11,36,39,49].

*Real world attacks on elliptic curve based FE.* The impossibilities exhibited by [15,6] work by arguing that there exist scenarios which preclude existence of a simulator by information theoretic arguments. However, non-existence of a simulator does not imply real world attacks in the sense of distinguishing between ciphertexts or recovering any useful information about the message or the key. Arguably, attacks that cause actual loss of secrecy are the attacks that we care about in practice, and this is the class of attacks we consider in this work.

For pairing friendly elliptic curves that are used for FE constructions, there has been extensive research effort studying practically feasible attacks. Attacks can be of two kinds: those that respect the algebraic structure of the underlying groups, which are called *generic* attacks, and those that do not, or *non-generic* attacks. Generic attacks are described as algorithms that act oblivious of particular group representations. Due to its importance and wide applicability, much research effort has been focused on studying the complexity of generic and non-generic attacks on pairing-friendly elliptic curves. By now, there is a long line of work [24,23,8,22] focused on constructing pairing friendly elliptic curves where the complexity of all known non-generic attacks is extremely high. If such elliptic curves are used to build cryptographic schemes, there is strong heuristic evidence that the only successful practically feasible attacks will be generic in nature. We stress that we will work with elliptic curve groups of prime order, and so factoring-based attacks will not be relevant.

A well known mathematical model to study generic attacks is the *Generic Group Model* (GGM) [40,48]. In the GGM, all algorithms obtain access to elements of the group via random "handles" (of sufficient length) and remain unaware of their actual representations. The GGM has a strong track record of usefulness; indeed, even notable critics of provable security, Koblitz and Menezes, despite their criticisms, admit that the generic group model has been unreasonably successful at resisting attack [37].

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

– *Capturing arbitrary usage scenarios:* We begin by providing a strong, simple and intuitive framework for security which captures all usage scenarios discussed above. Our framework captures function hiding in addition to data hiding; thus it guarantees that $\mathsf{CT}_{\boldsymbol{x}}$ and $\mathsf{SK}_f$ reveal no information about *either* $\boldsymbol{x}$ or $f$ beyond what is revealed by $f(\boldsymbol{x})$. Generalized this way, our framework can be seen to subsume program obfuscation. We also introduce the idea of having a separate encryption key in the context of Functional Encryption. This setting lies between public and symmetric key functional encryption, in that while the encryption key is not publicly known to all users, it is also not the same as the master secret key used for generating secret keys for users in the system. This allows for a division between the people that create encryptions and the people that issue secret keys. We believe this setting is well motivated in the real world, since it is often the case that there is a hierarchy that separates the people that create encrypted data and people that access it. A real-world example would be an FBI encrypted database where police officers can be granted access to parts of the database, but only FBI personnel can add to the database.
– *Resisting generic attacks:* We show that our inner product FE scheme is secure under our strong framework in the Generic Group Model, resolving the problem left open by [15] and [9]. We obtain *unconditional statistical security* for our scheme under our framework in the GGM. Our positive results also translate to the setting of obfuscation, achieving obfuscation for hyperplane membership secure against generic attacks.
– *Concrete security analysis:* Our security analysis is concrete, and as a result we can show exactly what parameters are needed to (provably) achieve security against attackers with different computational resources. For example, we show that with a pairing-friendly elliptic curve group whose order is a 222-bit prime, an attacker who is restricted to $2^{80}$ generic computations, breaks our scheme with at most $2^{-60}$ probability of success. Additional security calculations are provided in Table 1.

| Adversary Runtime | Success Probability | Required Prime Group Order (bit-length) |
|---|---|---|
| $2^{80}$ | $2^{-60}$ | 222 bits |
| $2^{80}$ | $2^{-80}$ | 242 bits |
| $2^{100}$ | $2^{-80}$ | 282 bits |
| $2^{128}$ | $2^{-80}$ | 338 bits |
| $2^{128}$ | $2^{-128}$ | 386 bits |

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

*Our perspective.* By showing that our strong security framework is realizable against all generic attacks, we are providing strong evidence of real-world security even when the generic model is instantiated in a heuristic manner – in our case with a suitably chosen pairing-friendly elliptic curve group. Much care and study

is required for how, what, and when security is preserved in such instantiations – indeed this is a very active and important area of research in our community for the Random Oracle Model. We believe that guarantees obtained by such analysis are extremely useful in practice. For example, consider the example of an IBE used in practice, say in a large organization [1]. Suppose the public parameters are published, and some user creates and publishes $2n$ encryptions for users who have yet to obtain their secret keys. Now, if $n$ out of $2n$ users are chosen in some arbitrary, ciphertext-dependent way, and these users obtain their keys, are the remaining $n$ encryptions secure? Simulation based definitions are the only definitions we know that capture security of the IBE in such scenarios, but it was shown by [15] that there cannot exist a simulator for many-AD-SIM security of IBE. On the positive side, [15] also showed that IBE does satisfy many-AD-SIM in the Random Oracle Model. We believe that this is evidence that IBEs indeed provide *practical security* in scenarios such as the above, *even despite* the impossibility of simulation in this scenario.

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

*Function privacy and obfuscation.* The question of function privacy (or key hiding) was considered by Shen et al. [47], in the symmetric key setting and more recently by Boneh et al. [13,14] in the public key setting under IND based definitions. [47] provide a construction of FE for inner product predicates in the standard model, under the IND based notion of security, using composite order groups and assuming hardness of factoring (even when viewed in the GGM). Our result, on the other hand, is unconditionally statistically secure in the generic group model, under a strong simulation based definition of security, using prime order groups. Our construction for inner product FE is inspired by the scheme of [36] and the works of [28,25,42,39,38]. It implies a program obfuscator for hyperplane membership in the generic group model – for details see Appendix D, a candidate for which was also given by [18] under a strong variant of the DDH assumption.

*Our Techniques.* Prior to our work, the only techniques to achieve positive results for many-AD-SIM security of FE were in the programmable ROM, for the anonymous IBE and public-index functionalities, based on techniques to build non-committing encryption in the ROM [15]. We develop new and entirely different techniques to achieve positive results for inner product FE in the GGM under a definition stronger than many-AD-SIM.

As an illustrative example, consider the scenario where the adversary has the encryption key. In this setting, the adversary may encrypt any vector of his

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

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

## 2 Preliminaries

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

### 2.1 Functional Encryption

A functional encryption scheme $\mathcal{FE}$ consists of four algorithms defined as follows.

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

– $\mathsf{Encrypt}(\mathsf{PP}, \mathsf{EK}, \boldsymbol{x})$ is a p.p.t. algorithm that takes as input the public parameters $\mathsf{PP}$, the encryption key $\mathsf{EK}$ and an input message $\boldsymbol{x} \in \mathcal{X}_{\mathsf{PP}}$ and outputs a ciphertext $\mathsf{CT}_{\boldsymbol{x}}$.
– $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_f, \mathsf{CT}_{\boldsymbol{x}})$ is a deterministic algorithm that takes as input the public parameters $\mathsf{PP}$, the secret key $\mathsf{SK}_f$ and a ciphertext $\mathsf{CT}_{\boldsymbol{x}}$ and outputs $f(\boldsymbol{x})$.

**Definition 1 (Correctness).** *A functional encryption scheme $\mathcal{FE}$ is correct if for all $(\mathsf{PP}, \mathsf{MSK}, \mathsf{EK})$ generated by $\mathsf{Setup}(1^\kappa)$, all $f \in \mathcal{F}_{\mathsf{PP}}$ and $\boldsymbol{x} \in \mathcal{X}_{\mathsf{PP}}$,*

$$\Pr[\mathsf{Decrypt}\big(\mathsf{KeyGen}(\mathsf{PP}, \mathsf{MSK}, f), \mathsf{Encrypt}(\mathsf{PP}, \mathsf{EK}, \boldsymbol{x})\big) \neq f(\boldsymbol{x})]$$

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

*Remark 1.* A functional encryption scheme $\mathcal{FE}$ may permit some *intentional leakage of information*. In this case, the secret $\mathsf{SK}_f$ or the ciphertext $\mathsf{CT}_{\boldsymbol{x}}$ may leak some legitimate information about the function $f$ or the message $\boldsymbol{x}$ respectively. A common example of this type of information is the length of the message $|\boldsymbol{x}|$ that is leaked in any public key encryption scheme. This is captured by [15] via the "empty" key, by [6] by giving this information to the simulator directly and by [9] by restricting to adversaries who do not trivially break the system by issuing challenges that differ in such leakage. We use the approach of [6] and pass on any intentionally leaked information directly to the simulator.

### 2.2 Generic Group (GG) Model Overview

The generic group model [40,48] 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 multiplications 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 2 (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 1 (Schwartz Zippel Lemma).** *Let $g_1, g_2$ be any two* different *$\ell$-variate polynomials with coefficients in field $\mathbb{Z}_p$. Let the degree of the polynomial $g_1 - g_2$ be $t$. Then,*

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

## 3 Wishful Security for Functional Encryption

In this section, we present the dream version security definition for Functional Encryption, which captures data hiding as well as function hiding in the strongest, most intuitive way via the ideal world-real world paradigm. This definition extends and generalizes the definition of [15,9] 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.

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

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

Throughout the interaction, $\mathcal{O}$ maintains a two-dimensional table $\mathcal{T}$ with rows indexed by messages $\boldsymbol{x}_1, \ldots \boldsymbol{x}_{\mathsf{rows}}$ and columns indexed by functions $f_1, \ldots, f_{\mathsf{cols}}$, and the entry corresponding to row $\boldsymbol{x}_i$ and column $f_j$ is $f_j(\boldsymbol{x}_i)$. 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[5]. The environment $\mathsf{Env}$ interacts arbitrarily with the adversary $\mathcal{S}$. The interaction between the players is described below:

[5] For eg., for the inner product functionality, $\mathcal{O}$ needs to be provided the dimension of the vectors.

– **External ciphertexts and keys:**
  - **Ciphertexts:** $\mathsf{Env}$ may send $\mathcal{O}$ ciphertext commands $(\mathsf{CT}, \boldsymbol{x})$ upon which $\mathcal{O}$ creates a new row corresponding to $\boldsymbol{x}$, populates all the newly formed entries $f_1(\boldsymbol{x}), \ldots, f_{\mathsf{cols}}(\boldsymbol{x})$ and returns the newly populated table entires to $\mathcal{S}$.
  - **Keys:** $\mathsf{Env}$ may send $\mathcal{O}$ secret key commands $(\mathsf{SK}, f)$ upon which $\mathcal{O}$ creates a new column corresponding to $f$, populates all the newly formed entries $f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_{\mathsf{rows}})$ and returns the newly populated table entries to $\mathcal{S}$.
– **Switch to public key mode:** Upon receiving a command ($\mathsf{PK\ mode}$) from $\mathsf{Env}$, $\mathcal{O}$ forwards this message to $\mathcal{S}$. From this point on, $\mathcal{S}$ may query $\mathcal{O}$ for the function value corresponding to any message $\boldsymbol{x} \in \mathcal{X}$ of its choice, and any key in the system. Upon receiving command $(\boldsymbol{x}, \mathsf{keys})$, $\mathcal{O}$ updates $\mathcal{T}$ as follows: it adds a new row corresponding to $\boldsymbol{x}$, 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 $\mathsf{Env}$ with which it interacts arbitrarily. Hence, we omit modeling this option in our ideal world. We define $\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa)$ to be the view of $\mathsf{Env}$ in the ideal world.

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

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

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

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

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

While simulation based security has been shown impossible to achieve even for data privacy alone, we will show that the stronger definition presented above

can be achieved against a large class of real world attacks, namely generic attacks. We believe that this provides evidence that FE schemes enjoy far greater security in practice.

## 4 Functional Encryption for Inner Products over Prime Order Groups

We present a new functional encryption scheme for inner products in the encryption key setting from prime order bilinear groups. Our scheme starts from the composite order scheme for inner product FE presented in [36]. It then applies a series of transformations, as developed in [28,25,41,42,38], to convert it to a scheme over prime order groups. We will show our scheme to be be fully simulation secure in the generic group model. To begin with, we define some notation that will be useful to us.

*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 $\boldsymbol{v} = (v_1, \cdots, v_d) \in \mathbb{Z}_p^d$ for some $d \in \mathbb{Z}^+$ and $g \in \mathcal{G}$. Then we define $g^{\boldsymbol{v}} := (g^{v_1}, \ldots, g^{v_d})$. For ease of notation, we will refer to $(g^{v_1}, \ldots, g^{v_d})$ by $(v_1, \ldots, v_d)$. This notation allows us to do scalar multiplication and vector addition over tuples of group elements as:

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

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

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

where the vector dot product $\boldsymbol{v} \cdot \boldsymbol{w}$ in the last term is taken modulo $p$. 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 [38,41,42]. For a fixed dimension $d$, let $\mathbb{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ and $\mathbb{B}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_d^*)$ be two random bases (represented as column vectors) for the vector space $\mathbb{Z}_p^d$. Furthermore, they are chosen so that

$$\begin{pmatrix} \boldsymbol{b}_1^T \\ \vdots \\ \boldsymbol{b}_d^T \end{pmatrix} \cdot \left( \boldsymbol{b}_1^* \cdots \boldsymbol{b}_d^* \right) = \psi \cdot \mathbf{I}_{d \times d}, \tag{1}$$

where $\mathbf{I}_{d \times d}$ is the identity matrix and $\psi \xleftarrow{\$} \mathbb{Z}_p$. Lewko [38] describes a standard procedure which allows one to pick such bases.

We use the notation $(\mathbb{B}, \mathbb{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^3)$ in the rest of this work to describe the selection of such basis vectors for $d = 3$. 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 vector of formal polynomials $a_1\boldsymbol{b}_1 + a_2\boldsymbol{b}_2 + a_3\boldsymbol{b}_3$, and with the tuple $(a_1, a_2, a_3)^*$, the vector $a_1\boldsymbol{b}_1^* + a_2\boldsymbol{b}_2^* + a_3\boldsymbol{b}_3^*$.

*Construction.* The functionality $\mathcal{F} : \mathbb{Z}_p^n \times \mathbb{Z}_p^n \to \{0, 1\}$ is described as $\mathcal{F}(\boldsymbol{x}, \boldsymbol{v}) = 1$ if $\langle \boldsymbol{x} \cdot \boldsymbol{v} \rangle = 0 \mod p$, and 0 otherwise. Let $\mathsf{GroupGen}$ be a group generation algorithm which takes as input a security parameter $\kappa$ and outputs the description of a bilinear group of order $p$, where $p$ is a $\kappa$-bit prime. In the description of the scheme and in the proof, we will "work in the exponent" for ease of notation as described at the beginning of this section.

The four algorithms $\mathsf{Setup}, \mathsf{KeyGen}$, $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ are defined as follows.

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

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

- $\mathsf{Encrypt}(\mathsf{EK}, \boldsymbol{x})$: Let $\boldsymbol{x} = (x_1, \ldots, x_n)$, $x_i \in \mathbb{Z}_p$. Let $s, \alpha, r_1, \ldots, r_n \xleftarrow{\$} Z_p$ and construct $\mathsf{CT}_{\boldsymbol{x}} = (C_0, C_1, \ldots, C_n)$ as

$$C_0 = s \cdot P \cdot \boldsymbol{b}_1,$$

  and for $i \in [1, n]$,

$$C_i = s \cdot (H_i \cdot \boldsymbol{b}_1 + R_i \cdot \boldsymbol{b}_3) + \alpha \cdot x_i \cdot (Q \cdot \boldsymbol{b}_2 + R_0 \cdot \boldsymbol{b}_3) + r_i \cdot R \cdot \boldsymbol{b}_3.$$

- $\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{v})$: Let $\boldsymbol{v} = (v_1, \ldots, v_n), v_i \in \mathbb{Z}_p$. Let $\delta_1, \ldots, \delta_n, \zeta, T \xleftarrow{\$} \mathbb{Z}_p$ and construct $\mathsf{SK}_{\boldsymbol{v}} = (K_0, K_1, \ldots, K_n)$ as

$$K_0 = \left(-\sum_{i=1}^{n} H_i \cdot \delta_i\right) \cdot \boldsymbol{b}_1^* + T \cdot \boldsymbol{b}_3^*,$$

  and for $i \in [1, n]$,

$$K_i = \delta_i \cdot P \cdot \boldsymbol{b}_1^* + Q \cdot \zeta \cdot v_i \cdot \boldsymbol{b}_2^*.$$

- $\mathsf{Decrypt}(\mathsf{SK}_{\boldsymbol{v}}, \mathsf{CT}_{\boldsymbol{x}})$: Compute $b = \boldsymbol{e}(C_0, K_0) \cdot \prod_{i=1}^{i=n} \boldsymbol{e}(C_i, K_i)$ 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 dimension 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 Proof of Security

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

### 5.1 Simulator Construction

*Intuition.* Broadly speaking, our simulator will run the adversary and provide secret keys and ciphertexts to him, as well as simulate the GG oracle. Our simulator maintains a table where it associates each group handle that it issues to the adversary with a formal polynomial. Through its interaction with the generic group oracle (played by $\mathcal{S}$), $\mathcal{A}$ may learn relations between the group handles that it obtains. Note that since we are in the GG model, $\mathcal{A}$ will only be able to learn admissible relations (Definition 2). Whatever dependencies $\mathcal{A}$ learns, $\mathcal{S}$ programs these using its table. To do this, it keeps track of what $\mathcal{A}$ is doing via its requests to the GG oracle, extracts necessary information from $\mathcal{A}$ cleverly where required and sets up these (formal polynomial) relations, thus ensuring that the real and ideal world views are indistinguishable. This is tricky in the public key mode, where the adversary may encrypt messages of its choice (using potentially bad randomness) and attempt to learn relations with existing keys using arbitrary generic group operations. In this case, the simulator needs to be able to extract the message from the adversary, obtain the relevant function values from the oracle, and program the dependencies into the generic group.

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

- **Initialization:** $\mathcal{S}$ constructs a table called *simulation table* to simulate the GG oracle $(p, \mathcal{G}, \mathcal{G}_T, e)$. A simulation table consists of two parts one each for the source group $\mathcal{G}$ and the target group $\mathcal{G}_T$ respectively. Each part is a list that contains two columns labelled formal polynomial and group handle respectively. Group handles are strings from $\{0,1\}^{2\kappa}$. A formal polynomial is a multivariate polynomial defined over $\mathbb{Z}_p$. We assume that there is a canonical ordering amongst the variables used to create the formal polynomial entries and thus each polynomial may be represented by a unique canonical representation.
- **Setup:** Upon receiving the dimension $n$ of message and key space 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 picks 18 new formal variables that represent the bases $(\mathbb{B}, \mathbb{B}^*) \leftarrow \mathsf{Dual}(\mathbb{Z}_p^3)$, as well as a new formal variable $\psi$. Next, $\mathcal{S}$ picks new formal variables $P, Q, R, R_0, \{H_i, R_i\}_{i=1}^{i=n}$. 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 $\mathsf{MSK} = \left\{P, Q, \{H_i\}_{i=1}^{i=n}, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*\right\}$. 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 $\mathsf{PP} = (p, \mathcal{G}, \mathcal{G}_T, e)$. This amounts to $\mathcal{S}$ providing the adversary with oracle access to $\mathcal{G}, \mathcal{G}_T, e$ and sending him $p$.
– **Request for Public Key:** When $\mathcal{S}$ receives the command $\mathsf{PK\ mode}$ from $\mathcal{O}$, he sends the group handles of $\mathsf{EK}$ to $\mathcal{A}$.
– **External Ciphertexts and Keys:** At any time, $\mathcal{S}$ may receive a message of the form $\mathsf{MsgIdx}_{\boldsymbol{x}}$, $f_1(\boldsymbol{x}), \ldots, f_{\mathsf{cols}}(\boldsymbol{x})$ from $\mathcal{O}$. In response:
  - $\mathcal{S}$ follows the outline of the $\mathsf{Encrypt}$ algorithm in the following way. He picks new formal variables $s, \alpha, \{x_i\}_{i=1}^n, \{r_i\}_{i=1}^n$ (all indexed by the particular index $\mathsf{MsgIdx}_{\boldsymbol{x}}$, dropped here for notational convenience). He then constructs the formal polynomials associated with the following 3-tuples:

$$C = \Big\{C_0 = (sP,\ 0,\ 0), \big\{C_i = (sH_i,\ \alpha x_i Q,\ sR_i + \alpha x_i R_0 + r_i R)\big\}_{i=1}^n\Big\},$$

    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 $f_1(\boldsymbol{x}),\ \ldots,\ f_{\mathsf{cols}}(\boldsymbol{x})$ that were received. To do this, $\mathcal{S}$ retrieves the formal polynomials associated with all the keys in the table $\{K^j = (K_0^j, K_1^j, \ldots, K_n^j)\}_{j \in [\mathsf{cols}]}$. Then, for each $j$, he computes the formal polynomials associated with the decrypt operation between $C$ and $K^j$, i.e., $b = \boldsymbol{e}(C_0, K_0^j) \cdot \prod_{i=1}^{i=n} \boldsymbol{e}(C_i, K_i^j)$. If $f_j(\boldsymbol{x}) = 0$, 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 $\mathsf{KeyIdx}_j, f_j(\boldsymbol{x}_1), \ldots, f_j(\boldsymbol{x}_{\mathsf{rows}})$ message by following the $\mathsf{KeyGen}$ 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 C.

**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 $\mathsf{expr}$ is an expression containing a term of the form $\sum_{i=1}^n c_i \upsilon_i$ 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 $\boldsymbol{x} = (c_1, \ldots, c_n)$. $\mathcal{S}$ then sends the message $(\boldsymbol{x}, \mathsf{keys})$ to $\mathcal{O}$. Upon receiving $\mathsf{MsgIdx}_{\boldsymbol{x}}, f_1(\boldsymbol{x}), \ldots, f_{\mathsf{cols}}(\boldsymbol{x})$ from $\mathcal{O}$, $\mathcal{S}$ computes the decrypt expressions for the extracted message with all the keys and programs the linear relations in the generic group oracle as in the previous step.

In the full version of the paper we show that the real and ideal worlds are indistinguishable to $\mathsf{Env}$. Formally, we prove the following theorem:

**Theorem 2.** *For all p.p.t. adversaries $\mathcal{A}$, the simulator $\mathcal{S}$ constructed in Section 5.1 is such that for all* $\mathsf{Env}$ *with auxiliary input $z$, $\{\mathsf{VIEW}_{\mathsf{IDEAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*} \approx \{\mathsf{VIEW}_{\mathsf{REAL}}(1^\kappa, z)\}_{\kappa\in\mathbb{Z}^+, z\in\{0,1\}^*}$ in the generic group model.*

### 5.2 Concrete parameters

From the proof of Theorem 2, we observe that the only case for distinguishability between real and ideal worlds is the hybrid where we move from Generic Group elements to polynomials in formal variables.

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

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

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

### 5.3 Practical considerations

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

We also note that our scheme is shown to be secure against generic attacks and that non-generic attacks do exist in all known bilinear groups. However, a long list of previous research focuses on constructing elliptic curves where the complexity of any non-generic attack is worse than generic attacks [24,23,8,22,12] making our work relevant and meaningful. These constructions are practical as well. Hence we believe that FE constructions over suitably chosen elliptic curve groups have the potential of being practically secure.

## References


1. Voltage security. http://www.voltage.com/.
2. S. Agrawal, S. Agrawal, and M. Prabhakaran. Cryptographic agents: Towards a unified theory of computing on encrypted data. To appear in Eurocrypt 2015.
3. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.
4. S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010.
5. S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.
6. S. Agrawal, S. Gurbanov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013.
7. P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology Eprint Arxiv, 2013. http://eprint.iacr.org/2013/689.pdf.
8. D. Aranha, L. Fuentes-Castaeda, E. Knapp, A. Menezes, and F. Rodrguez-Henrquez. Implementing pairings at the 192-bit security level. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography Pairing 2012*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195. Springer Berlin Heidelberg, 2013.
9. M. Barbosa and P. Farshim. On the semantic security of functional encryption schemes. In K. Kurosawa and G. Hanaoka, editors, *Public-Key Cryptography, PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 143–161. Springer Berlin Heidelberg, 2013.
10. M. Bellare and A. O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS*, 2013.
11. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
12. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
13. D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, 2013.
14. D. Boneh, A. Raghunathan, and G. Segev. Function-private subspace-membership encryption and its applications. In *Asiacrypt*, 2013.
15. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.
16. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.
17. X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.
18. R. Canetti, G. Rothblum, and M. Varia. Obfuscation of hyperplane membership. In *TCC*, 2010.
19. A. D. Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO*, 2013.
20. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.
21. C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

22. C. Costello. Particularly friendly members of family trees. *IACR Cryptology ePrint Archive*, 2012:72, 2012.
23. D. 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.
24. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.
25. D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.
26. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
27. S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.
28. S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2010.
29. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.
30. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT*, pages 578–602, 2014.
31. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.
32. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.
33. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012.
34. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute based encryption for circuits. In *STOC*, 2013.
35. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
36. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
37. N. Koblitz and A. Menezes. Another look at generic groups. In *Advances in Mathematics of Communications*, pages 13–28, 2006.
38. A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.
39. A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
40. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. In *MATHEMATICAL NOTES*, volume 55, 1994.
41. T. Okamoto and K. Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008.
42. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *Asiacrypt*, 2009.

43. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/.
44. A. Sahai and B. Waters. Functional encryption:beyond public key cryptography. Power Point Presentation, 2008. http://userweb.cs.utexas.edu/~bwaters/ presentations/files/functional.ppt.
45. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
46. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
47. E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.
48. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266. Springer-Verlag, 1997.
49. B. Waters. Functional encryption for regular languages. In *CRYPTO*, 2012.

## A Notation

We say that a function $f : \mathbb{Z}^+ \to \mathbb{R}^+$ is negligible if $f(\lambda) \in \lambda^{-\omega(1)}$. For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over some set $\Omega$ 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 $\mathcal{D}_1(\lambda)$ and $\mathcal{D}_2(\lambda)$ are statistically close or statistically indistinguishable if $\mathrm{SD}\big(\mathcal{D}_1(\lambda), \mathcal{D}_2(\lambda)\big)$ is a negligible function of $\lambda$.

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

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

is a negligible function of $\lambda$.

We use $a \xleftarrow{\$} S$ to denote that $a$ is chosen uniformly at random from the set $S$.

## B Correctness of Inner Product Scheme

For any $\mathsf{SK}_{\boldsymbol{v}}$ and $\mathsf{CT}_{\boldsymbol{x}}$, 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.

$$
\begin{aligned}
\boldsymbol{e}(C_0, K_0) &= \boldsymbol{e}\left( (sP \cdot \boldsymbol{b}_1), \big((-\sum_{i=1}^{n} H_i \cdot \delta_i) \cdot \boldsymbol{b}_1^* + T \cdot \boldsymbol{b}_3^*\big) \right) \\
&= (-sP \sum_{i=1}^{n} H_i \delta_i) \cdot (\boldsymbol{b}_1^T \cdot \boldsymbol{b}_1^*) + (\times)(\boldsymbol{b}_1^T \cdot \boldsymbol{b}_3^*) \\
&= \psi(-sP \sum_{i=1}^{n} H_i \delta_i) \text{ (by Equation 1)} \\
\boldsymbol{e}(C_i, K_i) &= \boldsymbol{e}\left( (s(H_i \cdot \boldsymbol{b}_1 + R_i \cdot \boldsymbol{b}_3) + \alpha \cdot x_i \cdot (Q \cdot \boldsymbol{b}_2 + R_0 \cdot \boldsymbol{b}_3) + r_i \cdot \boldsymbol{b}_3), \right. \\
&\quad \left. (\delta_i \cdot P \cdot \boldsymbol{b}_1^* + Q \cdot \zeta \cdot v_i \cdot \boldsymbol{b}_2^*) \right) \\
&= (sH_i\delta_i P)\boldsymbol{b}_1^T \boldsymbol{b}_1^* + (\alpha x_i Q \cdot Q\zeta v_i)\boldsymbol{b}_2^T \boldsymbol{b}_2^* + (\times)(\boldsymbol{b}_1^T \cdot \boldsymbol{b}_2^* + \boldsymbol{b}_2^T \cdot \boldsymbol{b}_1^* + \\
&\qquad \boldsymbol{b}_3^T \cdot \boldsymbol{b}_1^* + \boldsymbol{b}_3^T \cdot \boldsymbol{b}_2^*) \\
&= \psi(sH_i\delta_i P + \alpha\zeta Q^2 x_i v_i) \text{ (by Equation 1)} \\
\text{Thus,} \quad & \boldsymbol{e}(C_0, K) \cdot \prod_{i=1}^{i=n} \boldsymbol{e}(C_i, K_i) \\
&= \psi(-sP \sum_{i=1}^{n} H_i \delta_i) + \sum_{i=1}^{n} \big(\psi(sH_i\delta_i P + \alpha\zeta Q^2 x_i v_i)\big) \\
&= \psi Q^2 \alpha\zeta (\sum_{i=1}^{n} x_i v_i)
\end{aligned}
$$

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

## C 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 to the first part of the table. 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 first part of the simulation table and returns the newly generated handle to $\mathcal{A}$. He 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 and 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 second part of the simulation table and returns the newly generated handle to $\mathcal{A}$.

# D Obfuscation scheme

In this section we present an obfuscation scheme for hyperplane membership. We begin by providing a definition for obfuscation schemes from [18].

## D.1 Formal definition of obfuscation

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

- **Approximate functionality**: There exists a negligible function $\epsilon$ such that for every $\kappa$, every circuit $C \in C_\kappa$ and every $x$ in the input space of $C$, $Pr[\mathcal{O}(C)(x) = C(x)] > 1 - \epsilon(\kappa)$, where the probability is over the randomness of $\mathcal{O}$. If this probability always equals 1, then we say that $\mathcal{O}$ has exact functionality.
- **Polynomial slowdown**: There exists a polynomial $q$ such that for every $\kappa$, every circuit $C \in C_\kappa$, and every possible sequence of coin tosses for $\mathcal{O}$, the circuit $\mathcal{O}(C)$ runs in time at most $q(|C|)$.
- **Virtual black-box**: For every p.p.t. adversary $A$ and polynomial $\delta$, there exists a p.p.t. simulator $S$ such that for all sufficiently large $\kappa$, and for all $C \in C_\kappa$,

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

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

### D.2 Construction

Hyperplane membership testing amounts to computing inner-product over a vector space [18], for which we constructed a functional encryption scheme in Section 4. The circuit family for hyperplane membership, though, is defined in a slightly different way because the circuits have the description of a hyperplane hardwired in them, which is just a vector. More formally, let $p$ be a $\kappa$-bit prime and $n$ a positive integer ($n > 1$). For a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$, let $F_{\boldsymbol{v}}$ be a circuit which on input $\boldsymbol{x} \in \mathbb{Z}_p^n$ outputs 1 if $\langle \boldsymbol{x} \cdot \boldsymbol{v} \rangle = 0 \mod p$, and 0 otherwise. We provide an obfuscator $\mathcal{O}$ for the function family $\mathcal{F}_{p,n} = \{F_{\boldsymbol{v}} \mid \boldsymbol{v} \in \mathbb{Z}_p^n\}$, basing it directly on the functional encryption scheme from Section 4.

- Run $\mathsf{Setup}(1^\kappa)$ to obtain $(\mathsf{PP}, \mathsf{MSK}, \mathsf{EK})$. Publish these values as public parameters.
- **Obfuscator $\mathcal{O}$:** On input $\boldsymbol{v} \in \mathcal{F}_{p,n}$, execute $\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{v})$ to get $\mathsf{SK}_{\boldsymbol{v}}$. Output a circuit with $\mathsf{EK}$ and $\mathsf{SK}_{\boldsymbol{v}}$ hardwired. On input $\boldsymbol{x}$, this circuit first computes $\mathsf{CT}_{\boldsymbol{x}} \leftarrow \mathsf{Encrypt}(\mathsf{EK}, \boldsymbol{x})$, then outputs $\mathsf{Decrypt}(\mathsf{SK}_{\boldsymbol{v}}, \mathsf{CT}_{\boldsymbol{x}})$.

### D.3 Proof of security

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

- **Approximate functionality**: The scheme $\mathcal{O}$ achieves exact functionality from the exact correctness of the underlying FE scheme.
- **Polynomial slowdown**: The scheme achieves polynomial slowdown because of the polynomial runtime of $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ algorithms of the underlying FE scheme.
- **Virtual black-box**: The scheme satisfies the virtual black-box property *in the generic group model* from the proof of security of the underlying FE scheme. We defer a formal proof of this last property to the full version.