Simple Functional Encryption Schemes for Inner Products

Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval

Département d'Informatique, École normale supérieure
{michel.abdalla,florian.bourse,angelo.decaro,david.pointcheval}@ens.fr

March 28, 2015

Abstract

Functional encryption is a new paradigm that allows users to finely control the amount of information that is revealed by a ciphertext to a given receiver. Recent papers have focused their attention on constructing schemes for general functionalities at expense of efficiency. Our goal, in this paper, is to construct functional encryption schemes for less general functionalities which are still expressive enough for practical scenarios. We propose a functional encryption scheme for the *inner-product* functionality, meaning that decrypting an encrypted vector $\mathbf{x}$ with a key for a vector $\mathbf{y}$ will reveal only $\langle \mathbf{x}, \mathbf{y} \rangle$ and nothing else, whose security is based on the DDH assumption. Despite the simplicity of this functionality, it is still useful in many contexts like descriptive statistics. In addition, we generalize our approach and present a generic scheme that can be instantiated, in addition, under the LWE assumption and offers various trade-offs in terms of expressiveness and efficiency.

Keywords: Functional Encryption, Inner-Product, Generic Constructions.

---

1 Introduction

Functional Encryption. Whereas, in traditional public-key encryption, decryption is an all-or-nothing affair (i.e., a receiver is either able to recover the entire message using its key, or nothing), in *functional encryption* (FE), it is possible to finely control the amount of information that is revealed by a ciphertext to a given receiver. For example, decrypting an encrypted data set with a key for computing the mean will reveal only the mean computed over the data set and nothing else. Somewhat more precisely, in a functional encryption scheme for functionality F, each secret key (generated by a master authority having a *master secret key*) is associated with value k in some *key space* K; Anyone can encrypt via the public parameters; When a ciphertext Ct_x that encrypts x, in some *message space* X, is decrypted using a secret key Sk_k for value k, the result is $F(k, x)$. A notable subclass of functional encryption is that of *predicate encryption* (PE) which are defined for functionalities whose message space X consists of two subspaces I and M called respectively *index space* and *payload space*. In this case, the functionality F is defined in terms of a predicate $P : K \times I \to \{0, 1\}$ as follows: $F(k, (\mathsf{ind}; \mathsf{m})) = \mathsf{m}$ if $P(k, \mathsf{ind}) = 1$, and $\perp$ otherwise, where $k \in K$, $\mathsf{ind} \in I$ and $\mathsf{m} \in M$. Those schemes are also called *predicate encryption with private-index*. Examples of those schemes are Anonymous Identity-Based Encryption (AIBE) [BF01, Gen06], Hidden Vector Encryption [BW07] and Orthogonality [KSW08, LOS+10, OT12], among the others. On the other hand, when the index ind is easily readable from the ciphertext those schemes are called *predicate encryption with public-index* (PIPE). Examples of PIPE schemes are Identity-Based Encryption (IBE) [Sha84, BF01, Coc01], Attribute-Based Encryption (ABE) [SW05, GPSW06], Functional Encryption for Regular Languages [Wat12].

The standard notion of security for functional encryption is that of *indis-tinguishability-based security* (IND). Informally, it requires that an adversary cannot tell apart which of two messages x_0, x_1 has been encrypted having oracle access to the key generation algorithm under the constraint that, for each k for which the adversary has seen a secret key, it holds that $F(k, x_0) = F(k, x_1)$. This models the idea that an individual's messages are still secure even if an arbitrary number of other users of the system collude against that user. Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10] showed that the IND definition is weak in the sense that a trivially insecure scheme implementing a certain functionality can be proved IND-secure anyway. The authors, then, initiate the study of *simulation-based* (SIM) notions of security for FE, which asks that the "view" of the adversary can be simulated by simulator given neither ciphertexts nor keys but only the corresponding outputs of the functionality on the underlying plaintexts, and shows that SIM-security is not always achievable.

In a recent series of outstanding results, [GGH+13, BCP14, Wat14, GGHZ14] proposed IND-secure FE schemes for general circuits whose security is based either on indistinguishable obfuscation and its variants or polynomial hardness of simple assumptions on multilinear maps. Those schemes are far from being practical and this led us to investigate the possibility of having functional encryption schemes for *functionalities of practical interest* which are still expressive enough for practical scenarios. In doing so, we seek for schemes that offer *simplicity*, in terms of understanding of how the schemes work, and *adaptability* in terms of the possibility of choosing the instantiations and the parameters that better fit the constraints and needs of a specific scenario the user is interested in.

This Work. In this paper, we focus on the *inner-product* functionality, which has several practical applications. For example, in descriptive statistics, the discipline of quantitatively describing the main features of a collection of information, the *weighted mean* is a useful tool.

Here are a few examples:

Slugging average in baseball. A batter's slugging average, also called slugging percentage, is computed by: $\mathsf{SLG} = (1 * \mathsf{SI} + 2 * \mathsf{DO} + 3 * \mathsf{TR} + 4 * \mathsf{HR})/\mathsf{AB}$, where SLG is the slugging percentage, SI is the number of singles, DO the number of doubles, TR the number of triples, HR the number of home runs, and AB is total number of at-bats. Here, each single has a *weight* of 1, each double has a *weight* of 2, etc. The average counts home runs four times as important as singles, and so on. An at-bat without a hit has a *weight* of zero.

Course grades. A teacher might say that the *test average* is 60% of the grade, *quiz average* is 30% of the grade, and a *project* is 10% of the grade. Suppose Alice got 90 and 78 on the tests; 100, 100 and 85 on the quizzes; and an 81 on the project. Then, Alice's test average is $(90 + 78)/2 = 84$, quiz average is $(100 + 100 + 85)/3 = 95$, and her course grade would then be: $.60 \cdot 84 + .30 \cdot 95 + .10 \cdot 81 = 87$.

Our goal then is to design a simple and efficient functional encryption scheme for inner products that can be used, for instance, to compute a weighted mean and to protect the privacy of Alice's grades, in the example involving course grades. In fact, we can imagine that Alice's grades, represented as a vector $\mathbf{x} = (x_1, \ldots, x_\ell)$ in some finite field, says $\mathbb{Z}_p$ for prime p, are encrypted in a ciphertext $\mathsf{Ct}_{\mathbf{x}}$ and the teacher has a secret key $\mathsf{Sk}_{\mathbf{y}}$ for the vector of weights $\mathbf{y} = (y_1, \ldots, y_\ell)$. Then Alice's course grade can be computed as the *inner-product* of $\mathbf{x}$ and $\mathbf{y}$, written as $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i \in [\ell]} x_i \cdot y_i$. We would like to stress here that, unlike the inner-product predicate schemes in [KSW08, LOS+10, OT12], our goal is to output the actual value of the inner product.

A very simple scheme can be constructed to compute the above functionality whose security can be based on the DDH assumption. Informally, it is like this:

$$\begin{aligned} mpk &= \left(\mathbb{G}, (h_i = g^{s_i})_{i \in [\ell]}\right) \\ \mathsf{Ct}_{\mathbf{x}} &= \left(\mathsf{ct}_0 = g^r, (\mathsf{ct}_i = h_i^r \cdot g^{x_i})_{i \in [\ell]}\right) \\ \mathsf{Sk}_{\mathbf{y}} &= \langle \mathbf{s}, \mathbf{y} \rangle = \sum_{i \in [\ell]} s_i \cdot y_i, \end{aligned}$$

where $msk = \mathbf{s} = (s_1, ..., s_\ell)$ is the master secret key used to generate secret keys $\mathsf{Sk}_{\mathbf{y}}$. Then, decryption is done by computing the discrete log of $(\prod_{i \in [\ell]} \mathsf{ct}_i^{y_i})/\mathsf{ct}_0^{\mathsf{Sk}_{\mathbf{y}}}$. Please refer to Section 3 for more details.

Despite its simplicity, this DDH-based scheme can be proved secure, in a selective security model[1], against any adversary that issues an unbounded, but polynomially related to the security parameter, number of secret key queries. The adversary will not learn anything more than what it is implied by the linear combination of their keys.

An astute reader could now ask what happens if an adversary possesses secret keys $\mathsf{Sk}_{\mathbf{y}_i}$, for $i \in [q]$, such that the $\mathbf{y}_i$'s form a basis for $\mathbb{Z}_p^\ell$. Clearly, this adversary can then recover completely $\mathbf{x}$ from the ciphertext and wins the security game. But notice that this has nothing to do with the specific implementation of the functionality, it is something inherent to the functionality itself. This happens also for other functionalities: Consider the case of the *circuit* functionality, where secret keys correspond to Boolean circuits over ℓ Boolean variables and one-bit output, and ciphertexts to vectors in $\{0,1\}^\ell$. Then, an adversary having secret keys for circuits C_i, where C_i extract the i-th bit of the input, can recover completely the input from a ciphertext

[1] In the selective model, the adversary is asked to commit to its challenge before seeing the public parameters.

no matter how the scheme, supporting this circuit functionality, is implemented. This subtle aspect will appear in the security proof, if the adversary asks such a set of secret keys then the simulator will not be able to answer all those queries. Notice that this is reasonable, because it is like requiring security when the adversary possesses the master secret key.

One drawback of the above scheme is that restrictions must be put in place in order to guarantee that the final computed value has small magnitude and the discrete log can be computed efficiently. To overcome this limitation, let us first describe some interesting characteristics of the above scheme. To start, please notice that a ciphertext for a vector $\mathbf{x}$ consists of ElGamal ciphertexts [ElG84] $(g^r, (h_i^r \cdot g^{x_i})_i)$ under public keys $h_i = g^{s_i}$, sharing the same randomness r. Then, a secret key for a vector $\mathbf{y}$ consists of a linear combination of the underlying ElGamal secrets s_i. Now notice that, by the ElGamal scheme's homomorphic properties, it holds that

$$\prod_{i\in[\ell]} \mathsf{ct}_i^{y_i} = \prod_{i\in[\ell]} h_i^{r\cdot y_i} \cdot g^{x_i \cdot y_i} = h^r \cdot g^{\langle \mathbf{x},\mathbf{y}\rangle}$$

where h is an ElGamal public key corresponding to secret key $\langle \mathbf{s}, \mathbf{y}\rangle$. The above observations point out that, by possibly combining public-key encryption schemes secure under randomness reuse [BBS03], and having specific syntactical, non-security-related, properties, we can generalize the above construction with the aim of (1) having a scheme whose security can be based on different assumptions and that can provide different trade-offs in terms of efficiency and expressiveness, (2) have a generic proof of security that reduces security to that of the underlying public-key encryption scheme. We present our generalization in Section 4.

Related Work. One of the first example of investigation on reductions between various primitives has been given by [Rot11] who shows a simple reduction between any semantically secure private-key encryption scheme which possesses a simple homomorphic, namely that the product of two ciphertexts Ct_1 and Ct_2, encrypting plaintexts m_1 and m_2, yields a new ciphertext $\mathsf{Ct} = \mathsf{Ct}_1 \cdot \mathsf{Ct}_2$ which decrypts to $\mathsf{m}_1 + \mathsf{m}_2 \mod 2$. Goldwasser *et al.* [GLW12] investigated the construction of PIPE schemes for specific predicates. In particular, they show how public-key encryption schemes which possess a linear homomorphic property over their keys as well as hash proof system features with certain algebraic structure can be used to construct an efficient identity-based encryption (IBE) scheme that is secure against bounded collusions (BC-IBE). This weaker security notion restricts the adversary to issue only a bounded number of secret keys during the security game. In more details, they rely on a public-key encryption scheme whose secret keys and public keys are elements of respective groups (with possibly different operations, which we denote by $+$ and $\cdot$), and there exists an homomorphism μ such that $\mu(\mathsf{sk}_0 + \mathsf{sk}_1) = \mu(\mathsf{sk}_0) \cdot \mu(\mathsf{sk}_1)$, where $\mu(\mathsf{sk}_0)$ and $\mu(\mathsf{sk}_1)$ are valid public keys for which sk_0 and sk_1 yield correct decryption, respectively. Then, to obtain a BC-IBE, the construction by Goldwasser *et al.* generates multiple public-key/secret-key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$, letting the public-parameters and the master secret key of the scheme be $\mathsf{params} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and $msk = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$, respectively. Then, an efficient map ϕ associates every identity ID with a vector $[id_1, \ldots, id_\ell]$, and a message m is encrypted for an identity ID as the ciphertext $\mathsf{Ct} = \mathsf{Encrypt}(\mathsf{pk}_{\mathsf{ID}}, m)$, where $\mathsf{pk}_{\mathsf{ID}} = \prod_{i=1}^n \mathsf{pk}_i^{\mathsf{ID}_i}$. Then, μ guarantees that Ct can be decrypted using $\mathsf{sk}_{\mathsf{ID}} = \sum_{i=1}^n \mathsf{ID}_i \cdot \mathsf{sk}_i$, since by the homomorphism it holds that $\mathsf{pk}_{\mathsf{ID}} = \mu(\mathsf{sk}_{\mathsf{ID}})$. The map ϕ is subject to a combinatorial requirement that disallows computing $\mathsf{sk}_{\mathsf{ID}}$ given $\mathsf{sk}_{\mathsf{ID}'}$ for t different $\mathsf{ID}' \neq \mathsf{ID}$. Later, Tessaro and Wilson [TW14] presented generic constructions of BC-IBE which rely on encryption schemes that solely satisfy the standard security notion of semantic security

in addition to some syntactical, non-security-related, properties. We will use Tessaro and Wilson to present a generalization of our results.

As already mentioned, in a recent series of outstanding results, [GGH+13, BCP14, Wat14, GGHZ14] proposed IND-secure FE scheme for general circuits whose security is based either on indistinguishable obfuscation and its variants or polynomial hardness of simple assumptions on multilinear maps. Clearly, those schemes can be used to implement the inner-product functionality, but this would defeat our main goal which is to construct somewhat efficient functional encryption schemes for less general functionalities which are still expressive enough for practical scenarios.

In another line of research, Katz, Sahai, and Waters [KSW08] proposed a functional encryption scheme for a functionality called *orthogonality* (in the literature it is also know as inner-product encryption), meaning that decrypting an encrypted vector $\mathbf{x}$ with a key for a vector $\mathbf{y}$ will reveal only if $\langle \mathbf{x}, \mathbf{y} \rangle$ is equals to zero (meaning that the two vectors are orthogonal) or not, and nothing else. This functionality is of little help in our case because what we need is for the decryptor to be able to recover the value $\langle \mathbf{x}, \mathbf{y} \rangle$.

Parameter Sizes For Our Constructions. In Table 1, we present the size of the parameters and ciphertexts for our concrete construction. Each column refers to an instantiation of our general scheme and is indexed by its underlying assumption. Each row describes the size of an element of the scheme. The master public key size does not include public parameters that can be re-used such that $(g, \mathbf{A})$. Message space is the number of different messages that can be decrypted using this instantiation. The DDH instantiation can give short ciphertexts and keys using elliptic curves, but requires the computation of a discrete logarithm in the decryption, which makes it usable for small message space only. LWE enables short ciphertexts and keys together with a message space that is independent of the security parameter, which allows shorter ciphertexts for a smaller message space.

	DDH	LWE
mpk	$\ell \log p$	$m\ell \log q$
msk	$\ell \log p$	$n\ell \log q$
$\mathsf{Ct}_{\mathbf{x}}$	$(\ell + 1) \log p$	$(n + \ell) \log q$
$\mathsf{Sk}_{\mathbf{y}}$	$\log p$	$n \log q$
message space	bounded by computation	ℓM^2
$\lvert\{\langle \mathbf{x}, \mathbf{y} \rangle\}\rvert$	of discrete logarithms	computations mod p

Table 1: Parameter Sizes. Here, ℓ is the length of the vectors encrypted in the ciphertexts and encoded in the secret keys. In DDH, p is the size of a group element. In LWE, q is the size of a group element, p is the order of $M^2\ell$, and M is a bound on the message space.

2 Basic Tools

In this section, we recall some of the definitions and basic tools that will be used in the remaining sections, such as the syntax of code-based games, functional encryption and the assumptions.

2.1 Notation and Conventions

Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$, then $\{0,1\}^n$ denotes the set of n-bit strings, and $\{0,1\}^*$ is the set of all bit strings. The empty string is denoted ε. More generally, if S is a set, then S^n is the set of n-tuples of elements of S, $S^{\leq n}$ is the set of tuples of length at most n. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. If S is finite, then $x \xleftarrow{R} S$ denotes the assignment to x of an element chosen uniformly at random from S. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ on input x, and if $\mathcal{A}$ is randomized, then $y \xleftarrow{R} \mathcal{A}(x)$ denotes that the output of an execution of $\mathcal{A}(x)$ with fresh coins is assigned to y. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for polynomial time and "PTA" for polynomial-time algorithm or adversary. We denote by $\lambda \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for every $c \in \mathbb{N}$ there exists a $\lambda_c \in \mathbb{N}$ such that $\nu(\lambda) \leq \lambda^{-c}$ for all $\lambda > \lambda_c$, and it is said to be *overwhelming* if the function $|1 - \nu(\lambda)|$ is negligible.

Let $\mathbf{e}_i$ be the vector with all 0 but one 1 in the i-th position.

Code-Based Games. We use the code-based game-playing [BR06] to define the security notions. In such games, there exist procedures for initialization (**Initialize**) and finalization (**Finalize**) and procedures to respond to adversary oracle queries. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted $\mathrm{G}(\mathcal{A})$, is called the output of the game, and "$\mathrm{G}(\mathcal{A}) = y$" denotes the event that the output takes a value y. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true.

2.2 Public-Key Encryption

Definition 2.1 [Public-Key Encryption Scheme] A *public-key encryption* (PKE) scheme $\mathcal{E}$ is a tuple $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ of 3 algorithms:

1. $\mathsf{Setup}(1^\lambda)$ outputs *public* and *secret* keys $(\mathsf{pk}, \mathsf{sk})$ for *security parameter* λ;
2. $\mathsf{Encrypt}(\mathsf{pk}, m)$, on input public key pk and message m in the allowed message space, outputs *ciphertext* Ct;
3. $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Ct})$ on input secret key sk and ciphertext Ct, outputs messages m'.

In addition we make the following *correctness* requirement: for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all messages m and ciphertexts $\mathsf{Ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m)$, we have that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Ct}) = m$ except with negligible probability.

We often also allow public-key encryption schemes to additionally depend on explicit public parameters params (randomly generated in an initial phase and shared across multiple instances of the PKE scheme) on which all of Setup, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ are allowed to depend. Examples include the description of a group $\mathbb{G}$ with its generator g. We will often omit them in the descriptions of generic constructions from PKE schemes.

Indistinguishability-Based Security. We define *security against chosen-plaintext attacks* (IND-CPA *security*, for short) for a PKE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ via the security game depicted on Figure 1. Then, we say that $\mathcal{E}$ is secure against chosen-plaintext attacks

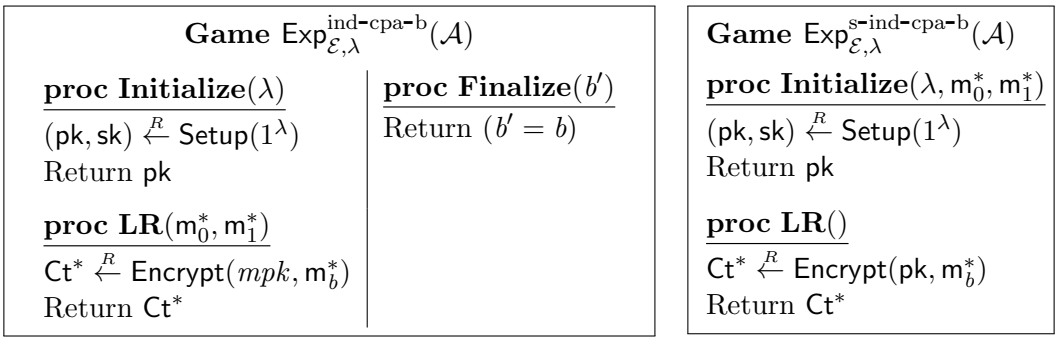

Figure 1: Games $\mathsf{Exp}^{\text{ind-cpa-b}}_{\mathcal{E},\lambda}(\mathcal{A})$ and $\mathsf{Exp}^{\text{s-ind-cpa-b}}_{\mathcal{E},\lambda}(\mathcal{A})$ define IND-CPA and s-IND-CPA security (respectively) of $\mathcal{E}$. The procedure **Finalize** is common to both games, which differ in their **Initialize** and **LR** procedures.

(IND-CPA secure, for short) if

$$\left|\Pr[\mathsf{Exp}^{\text{ind-cpa-0}}_{\mathcal{E},\lambda}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^{\text{ind-cpa-1}}_{\mathcal{E},\lambda}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

We also define *selective security against chosen-plaintext attacks* (s-IND-CPA *security*, for short) when the challenge messages m_0^* and m_1^* have to be chosen before hand. Actually, in this case, the procedures **Initialize** and **LR** can be merged into an **Initialize** procedure that outputs both the public key pk and the challenge ciphertext Ct^*.

2.3 Functional Encryption

Following Boneh *et al.* [BSW11], we start by defining the notion of functionality and then that of functional encryption scheme $\mathcal{FE}$ for functionality F.

Definition 2.2 [Functionality] A *functionality* F defined over (K, X) is a function $F : K \times X \to \Sigma \cup \{\perp\}$ where K is the *key space*, X is the *message space* and Σ is the *output space* and $\perp$ is a special string not contained in Σ. Notice that the functionality is undefined for when either the key is not in the key space or the message is not in the message space.

Definition 2.3 [Functional Encryption Scheme] A *functional encryption* (FE) scheme $\mathcal{FE}$ for functionality F is a tuple $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ of 4 algorithms:

1. $\mathsf{Setup}(1^\lambda)$ outputs *public* and *master secret* keys (mpk, msk) for *security parameter* λ;
2. $\mathsf{KeyDer}(msk, k)$, on input a master secret key msk and *key* $k \in K$ outputs *secret key* sk_k;
3. $\mathsf{Encrypt}(mpk, x)$, on input public key mpk and *message* $x \in X$ outputs *ciphertext* Ct;
4. $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_k)$ outputs $y \in \Sigma \cup \{\perp\}$.

We make the following *correctness* requirement: for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, all $k \in K$ and $m \in M$, for $\mathsf{sk}_k \leftarrow \mathsf{KeyDer}(msk, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Encrypt}(mpk, m)$, we have that $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_k) = F(k, m)$ whenever $F(k, m) \neq \perp$[2], except with negligible probability.

[2]See [BO13, ABN10] for a discussion about this condition.

Indistinguishability-Based Security. For a functional encryption scheme $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ for functionality F, defined over (K, X), we define *security against chosen-plaintext attacks* (IND-FE-CPA *security*, for short) via the security game depicted on Figure 2. Then, we say that $\mathcal{FE}$ is secure against chosen-plaintext attacks (IND-FE-CPA secure, for short) if

$$\left|\Pr[\mathsf{Exp}^{\text{ind-fe-cpa-0}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^{\text{ind-fe-cpa-1}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

We also define *selective security against chosen-plaintext attacks* (s-IND-FE-CPA *security*, for short) when the challenge messages m_0^* and m_1^* have to be chosen before hand.

Game $\mathsf{Exp}^{\text{ind-fe-cpa-}b}_{\mathcal{FE},\lambda}(\mathcal{A})$

proc Initialize(λ)
$(mpk, msk) \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
$V \leftarrow \emptyset$
Return mpk

proc KeyDer(k)
$V \leftarrow V \cup \{k\}$
$\mathsf{sk}_k \xleftarrow{R} \mathsf{KeyDer}(msk, k)$
Return sk_k

proc LR$(\mathsf{m}_0^*, \mathsf{m}_1^*)$
$\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathsf{m}_b^*)$
Return Ct^*

proc Finalize(b')
if $\exists k \in V$ such that
$F(k, \mathsf{m}_0^*) \neq F(k, \mathsf{m}_1^*)$
then return false
Return $(b' = b)$

Game $\mathsf{Exp}^{\text{s-ind-fe-cpa-}b}_{\mathcal{FE},\lambda}(\mathcal{A})$

proc Initialize$(\lambda, \mathsf{m}_0^*, \mathsf{m}_1^*)$
$(mpk, msk) \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
$V \leftarrow \emptyset$
Return mpk

proc LR$()$
$\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathsf{m}_b^*)$
Return Ct^*

Figure 2: Games $\mathsf{Exp}^{\text{ind-fe-cpa-}b}_{\mathcal{FE},\lambda}(\mathcal{A})$ and $\mathsf{Exp}^{\text{s-ind-fe-cpa-}b}_{\mathcal{FE},\lambda}(\mathcal{A})$ define IND-FE-CPA and s-IND-FE-CPA security (respectively) of $\mathcal{FE}$. The procedures **KeyDer** and **Finalize** are common to both games, which differ in their **Initialize** and **LR** procedures.

Inner-Product Functionality. In this paper we are mainly interested in the *inner-product functionality* over $\mathbb{Z}_p$ (IP, for short) defined in the following way. It is a family of functionalities with key space K_ℓ and message space X_ℓ both consisting of vectors in $\mathbb{Z}_p$ of length ℓ: for any $k \in K_\ell, x \in X_\ell$ the functionality $\mathsf{IP}_\ell(k, x) = \langle k, x \rangle \mod p$. When it is clear from the context we remove the reference to the length ℓ.

3 Inner-Product from DDH

In this section, we present our first functional encryption scheme for the inner-product functionality whose security can be based on the plain DDH assumption.

The Decisional Diffie-Hellman assumption. Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time algorithm that takes as input a security parameter 1^λ, and outputs a triplet $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a group of order p that is generated by $g \in \mathbb{G}$, and p is an λ-bit prime number. Then, the *Decisional Diffie-Hellman (DDH) assumption* states that the tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable, where $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, and $a, b, c \in \mathbb{Z}_p$ are chosen independently and uniformly at random.

Construction 3.1 [DDH-IP Scheme] We define our functional encryption scheme for the inner-product functionality $\mathsf{IP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ samples $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $\mathbf{s} = (s_1, \ldots, s_\ell) \leftarrow \mathbb{Z}_p^\ell$, and sets $mpk = (h_i = g^{s_i})_{i\in[\ell]}$ and $msk = \mathbf{s}$. The algorithm returns the pair (mpk, msk);
- $\mathsf{Encrypt}(mpk, \mathbf{x})$ on input master public key mpk and message $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_p^\ell$, chooses a random $r \leftarrow \mathbb{Z}_p$ and computes $\mathsf{ct}_0 = g^r$ and, for each $i \in [\ell]$, $\mathsf{ct}_i = h_i^r \cdot g^{x_i}$. Then the algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$;
- $\mathsf{KeyDer}(msk, \mathbf{y})$ on input master secret key msk and vector $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}_p^\ell$, computes and outputs secret key $\mathsf{sk}_\mathbf{y} = \langle \mathbf{y}, \mathbf{s} \rangle$;
- $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_\mathbf{y})$ on input master public key mpk, ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$ and secret key $\mathsf{Sk}_\mathbf{y}$ for vector $\mathbf{y}$, returns the discrete logarithm in basis g of

$$\prod_{i\in[\ell]} \mathsf{ct}_i^{y_i} / \mathsf{ct}_0^{\mathsf{sk}_\mathbf{y}}.$$

Correctness. For all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, all $\mathbf{y} \in \mathbb{Z}_p^\ell$ and $\mathbf{x} \in \mathbb{Z}_p^\ell$, for $\mathsf{sk}_\mathbf{y} \leftarrow \mathsf{KeyDer}(msk, \mathbf{y})$ and $\mathsf{Ct} \leftarrow \mathsf{Encrypt}(mpk, \mathbf{x})$, we have that

$$\begin{aligned}
\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}) &= \frac{\prod_{i\in[\ell]} \mathsf{ct}_i^{y_i}}{\mathsf{ct}_0^{\mathsf{sk}_\mathbf{y}}} = \frac{\prod_{i\in[\ell]} (g^{s_i r + x_i})^{y_i}}{g^{r(\sum_{i\in[\ell]} y_i s_i)}} \\
&= g^{\sum_{i\in[\ell]} y_i s_i r + \sum_{i\in[\ell]} y_i x_i - r(\sum_{i\in[\ell]} y_i s_i)} \\
&= g^{\sum_{i\in[\ell]} y_i x_i} = g^{\langle \mathbf{x}, \mathbf{y} \rangle}.
\end{aligned}$$

The above scheme limits the expressiveness of the functionality that can be computed because in order to recover the final inner-product value a discrete logarithm computation must take place. In the next section, in order to overcome this limitation and to generalize to other settings we will present a generic scheme whose security can be based on the semantic security of the underlying public-key encryption scheme under randomness reuse. Before moving to the generic scheme and its proof of security, we sketch below the proof of security for the above IP scheme to offer to the reader useful intuitions that will be reused in the proof of security of our generic functional encryption scheme for the inner-product functionality.

Theorem 3.2 Under the DDH assumption, the above IP scheme is s-IND-FE-CPA.

Proof Sketch. For the sake of contradiction, suppose that there exists an adversary $\mathcal{A}$ that breaks s-IND-FE-CPA security of our IP scheme with non-negligible advantage. Then, we construct a simulator $\mathcal{B}$ that given in input an instance of the DDH assumption, (g, g^a, g^b, g^c) where c is either ab or uniformly random, breaks it by using $\mathcal{A}$.

If the challenge messages $\mathbf{x}_0$ and $\mathbf{x}_1$ are different, there exists a vector in the message space for which the key shouldn't be known by the adversary ($\mathbf{x}_1 - \mathbf{x}_0$ is one of them).

To generate the master public key, $\mathcal{B}$ first generates secret keys for a basis $(\mathbf{z}_i)$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Setting implicitly a as secret key for $\mathbf{x}_1 - \mathbf{x}_0$, $\mathcal{B}$ generates the master public key using g^a. Actually, once group elements are generated for the basis $(\mathbf{y}_i)$ completed with $(\mathbf{x}_1 - \mathbf{x}_0)$, one can find the public key, for the canonical basis.

To generate the challenge ciphertext, $\mathcal{B}$ chooses a random bit μ and using g^b and g^c, generates a ciphertext for message $\mathbf{x}_\mu$.

Finally, notice that by the constraints of the s-IND-FE-CPA security game, $\mathcal{A}$ is allowed to ask only secret keys for vectors in the vector sub-space generated by the $\mathbf{z}_i$'s, and thus orthogonal to $\mathbf{x}_1 - \mathbf{x}_0$. For those vectors, $\mathcal{B}$ will be able to generate the corresponding secret keys.

Now, if $c = ab$, then the challenge ciphertext is a well-distributed ciphertext of the message $\mathbf{x}_\mu$. On the other hand, if c is random, the challenge ciphertext encrypts message $u\mathbf{x}_0 + (1-u)\mathbf{x}_1$ where u is uniformly distributed and in this case μ is information theoretically hidden: for all $\mathsf{sk}_\mathbf{y}$ asked, since $\mathbf{y}$ is orthogonal to $\mathbf{x}_1 - \mathbf{x}_0$, and thus $\langle \mathbf{x}_0, \mathbf{y}\rangle = \langle \mathbf{x}_1, \mathbf{y}\rangle$, while the ciphertexts decrypt to the inner products

$$\begin{aligned}\langle u\mathbf{x}_0 + (1-u)\mathbf{x}_1, \mathbf{y}\rangle &= u\langle \mathbf{x}_0, \mathbf{y}\rangle + (1-u)\langle \mathbf{x}_1, \mathbf{y}\rangle \\ &= u\langle \mathbf{x}_\mu, \mathbf{y}\rangle + (1-u)\langle \mathbf{x}_\mu, \mathbf{y}\rangle \\ &= \langle \mathbf{x}_\mu, \mathbf{y}\rangle \ .\end{aligned}$$

4 A Generic Inner-Product Encryption Scheme

In this section, we present a generic functional encryption scheme for the inner-product functionality based on any public-key encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ that has the following structural and homomorphic properties.

Structure. $\mathcal{E}$'s secret keys are elements of a group $(G, +, 0_G)$, public keys are elements of group $(H, \cdot, 1_H)$, and messages are elements of $\mathbb{Z}_q$ for some q.

In addition, we require the ciphertexts to consist of two parts ct_0 and ct_1. The first part ct_0 corresponds to some *commitment* $\mathsf{C}(r)$ of the randomness r used for the encryption. The second part ct_1 is the *encryption* $\mathsf{E}(\mathsf{pk}, x; r)$ in a group $(I, \cdot, 1_I)$ of the message x under public key pk and randomness r.

Linear Key Homomorphism. We say that a PKE has *linear key homomorphism* (LKH, for short) if for any two secret keys $\mathsf{sk}_1, \mathsf{sk}_2 \in G$ and any $y_1, y_2 \in \mathbb{Z}_q$, the component-wise G-linear combination formed by $y_1\mathsf{sk}_1 + y_2\mathsf{sk}_2$ can be computed efficiently only using public parameters, the secret keys sk_1 and sk_2 and the coefficients y_1 and y_2. And this combination $y_1\mathsf{sk}_1 + y_2\mathsf{sk}_2$ also functions as a secret key to a public key that can be computed as $\mathsf{pk}_1^{y_1} \cdot \mathsf{pk}_2^{y_2}$, where pk_1 (resp. pk_2) is a public key corresponding to sk_1 (resp. sk_2).

Linear Ciphertext Homomorphism Under Shared Randomness. We say that a PKE has *linear ciphertext homomorphism under shared randomness* (LCH, for short) if it holds that $\mathsf{E}(\mathsf{pk}_1\mathsf{pk}_2, x_1 + x_2; r) = \mathsf{E}(\mathsf{pk}_1, x_1; r) \cdot \mathsf{E}(\mathsf{pk}_2, x_2; r)$.

4.1 Construction

Construction 4.1 [PKE-IP Scheme] Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a PKE scheme with the properties defined above, we define our *functional encryption scheme for the inner-product functionality* $\mathsf{IP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ calls $\mathcal{E}$'s key generation algorithm to generate ℓ independent $(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_\ell, \mathsf{pk}_\ell)$ pairs, sharing the same public parameters params. Then, the algorithm returns $mpk = (\mathsf{params}, \mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and $msk = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$.
- $\mathsf{KeyDer}(msk, \mathbf{y})$ on input master secret key msk and a vector $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}_q^\ell$, computes $\mathsf{sk}_\mathbf{y}$ as an G-linear combination of $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ with coefficients $(y_1, \ldots, y_\ell)$, namely $\mathsf{sk}_\mathbf{y} = \sum_{i\in[\ell]} y_i \cdot \mathsf{sk}_i$.
- $\mathsf{Encrypt}(mpk, \mathbf{x})$ on input master public key mpk and message $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$, chooses shared randomness r in the randomness space of $\mathcal{E}$, and computes $\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r)$ and $\mathsf{ct}_i = \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)$. Then the algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$.

- $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_{\mathbf{y}})$ on input master public key mpk, ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$, and secret key $\mathsf{Sk}_{\mathbf{y}}$ for vector $\mathbf{y} = (y_1, \ldots, y_\ell)$, returns the output of $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y}}, (\mathsf{ct}_0, \prod_{i\in[\ell]} \mathsf{ct}_i^{y_i}))$.

Correctness. For all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, all $\mathbf{y} \in \mathbb{Z}_q^\ell$ and $\mathbf{x} \in \mathbb{Z}_q^\ell$, for $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathsf{KeyDer}(msk, \mathbf{y})$ and $\mathsf{Ct} \leftarrow \mathsf{Encrypt}(mpk, \mathbf{x})$, we have that

$$\begin{aligned}
\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_{\mathbf{y}}) &= \mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y}}, (\mathsf{ct}_0, \prod_{i\in[\ell]} \mathsf{ct}_i^{y_i})) \\
&= \mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y}}, (\mathsf{ct}_0, \prod_{i\in[\ell]} \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)^{y_i})) \\
&= \mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y}}, (\mathsf{ct}_0, \mathcal{E}.\mathsf{Encrypt}(\prod_{i\in[\ell]} \mathsf{pk}_i^{y_i}, \sum_{i\in[\ell]} y_i x_i; r))) \\
&= \sum_{i\in[\ell]} y_i x_i \ .
\end{aligned}$$

by the LCH property. Finally, note that the decryption is allowed because $(\mathsf{Sk}_{\mathbf{y}}, \prod_{i\in[\ell]} \mathsf{pk}_i^{y_i})$ is a valid key pair, due to the LKH property.

4.2 Security: Simplified Case

If the underlying PKE scheme is semantically secure and remains secure under randomness-reuse then there is a very simple way to prove security of Construction 4.1.

Specifically, we define randomness-reuse as follows:

Randomness Reuse. We say that a PKE has *randomness reuse* (RR, for short) if $\mathsf{E}(\mathsf{pk}, x; r)$ is *efficiently computed* given the triple (x, pk, r), or the triple $(x, \mathsf{sk}, \mathsf{C}(r))$ where sk is a secret key corresponding to pk. In [BBS03], this property is also called *reproducibility* and guarantees that it is secure to reuse randomness when encrypting under several independent public keys. The idea of randomness reuse was first considered by [Kur02].

Then, we are able to prove the following theorem:

Theorem 4.2 If the underlying PKE $\mathcal{E}$ is s-IND-CPA, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, and remains secure under randomness-reuse, then Construction 4.1is s-IND-FE-CPA.

Proof: This proof follows the intuition provided in the proof sketch of Theorem 3.2. To prove the security of our scheme we will show that the s-IND-FE-CPA game is indistinguishable from a game where the challenge ciphertext encrypts a random combination of the challenge messages whose coefficients sum up to one. Thus, the challenge ciphertext decrypts to the expected values and information theoretically hides the challenge bit.

Given an adversary $\mathcal{A}$ that breaks the s-IND-FE-CPA security of our IP scheme with non-negligible probability ε, we construct an adversary $\mathcal{B}$ that breaks the s-IND-CPA security of the underlying PKE scheme $\mathcal{E}$ with comparable probability.

$\mathcal{B}$ starts by picking a random element a in the full message space of the underlying PKE $\mathcal{E}$, and sends challenge messages 0 and a to the challenger $\mathcal{C}$ of PKE security game. $\mathcal{C}$ answers by sending an encryption $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ of either 0 or a and public key pk.

$\mathcal{B}$ then invokes $\mathcal{A}$ on input the security parameter and gets two different challenge messages in output, namely $(\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,\ell}))_{i\in\{0,1\}}$ both in $\mathbb{Z}_q^\ell$.

Recall that, by the constraints of security game, the adversary can only issue secret key queries for vectors $\mathbf{y}$ such that $\langle \mathbf{x}_0, \mathbf{y}\rangle = \langle \mathbf{x}_1, \mathbf{y}\rangle$. Thus, we have that $\langle \mathbf{y}, \mathbf{x}_1 - \mathbf{x}_0\rangle = 0$ meaning that $\mathbf{y}$ is in the vector space defined by $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$.

Then, $\mathcal{B}$ generates the view for $\mathcal{A}$ in the following way:

Public Key. To generate master public key mpk, $\mathcal{B}$ does the following. First, $\mathcal{B}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then we can write the canonical vectors in the basis $((\mathbf{x}_1 - \mathbf{x}_0), \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathbb{Z}_q$ and $\alpha_i \in \mathbb{Z}_q$ such that:

$$\mathbf{e}_i = \alpha_i(\mathbf{x}_1 - \mathbf{x}_0) + \sum_{j\in[\ell-1]} \lambda_{i,j}\mathbf{z}_j. \tag{1}$$

Then, for $j \in [\ell-1]$, $\mathcal{B}$ sets $(\mathsf{pk}_{\mathbf{z}_j}, \mathsf{sk}_{\mathbf{z}_j}) = \mathcal{E}.\mathsf{Setup}(1^\lambda)$, and for $i \in [\ell]$,

$$\gamma_i = \prod_{j\in[\ell-1]} \mathsf{pk}_{\mathbf{z}_j}^{\lambda_{i,j}} \quad \text{and} \quad \mathsf{pk}_i = \mathsf{pk}^{\alpha_i}\gamma_i.$$

Eventually, $\mathcal{B}$ invokes $\mathcal{A}$ on input $mpk = (\mathsf{pk}_i)_{i\in[\ell]}$.

Notice that, $\mathcal{B}$ is implicitly setting $\mathsf{sk}_i = \alpha_i\mathsf{sk} + \sum_{j\in[\ell-1]} \lambda_{i,j}\mathsf{sk}_{\mathbf{z}_j}$ because of the LKH property, where sk is the secret key corresponding to pk, which is unknown to $\mathcal{B}$.

Challenge Ciphertext. $\mathcal{B}$ computes the challenge ciphertext Ct^* as follows. $\mathcal{B}$ randomly picks $b \xleftarrow{R} \{0,1\}$, computes $\mathcal{E}.\mathsf{E}(\gamma_i, 0; r)$ from ct_0 and $\sum_{j\in[\ell-1]} \lambda_{i,j}\mathsf{sk}_{\mathbf{z}_j}$ and $\mathcal{E}.\mathsf{E}(1_H, x_{b,i}; r)$ from secret key 0_G and ct_0, by randomness reuse. $\mathcal{B}$ then sets

$$\mathsf{ct}_0^* = \mathsf{ct}_0 \quad \text{and} \quad (\mathsf{ct}_i^* = \mathsf{ct}_1^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, 0; r) \cdot \mathcal{E}.\mathsf{E}(1_H, x_{b,i}; r))_{i\in[\ell]} \ ,$$

Then the algorithm returns the challenge ciphertext $\mathsf{Ct}^* = (\mathsf{ct}_0^*, (\mathsf{ct}_i^*)_{i\in[\ell]})$.

Secret Keys. To generate a secret key for vector $\mathbf{y}$, $\mathcal{B}$ computes $\mathsf{sk}_{\mathbf{y}}$ as

$$\mathsf{sk}_{\mathbf{y}} = \sum_{j\in[\ell-1]} \left(\sum_{i\in[\ell]} y_i\lambda_{i,j} \right) \mathsf{sk}_{\mathbf{z}_j}$$

At the end of the simulation, if $\mathcal{A}$ correctly guesses b, then $\mathcal{B}$ returns 0 ($\mathcal{B}$ guesses that $\mathcal{C}$ encrypted 0), else $\mathcal{B}$ returns 1 ($\mathcal{B}$ guesses that $\mathcal{C}$ encrypted a). This concludes the description of adversary $\mathcal{B}$.

It remains to verify that $\mathcal{B}$ correctly simulates $\mathcal{A}$'s environment.

First see that the master public key is well distributed, because we are just applying a change of basis to a well distributed master public key. Now it holds that $\alpha_i = \frac{x_{1,i} - x_{0,i}}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}$ because

$$\begin{aligned} x_{1,i} - x_{0,i} &= \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{e}_i \rangle \\ &= \alpha_i \|\mathbf{x}_1 - \mathbf{x}_0\|^2 + \sum_{j\in[\ell-1]} \lambda_{i,j} \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{z}_j \rangle \\ &= \alpha_i \|\mathbf{x}_1 - \mathbf{x}_0\|^2 \ . \end{aligned}$$

Now recall that a vector $\mathbf{y}$ satisfying the security game constraints is such that $\langle \mathbf{y}, \mathbf{x}_0 \rangle = \langle \mathbf{y}, \mathbf{x}_1 \rangle$, so

$$\sum_{i \in [\ell]} y_i \alpha_i = \sum_{i \in [\ell]} y_i \frac{x_{1,i} - x_{0,i}}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2} = 0$$

which in turn implies that a secret key $\mathsf{sk}_{\mathbf{y}}$ for the vector $\mathbf{y}$ is distributed as

$$\begin{aligned} \mathsf{sk}_{\mathbf{y}} = \sum_{i \in [\ell]} y_i \mathsf{sk}_i &= \sum_{i \in [\ell]} y_i \alpha_i \mathsf{sk} + \sum_{i \in [\ell]} \sum_{j \in [\ell - 1]} y_i \lambda_{i,j} \mathsf{sk}_{\mathbf{z}_j} \\ &= \sum_{j \in [\ell - 1]} \left(\sum_{i \in [\ell]} y_i \lambda_{i,j} \right) \mathsf{sk}_{\mathbf{z}_j} \end{aligned}$$

On the other hand, if $\mathcal{A}$ asks for a secret key for some vector $\mathbf{y} \notin (\mathbf{x}_1 - \mathbf{x}_0)^{\perp}$, $\mathcal{B}$ would need to know sk in order to generate a correct secret key for $\mathbf{y}$.

Now, we have to analyze the following two cases, depending on which message was encrypted by $\mathcal{C}$ in the challenge ciphertext:

1. $\mathcal{C}$ encrypted 0. Then, the challenge ciphertext Ct^* for message $\mathbf{x}_b$ is distributed as

   $$\mathsf{ct}_0^* = \mathsf{ct}_0$$

   and

   $$\begin{aligned} \mathsf{ct}_i^* &= \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, x_{b,i}; r) \\ &= \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_{b,i}; r) \ , \end{aligned}$$

   thanks to the LCH property, and then as in the real game.

   Thus, in this case, $\mathcal{B}$ generates a view identical to that $\mathcal{A}$ would see in the real game. Hence, the advantage of $\mathcal{B}$ in this game is ε, the same advantage as $\mathcal{A}$ against s-IND-FE-CPA of IP when 0 has been encrypted.

2. $\mathcal{C}$ encrypted a. First, in Equation 1, we have $\alpha_i = (x_{1,i} - x_{0,i})/\|\mathbf{x}_1 - \mathbf{x}_0\|^2$. Let us analyze the distribution of the challenge ciphertext in this case. We have $\mathsf{ct}_0^* = \mathsf{ct}_0$ and

   $$\begin{aligned} \mathsf{ct}_i^* &= \mathcal{E}.\mathsf{E}(\mathsf{pk}, a; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, x_{b,i}; r) \\ &= \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_{b,i} + \alpha_i a; r) \\ &= \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, \hat{x}_i; r), \end{aligned}$$

   thanks to the LCH property, where $\hat{x}_i$ is defined as follows:

   $$\begin{aligned} \hat{x}_i = x_{b,i} + \alpha_i a &= \frac{a}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}(x_{1,i} - x_{0,i}) + x_{b,i} \\ &= \frac{a}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}(x_{1,i} - x_{0,i}) + x_{0,i} + b(x_{1,i} - x_{0,i}). \end{aligned}$$

   Let us set $u = a/\|\mathbf{x}_1 - \mathbf{x}_0\|^2 + b$, which is a random value in the full message space of $\mathcal{E}$, given that a is random in the same space, then $\hat{x}_i = u x_{1,i} + (1 - u) x_{0,i}$. Then, the challenge ciphertext is a valid ciphertext for the message $\hat{\mathbf{x}} = u\mathbf{x}_1 + (1 - u)\mathbf{x}_0$, which is a random linear combination of the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ whose coefficients sum up to one, as expected. Notice that b is information theoretically hidden because the distribution of u is independent from b. Hence, the advantage of $\mathcal{B}$ in this game is 0, when a random non-zero a has been encrypted.

Eventually, this shows that ε is bounded by the best advantage one can get against s-IND-FE-CPA of $\mathcal{E}$. Hence, taking the maximal values, the best advantage one can get against s-IND-FE-CPA of IP is bounded by the best advantage one can get against s-IND-FE-CPA of $\mathcal{E}$. ▮

4.3 Security: General Case

If the underlying PKE scheme does not support randomness-reuse defined in the previous section then it is still possible to prove the security of Construction 4.1 but we need the following generalized security properties.

Before stating them, for convenience we split the Setup algorithm in the following two algorithms to sample secret keys and generate the corresponding public keys.

- $\mathsf{SKGen}(1^\lambda)$ takes in input the security parameter and sample a secret key sk from the secret key space according to the same distribution induced by Setup.
- $\mathsf{PKGen}(\mathsf{sk}, \tau)$ takes in input a secret key sk and parameters τ, and generates a public key pk corresponding to sk according to the distribution induced by τ. We will omit τ when it is clear from the context.

We are ready to state the security properties that we require.

ℓ-Public-Key-Reproducibility. For a public-key encryption scheme $\mathcal{E}$ we define *ℓ-public-key-reproducibility* via the following security game:

Game $\mathsf{Exp}^{\ell\text{-pk-rep-b}}_{\mathcal{E},\lambda}(\mathcal{A})$

proc Initialize$(\lambda, \mathcal{M})$

$(\mathsf{sk}, (\alpha_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
if $b = 0$ **then**
  $(\mathsf{pk}_i = \mathcal{E}.\mathsf{PKGen}(\alpha_i \mathsf{sk} + \mathsf{sk}_i, \tau))_{i\in[\ell]}$
else
  $\mathsf{pk} \leftarrow \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}, \tau')$
  $(\mathsf{pk}_i = \mathsf{pk}^{\alpha_i} \cdot \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}_i, \tau_i))_{i\in[\ell]}$
Return $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}$

proc Finalize(b')

Return $(b' = b)$

with $\mathcal{M}$ samples tuples of the form $(\mathsf{sk}, (\alpha_i, \mathsf{sk}_i)_{i\in[\ell]})$ where sk and the sk_i's are in G, and the α_i's are in $\mathbb{Z}_q$.

Then, we say that $\mathcal{E}$ has ℓ-public-key-reproducibility if there exists $\tau, \tau', (\tau_i)_{i\in[\ell]}$ such that

$$\left|\Pr[\mathsf{Exp}^{\ell\text{-pk-rep-0}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^{\ell\text{-pk-rep-1}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

ℓ-Ciphertext-Reproducibility. For a public-key encryption scheme $\mathcal{E}$ we define *ℓ-ciphertext-reproducibility* via the following security game:

Game $\mathsf{Exp}^{\ell\text{-ct-rep-b}}_{\mathcal{E},\lambda}(\mathcal{A})$

proc Initialize$(\lambda, \mathcal{M})$

$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\mathsf{sk} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda), \mathsf{pk} \leftarrow \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}, \tau')$
$(\mathsf{pk}_i \leftarrow \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}_i, \tau_i))_{i\in[\ell]}$
$\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r), \mathsf{ct} = \mathcal{E}.\mathsf{E}(\mathsf{pk}, a; r)$
if $b = 0$ **then**
  $\mathsf{ct}_i = \mathsf{ct}^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)$
else
  $\mathsf{ct}_i = \mathsf{ct}^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}'(\mathsf{sk}_i, x_i, \mathsf{ct}_0)$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

proc Finalize(b')

Return $(b' = b)$

where

- $\mathcal{M}$ samples tuples of the form $(a, (\alpha_i, x_i, , \mathsf{sk}_i)_{i\in[\ell]})$, where sk_i's are in G, and a and the α_i's, x_i's are in $\mathbb{Z}_q$.
- E' is an algorithm that takes in input a secret key in H, a message in $\mathbb{Z}_q$, a first part ciphertext $\mathsf{C}(r)$ for some r in the randomness space, and output a second part ciphertext.

Then, we say that $\mathcal{E}$ has ℓ-ciphertext-reproducibility if there exists τ', τ_i's and algorithm E' such that

$$\left|\Pr[\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{FE},\lambda}(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Then, we are able to prove the following theorem:

Theorem 4.3 If the underlying PKE $\mathcal{E}$ is s-IND-CPA, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, ℓ-public-key-reproducible and ℓ-ciphertext-reproducible, then Construction 4.1 is s-IND-FE-CPA.

Proof: The proof strategy is essential that of Theorem 4.2. We prove security via a sequence of hybrid experiments.

Hybrid H_1: This is the s-IND-FE-CPA game.

Hybrid H_2: This is like H_1 except that the master public key is generated by invoking the algorithm $H_2.\mathsf{Setup}$ defined as follows:

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$	**proc LR**()
$(mpk, msk) \xleftarrow{R} \mathsf{Setup}(1^\lambda)$	$\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
$V \leftarrow \emptyset$	Return Ct^*
Return mpk	**proc Finalize**(b')
proc KeyDer$(\mathbf{y})$	**if** $\exists \mathbf{y} \in V$ such that
$V \leftarrow V \cup \{\mathbf{y}\}$	$F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$
$\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$	**then return** false
Return $\mathsf{sk}_\mathbf{y}$	Return $(b' = b)$

$H_2.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)$: The algorithm finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathbb{Z}_q$ and $\alpha_i \in \mathbb{Z}_q$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j .$$

Then, the algorithms samples $\mathsf{sk} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$ and, for $j \in [\ell]$, PKE secret key $\mathsf{sk}_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$. Finally, the algorithm sets:

$$\mathsf{sk}_i = \alpha_i \cdot \mathsf{sk} + \sum \lambda_{i,j} \mathsf{sk}_{\mathbf{z}_j}, \quad \mathsf{pk}_i = \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}_i, \tau) \ ,$$

where τ is the same used in the Setup algorithm. The algorithm returns $mpk = (\mathsf{pk}_i)_{i \in [\ell]}$ and $msk = (\mathsf{sk}_i)$.

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$	**proc LR**()
$\boxed{(mpk, msk) \xleftarrow{R} H_2.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)}$	$\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
$V \leftarrow \emptyset$	Return Ct^*
Return mpk	**proc Finalize**(b')
proc KeyDer$(\mathbf{y})$	**if** $\exists \mathbf{y} \in V$ such that
$V \leftarrow V \cup \{\mathbf{y}\}$	$F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$
$\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$	**then return** false
Return $\mathsf{sk}_\mathbf{y}$	Return $(b' = b)$

Hybrid H_3: This is like H_2 except that the master public key is generated by invoking the algorithm $H_3.\mathsf{Setup}$ and the secret keys are generated by invoking the algorithm $H_3.\mathsf{KeyDer}$ which are defined as follows.

$H_3.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)$: The algorithm finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathbb{Z}_q$ and $\alpha_i \in \mathbb{Z}_q$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j.$$

Then, the algorithms first samples $\mathsf{sk} \xleftarrow{R} \mathcal{E}.\mathsf{SKGen}(1^\lambda)$ and $\mathsf{pk} \xleftarrow{R} \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}, \tau')$. Then, for $j \in [\ell-1]$, a PKE secret keys $\mathsf{sk}_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$, sets $t_i = \sum \lambda_{i,j} \mathsf{sk}_{\mathbf{z}_j}$, $\mathsf{pk}_{t_i} \leftarrow \mathcal{E}.\mathsf{PKGen}(t_i, \tau_i)$ and computes

$$\mathsf{pk}_i = \mathsf{pk}^{\alpha_i} \cdot \mathsf{pk}_{t_i}.$$

The algorithm returns $mpk = (\mathsf{pk}_i)_{i \in [\ell]}$ and $msk' = (\mathsf{pk}, \mathsf{sk}_{\mathbf{z}_j})$.

$H_3.\mathsf{KeyDer}(msk', \mathbf{y})$: The algorithm computes secret key for vector $\mathbf{y}$ in the following way: $\mathsf{Sk}_{\mathbf{y}} = \sum_{i \in [\ell]} y_i \cdot t_i$.

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$

$\boxed{(mpk, msk') \xleftarrow{R} H_3.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)}$

$V \leftarrow \emptyset$

Return mpk

proc KeyDer$(\mathbf{y})$

$V \leftarrow V \cup \{\mathbf{y}\}$

$\boxed{\mathsf{sk}_{\mathbf{y}} \xleftarrow{R} H_3.\mathsf{KeyDer}(msk', k)}$

Return $\mathsf{sk}_{\mathbf{y}}$

proc LR()

$\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$

Return Ct^*

proc Finalize(b')

if $\exists \mathbf{y} \in V$ such that
$F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$
then return false

Return $(b' = b)$

Hybrid H_4: This is like H_3 except that the challenge ciphertext is generated by invoking the algorithm $H_4.\mathsf{Encrypt}$ defined as follows:

$H_4.\mathsf{Encrypt}(msk', \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r) \quad \text{and} \quad \left(\mathsf{ct}_i = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_{t_i}, x_i; r)\right)_{i \in [\ell]},$$

where r is some randomness in the random space of $\mathcal{E}$.

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{R} H_3.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $\mathsf{sk}_\mathbf{y} \xleftarrow{R} H_3.\mathsf{KeyDer}(msk', k)$ Return $\mathsf{sk}_\mathbf{y}$	**proc LR**() $\boxed{\mathsf{Ct}^* \xleftarrow{R} H_4.\mathsf{Encrypt}(msk', \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$ **then return** false Return $(b' = b)$

Hybrid H_5: This is like H_4 except that the challenge ciphertext is generated by invoking the algorithm $H_5.\mathsf{Encrypt}$ defined as follows:

$H_5.\mathsf{Encrypt}(msk', \mathsf{Ct}, \mathbf{x})$: Let $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$, then, the algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$\mathsf{ct}'_0 = \mathsf{ct}_0 \quad \text{and} \quad \left(\mathsf{ct}'_i = \mathsf{ct}_1^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}'(t_i, x_i, \mathsf{ct}_0; \tilde{r})\right)_{i \in [\ell]} ,$$

where $\tilde{r}$ is some randomness shared among all the invocation of $\mathcal{E}.\mathsf{E}$.

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{R} H_3.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $\mathsf{sk}_\mathbf{y} \xleftarrow{R} H_3.\mathsf{KeyDer}(msk', k)$ Return $\mathsf{sk}_\mathbf{y}$	**proc LR**() $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, 0)}$ $\boxed{\mathsf{Ct}^* \xleftarrow{R} H_5.\mathsf{Encrypt}(msk', \mathsf{Ct}, \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$ **then return** false Return $(b' = b)$

H_5 is indistinguishable from H_4 due to the ℓ-ciphertext-reproducability.

Hybrid H_6: This is like H_5 except that Ct encrypts a random value $a \in \mathbb{Z}_p$.

proc Initialize$(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{R} H_3.\mathsf{Setup}(1^\lambda, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $\mathsf{sk}_\mathbf{y} \xleftarrow{R} H_3.\mathsf{KeyDer}(msk', k)$ Return $\mathsf{sk}_\mathbf{y}$	**proc LR**() $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, a),\ a \leftarrow \mathbb{Z}_p}$ $\mathsf{Ct}^* \xleftarrow{R} H_5.\mathsf{Encrypt}(msk', \mathsf{Ct}, \mathbf{x}_b)$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$ **then return** false Return $(b' = b)$

We now show that the relevant distinguishing probabilities between adjacent hybrids are negligible, which completes the proof.

Indistinguishability of H_1 and H_2: The distribution of the master public key in both the games is identically. This is because in H_2 a simple change of basis is applied to a well distributed master secret key and this change of basis can be computed due to the linear key-homomorphism of $\mathcal{E}$, that tells us that the sk_i's as computed in H_2 are valid secret keys of $\mathcal{E}$.

Moreover, notice that, by our change of basis it holds that $\alpha_i = x_{1,i} - x_{0,i}$ because

$$\begin{aligned} x_{1,i} - x_{0,i} &= \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{e}_i \rangle \\ &= \alpha_i + \sum_{j \in [\ell-1]} \lambda_{i,j} \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{z}_j \rangle \\ &= \alpha_i \ . \end{aligned}$$

Then, the change of basis implies that for all the vectors $\mathbf{y} = (y_1, \ldots, y_\ell)$ satisfying the security game constraints, meaning that $\mathbf{y} \in (\mathbf{x}_1 - \mathbf{x}_0)^\perp$, it holds that $\sum_{i \in [\ell]} y_i \cdot \alpha_i = 0$. Thus, to generate a well-distributed secret key for a $\mathbf{y}$ satisfying the security game constraints, sk is not required.

Indistinguishability of H_2 and H_3: This holds under the ℓ-public-key-reproducibility of $\mathcal{E}$ for the distribution $\mathcal{M}$, induced by the challenge messages, defined as follows:

$\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}(1^\lambda)$: $\mathcal{M}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathbb{Z}_q$ and $\alpha_i \in \mathbb{Z}_q$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j .$$

Then, $\mathcal{M}$ first samples $\mathsf{sk} \xleftarrow{R} \mathcal{E}.\mathsf{SKGen}(1^\lambda)$, then, for $j \in [\ell-1]$, samples $\mathsf{sk}_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$, sets $t_i = \sum \lambda_{i,j} \mathsf{sk}_{\mathbf{z}_j}$ and gives in output $(\mathsf{sk}, (\alpha_i, t_i)_{i \in [\ell]})$.

For the sake of completeness, suppose that there exists and adversary $\mathcal{A}$ that distinguishes with non-negligible advantage H_2 from H_3. Then, we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and breaks the ℓ-public-key-reproducibility of $\mathcal{E}$.

$\mathcal{B}$ does the following. $\mathcal{B}$ invokes $\mathcal{A}$ to gets the challenge messages $\mathbf{x}_0, \mathbf{x}_1$. Then, $\mathcal{B}$ receives from the challenger of the ℓ-public-key-reproducibility values $(\mathsf{pk}_i, t_i)_{i \in [\ell]}$ on input the security parameter and distribution $\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}$. $\mathcal{B}$ sets master public key as $mpk = (\mathsf{pk}_i)_{i \in [\ell]}$ and gives it to $\mathcal{A}$. Then $\mathcal{B}$ answers secret key queries by using the t_i's which are enough to generate secret keys for vectors satisfying the security game constraints. $\mathcal{B}$ generates the challenge ciphertext by using mpk and uses $\mathcal{A}$'s guess as its guess for the ℓ-public-key-reproducibility game.

Finally, notice that if $\mathcal{B}$ plays $\mathsf{Exp}^{\ell\text{-pk-rep-0}}_{\mathcal{E},\lambda}(\mathcal{B})$ (resp. $\mathsf{Exp}^{\ell\text{-pk-rep-1}}_{\mathcal{E},\lambda}(\mathcal{B})$), then $\mathcal{B}$ perfectly simulates H_2 (resp. H_3).

Indistinguishability of H_3 and H_4: By linear ciphertext-homomorphism of $\mathcal{E}$, $H_3 = H_4$. In fact, notice that in both the games $\mathsf{pk}_i = \mathsf{pk}^\alpha \cdot \mathsf{pk}_{t_i}$, then it holds that

$$\mathcal{E}.\mathsf{E}(\mathsf{pk}^{\alpha_i} \cdot \mathsf{pk}_{t_i}, x_i; r) = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_{t_i}, x_i; r) \ .$$

Indistinguishability of H_4 and H_5: This holds under the ℓ-ciphertext-reproducibility of $\mathcal{E}$ for the distribution $\mathcal{M}$, induced by the challenge messages, defined as follows:

$\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}(1^\lambda)$: $\mathcal{M}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathbb{Z}_q$ and $\alpha_i \in \mathbb{Z}_q$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j.$$

Then, $\mathcal{M}$ samples, for $j \in [\ell-1]$, secret keys $\mathsf{sk}_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$.

Finally, $\mathcal{M}$, for $i \in [\ell]$, sets $t_i = \sum \lambda_{i,j} \mathsf{sk}_{\mathbf{z}_j}$ and returns $(0, (\alpha_i, x_{\beta,i}, \mathsf{sk}_{t_i})_{i\in[\ell]})$, where β is random bit.

For the sake of completeness, suppose that there exists and adversary $\mathcal{A}$ that distinguishes with non-negligible advantage H_4 from H_5. Then, we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and breaks the ℓ-ciphertext-reproducibility of $\mathcal{E}$.

$\mathcal{B}$ does the following. $\mathcal{B}$ invokes $\mathcal{A}$ to gets the challenge messages $\mathbf{x}_0, \mathbf{x}_1$. Then, $\mathcal{B}$ receives from the challenger of the ℓ-ciphertext-reproducibility values $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_{t_i}, t_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$ on input the security parameter and distribution $\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}$. $\mathcal{B}$ sets master public key as $mpk = (\mathsf{pk}_i = \mathsf{pk}^{\alpha_i} \cdot \mathsf{pk}_{t_i})_{i\in[\ell]}$ and gives it to $\mathcal{A}$. Then $\mathcal{B}$ answers secret key queries by using the t_i's which are enough to generate secret keys for vectors satisfying the security game constraints. $\mathcal{B}$ uses as challenge ciphertext $(\mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$ and uses $\mathcal{A}$'s guess as its guess for the ℓ-ciphertext-reproducibility game.

Finally, notice that if $\mathcal{B}$ plays $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\lambda}(\mathcal{B})$ (resp. $\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{E},\lambda}(\mathcal{B})$), then $\mathcal{B}$ perfectly simulates H_4 (resp. H_5).

Indistinguishability of H_5 and H_6: The only difference between H_5 and H_6 is in the message that Ct encrypts. In H_5, Ct is an encryption of 0 and H_5, it is an encryption of a for a random $a \in \mathbb{Z}_q$. Moreover, notice that sk is never used. Therefore under the the IND-CPA security of $\mathcal{E}$, H_5 is computational indistinguishable from H_6.

Advantage of any adversary in H_6. In this game, the challenge ciphertext is for message $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_\ell)$ defined as follows:

$$\begin{aligned}\hat{x}_i = x_{b,i} + \alpha_i a &= a(x_{1,i} - x_{0,i}) + x_{b,i} \\ &= a(x_{1,i} - x_{0,i}) + x_{0,i} + b(x_{1,i} - x_{0,i}).\end{aligned}$$

Let us set $u = a + b$, which is a random value in $\mathbb{Z}_p$, then $\hat{x}_i = u x_{1,i} + (1-u)x_{0,i}$. Then, the challenge ciphertext is a valid ciphertext for the message $\hat{\mathbf{x}} = u\mathbf{x}_1 + (1-u)\mathbf{x}_0$, which is a random linear combination of the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ whose coefficients sum up to one. Notice that b is information theoretically hidden because the distribution of u is independent from b. Hence, the advantage of any adversary in this game is 0.

∎

5 Instantiation from DDH

The scheme in Section 3 can be obtained by plugging into our generalization the ElGamal encryption scheme [ElG85] which supports the properties that we require. Namely:

Structure. Let p be a prime and $\mathbb{G}$ a group of order q where the DDH assumption is supposed to be hard, and g a generator of $\mathbb{G}$. Then, ElGamal's secret key space is the group $(\mathbb{Z}_q, +, 0)$, public key space is the group $(\mathbb{G}, \times, 1)$, and messages are elements of $\mathbb{Z}_q$. An ElGamal ciphertext is of the form $\mathsf{Ct} = (\mathsf{ct}_0 = g^r, \mathsf{ct}_1 = \mathsf{pk}^r \cdot g^x)$ as required. Thus, $\mathsf{C}(r) = g^r$ and $\mathsf{E}(\mathsf{pk}, x; r) = \mathsf{pk}^r \cdot g^x)$.

Linear Key Homomorphism. It is easy to see that for any two secret keys $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathbb{Z}_q$ and any $y_1, y_2 \in \mathbb{Z}_q$, the component-wise linear combination formed by $y_1\mathsf{sk}_1 + y_2\mathsf{sk}_2$ can be computed efficiently only using public parameters, the secret keys sk_1 and sk_2 and the coefficients y_1 and y_2. And this combination $y_1\mathsf{sk}_1 + y_2\mathsf{sk}_2$ also functions as a secret key to a public key that can be computed as $\mathsf{pk}_1^{y_1} \cdot \mathsf{pk}_2^{y_2} = g^{y_1\mathsf{sk}_1 + y_2\mathsf{sk}_2}$, where pk_1 (resp. pk_2) is a public key corresponding to sk_1 (resp. sk_2).

Linear Ciphertext Homomorphism Under Shared Randomness. It holds that

$$\begin{aligned}\mathsf{E}(\mathsf{pk}_1, x_1; r) \cdot \mathsf{E}(\mathsf{pk}_2, x_2; r) &= g^{r\cdot\mathsf{sk}_1} g^{x_1} \cdot g^{r\cdot\mathsf{sk}_2} g^{x_2} \\ &= g^{r\cdot(\mathsf{sk}_1+\mathsf{sk}_2)} \cdot g^{x_1+x_2} \\ &= \mathsf{E}(\mathsf{pk}_1\mathsf{pk}_2, x_1 + x_2; r) \ .\end{aligned}$$

Randomness Reuse It holds that

$$\begin{aligned}\mathsf{E}(\mathsf{pk}, x; r) &= (g^{\mathsf{sk}})^r \cdot g^x \\ &= (g^r)^{\mathsf{sk}} \cdot g^x\end{aligned}$$

Finally, notice that because ElGamal is secure under randomness reuse the simplified proof of security applies to the IP scheme obtained from it.

6 Instantiation from LWE

6.1 Tools and Assumptions

Gaussians and Lattices. The n-dimensional Gaussian function $\rho : \mathcal{R}^n \to (0, 1]$ is defined as

$$\rho(\mathbf{x}) = \exp(-\pi \cdot ||\mathbf{x}||^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle) \ .$$

Applying a linear transformation given by a (not necessarily square) matrix B with linearly independent columns yields the (possibly degenerate) Gaussian function

$$\rho_{\mathbf{B}}(\mathbf{x}) = \begin{cases} \rho(\mathbf{B}^+\mathbf{x}) = \exp(-\pi \cdot \mathbf{x}^\top \Sigma^+ \mathbf{x}) & \text{If } \mathbf{x} \in \text{span}(\mathbf{B}) = \text{span}(\Sigma) \\ 0 & \text{otherwise} \end{cases}$$

where $\Sigma = \mathbf{B}\mathbf{B}^\top \leq 0$. Because $\rho_{\mathbf{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

Lemma 6.1 Let $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ be the probability that a gaussian random variables of covariance matrix Σ is equal to $\mathbf{x}$. Then, for any invertible matrix β,

$$\rho_{\sqrt{\Sigma}}(\beta^{-1}\mathbf{x}) = \rho_{\beta\sqrt{\Sigma}}(\mathbf{x}) \ .$$

Lemma 6.2 Let $\Lambda \subset \mathbb{R}^n$ be a lattice. For any $\Sigma \geq \mathbf{0}$ and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_{\sqrt{\Sigma}}(\Lambda+\mathbf{c}) \leq \rho_{\sqrt{\Sigma}}(\Lambda)$. Moreover, if $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ for some $\epsilon > 0$ and $\mathbf{c} \in \mathrm{span}(\Lambda)$, then $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \geq \frac{1-\epsilon}{1+\epsilon} \cdot \rho_{\sqrt{\Sigma}}(\Lambda)$.

The LWE Assumption. The learning with errors (LWE) problem was introduced by Regev [Reg05]. Let n, q be integer parameters. For any noise distribution χ on $\mathbb{Z}_q$, and vector $\mathbf{s} \in \mathbb{Z}_q^n$, the oracle $\mathsf{LWE}_{q,n,\chi}(\mathbf{s})$ samples a fresh random n-dimensional vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, as well as noise $e \leftarrow \chi$, and returns $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$. The LWE assumption with noise χ states that for every PPT distinguisher $\mathcal{D}$,

$$\Pr[\mathbf{s} \leftarrow \mathbb{Z}_q^n \ : \ \mathcal{D}^{\mathsf{LWE}_{q,n,\chi}(\mathbf{s})} = 1] - \Pr[\mathbf{s} \leftarrow \mathbb{Z}_q^n \ : \ \mathcal{D}^{\mathsf{LWE}_{q,n,U}(\mathbf{s})} = 1] = \mathsf{negl}(n),$$

where U is the uniform distribution on $\mathbb{Z}_q$.

Generalized Leftover Hash Lemma. The following text and lemma are taken verbatim from [DRS04]. The predictability of a random variable A is $\max_a \Pr[A = a]$, and its min-entropy $\mathbf{H}_\infty(A)$ is $-\log(\max_a \Pr[A = a])$. The min-entropy of a distribution tells us how many nearly uniform random bits can be extracted from it. The notion of *nearly* is defined as follows. The statistical distance between two probability distributions A and B is $\mathbf{SD}(A, B) = \frac{1}{2} \sum_v |\Pr[A = v] - \Pr[B = v]|$.

Consider now a pair of (possibly correlated) random variables A, B. If the adversary finds out the value b of B, then predictability of A becomes $\max_a \Pr[A = a|B = b]$. On average, the adversary's chance of success in predicting A is then $\mathbb{E}_{b \leftarrow B}[\max_a \Pr[A = a|B = b]]$. Note that we are taking the average over B (which is not under adversarial control), but the worst case over A (because prediction of A is adversarial once b is known). Again, it is convenient to talk about security in log-scale, which is why we define the average min-entropy of A given B as simply the logarithm of the above:

$$\tilde{\mathbf{H}}_\infty(A|B) = -\log\left(\mathbb{E}_{b \leftarrow B}[\max_a \Pr[A = a|B = b]]\right) = -\log\left(\mathbb{E}_{b \leftarrow B}\left[2^{\mathbf{H}_\infty(A|B=b)}\right]\right) \ .$$

Lemma 6.3 Let A, B, C be random variables. Then If B has at most 2^λ possible values, then

$$\tilde{\mathbf{H}}_\infty(A|(B, C)) \geq \tilde{\mathbf{H}}_\infty((A, B)|C) - \lambda \geq \tilde{\mathbf{H}}_\infty(A|C) - \lambda \ ,$$

In particular, $\tilde{\mathbf{H}}_\infty(A|B) \geq \mathbf{H}_\infty((A, B)) - \lambda \geq \mathbf{H}_\infty(A) - \lambda$.

Lemma 6.4 [Generalized Leftover Hash Lemma [DRS04]] Assume $\{H_x : \{0,1\}^n \to \{0,1\}^\ell\}_{x \in X}$ is a family of universal hash functions. Then, for any random variables W and I,

$$\mathbf{SD}((H_X(W), X, I), (U_\ell, X, I)) \leq \frac{1}{2}\sqrt{2^{-\tilde{H}_\infty(W|I)} 2^\ell} \ .$$

6.2 Regev PKE Scheme

The message space is $\{0, \ldots, M\}$ for some integer M. Let $p > M$ be a prime modulus, and $q = \mathsf{poly}(n) > p$ be another prime modulus. Our public keys and ciphertexts consist of matrices and vectors over $\mathbb{Z}_q$. For every $v \in \mathbb{Z}_p$ (i.e., one entry of a message vector), define the "center" for v as $t(v) = \left\lfloor v \cdot \frac{q}{p} \right\rfloor \in \mathbb{Z}_q$. Let χ_σ denote an integer gaussian distribution over $\mathbb{Z}_q$ with standard deviation σ: $\chi_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(\mathbb{Z})}$. Let $m = m(n)$, and $\sigma = \sigma(n)$ and $\sigma' = \sigma'(n)$ be positive real Gaussian parameters. The following relations between parameters are required for correctness and security:

- $m \geq (1+\epsilon)(\ell + n + 1)\log q$
- q is a prime number of the size of $pn^2M\sqrt{\ell}$
- $\sigma \geq (\ell M + 1)\sigma$ the standard deviation of the errors in the scheme
- $\sigma' = o(\frac{1}{pM^2\ell\sqrt{m\log\lambda}})$ the standard deviation of the errors in the proof of security

All operations are performed over $\mathbb{Z}_q$.

Construction 6.5 We define our *public-key encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{SKGen}, \mathsf{PKGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows.

- $\mathsf{Gen}(1^\lambda)$ generates common parameters. Specifically, the algorithm samples $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, and returns $\mathsf{params} = (\mathbf{A}, \chi_\sigma)$. We always assume that all the other algorithms take in input params, thus we avoid to include explicitly.
- $\mathsf{SKGen}(1^\lambda)$ takes in input the security parameter and samples $\mathbf{s}$ from $\mathbb{Z}_q^n$ and returns $\mathsf{sk} = \mathbf{s}$.
- $\mathsf{PKGen}(\mathsf{sk}, \tau)$ takes in input secret key $\mathbf{s}$, parameters τ, and samples $\mathbf{e}$ from χ_σ^m and computes $\mathsf{pk} = \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$. Then the algorithm returns pk.

  Notice that, if τ describes an error distribution then $\mathbf{e}$ is sampled from this latter distribution.
- $\mathsf{Encrypt}(\mathsf{pk}, x)$ on input public key pk, and message $x \in \mathbb{Z}_p$, chooses random $\mathbf{r} \leftarrow \{0,1\}^m$ and computes

  $$\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r} \in \mathbb{Z}_q^n$$

  and

  $$\mathsf{ct}_1 = \langle \mathsf{pk}, \mathbf{r} \rangle + t(x) \in \mathbb{Z}_q \ .$$

  Then the algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{Ct}, \mathsf{sk})$ on input public key pk, ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ and secret key sk, Compute $d = \mathsf{ct}_1 - \langle \mathsf{ct}_0, \mathbf{s} \rangle$ and output the plaintext $x \in \mathbb{Z}_p$, where each x is such that $d - t(x) \in \mathbb{Z}_q$ is closest to 0 $\mod q$.

Linear Key Homomorphism. The first property comes from the fact that secret keys are elements uniformly sampled from the group $\mathbb{Z}_q^n$, so the secret key space is stable under addition. Moreover, $\sum \alpha_i \mathbf{s}_i$ is a correct secret key for $\sum \alpha_i(\mathbf{A}\mathbf{s}_i + \mathbf{e}_i)$ as long as $\sum \alpha_i \mathbf{e}_i$ remains small, which is true for small values of α_i.

Ciphertext Homomorphism Under Shared Randomness. It is easy to verify that:

$$\langle \mathsf{pk}_1, \mathbf{r} \rangle + t(x_1) + \langle \mathsf{pk}_2, \mathbf{r} \rangle + t(x_2) = \langle \mathsf{pk}_1 + \mathsf{pk}_2, \mathbf{r} \rangle + t(x_1 + x_2) \ .$$

By definition of t.

ℓ-Public-Key Reproducibility. To show that Construction 6.5 has *ℓ-public-key-reproducibility*, for any fixed constant ℓ, it is sufficient to show that there are error distributions with standard deviations $\sigma, \sigma', (\sigma_i)_{i\in[\ell]}$ such that $\mathsf{Exp}^{\ell\text{-pk-rep-0}}_{\mathcal{E},\lambda}(\mathcal{A})$ is indistinguishable from $\mathsf{Exp}^{\ell\text{-pk-rep-1}}_{\mathcal{E},\lambda}(\mathcal{A})$.

Theorem 6.6 Construction 6.5 has *ℓ-public-key-reproducibility* in a statistical sense.

Proof: Let τ be the description of an error distribution with standard deviation σ, and τ' an error distribution with standard deviation σ'.

Let $\{\tau_i\}_{i\in[\ell]}$ describe sampling ℓ errors with covariance matrix $\sqrt{\Sigma}$, where $\Sigma = \sigma^2 I_\ell - \sigma'^2 \vec{\alpha}\vec{\alpha}^\top$, where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_\ell)$. Notice that Σ is positive semi-definite if $\sigma > \sigma'(1 + M\sqrt{\ell})$ because α_i is smaller than M for any i.

Finally, let $\beta = \begin{pmatrix} I_\ell & \vec{\alpha} \end{pmatrix}$ and $\Sigma' = \begin{pmatrix} \Sigma & 0 \\ 0 & \sigma'^2 \end{pmatrix}$, where I_ℓ is the identity matrix.

Then, If $b = 0$, the errors appearing in pk_i come from the distribution $\chi_{\sigma I_\ell}$. If $b = 1$, the errors appearing in pk_i come from the distribution $\beta\chi_{\sqrt{\Sigma'}}$.

We show that these two distribution are statistically close if $\sigma > \sigma'(1 + M\sqrt{\ell})$. Let us set $\beta' = \begin{pmatrix} I_\ell & \vec{\alpha} \\ \vec{\mu}^\top & 1 + \vec{\mu}^\top\vec{\alpha} \end{pmatrix}$ for some $\vec{\mu}$. Let $\Sigma_0 = \beta'\sqrt{\Sigma'}(\beta'\sqrt{\Sigma'})^\top$ be of the target form $\begin{pmatrix} \sigma^2 I_\ell & 0 \\ 0 & {\gamma_0}^2 \end{pmatrix}$. If Σ_0 has this form, it means that $\beta'\sqrt{\Sigma'}$ gives us ℓ uncorrelated errors distributed as χ_σ and another error distributed as χ_{γ_0} which we can drop because it is not correlated to the other. Then $\forall \mathbf{z}, \epsilon \leftarrow \chi_{\sqrt{\Sigma'}}$

$$
\begin{aligned}
\mathbf{Pr}(\beta\epsilon = \mathbf{z}) &= \sum_s \mathbf{Pr}\left(\beta'\epsilon = \begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\top\mathbf{z} + s \end{pmatrix}\right) \\
&= \sum_s \mathbf{Pr}\left(\epsilon = \beta'^{-1}\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\top\mathbf{z} + s \end{pmatrix}\right) \\
&\propto \sum_s \rho_{\sqrt{\Sigma'}}\left(\beta'^{-1}\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\top\mathbf{z} + s \end{pmatrix}\right) \\
&\propto \sum_s \rho_{\beta'\sqrt{\Sigma'}}\left(\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\top\mathbf{z} + s \end{pmatrix}\right) \text{ by Lemma 6.1} \\
&\propto \sum_s \rho_{\sqrt{\Sigma_0}}(\mathbf{z})\rho_{\gamma_0}(\vec{\mu}^\top\mathbf{z} + s) \\
&\propto \rho_{\sqrt{\Sigma_0}}(\mathbf{z})\rho_{\gamma_0}(\vec{\mu}^\top\mathbf{z} + \mathbb{Z}) \\
&\propto \nu\rho_{\sqrt{\Sigma_0}}(\mathbf{z}) \text{ where } \nu \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \text{ as long as } \gamma_0 > 2 \text{ by Lemma 6.2}
\end{aligned}
$$

∎

ℓ-Ciphertext Reproducibility. We show that Construction 6.5 has *ℓ-ciphertext-reproducibility*, for any fixed constant ℓ, by taking error distributions with standard deviations $\sigma', (\sigma_i)_{i\in[\ell]}$ as chosen for the ℓ-public-key-reproducibility, and by the following alternative encryption algorithm

$$\mathsf{E}'((\mathbf{s}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r}') = \langle \mathbf{s}_i, \mathsf{ct}_0\rangle + \langle \mathbf{e}_i, \mathbf{r}'\rangle + t(x_i) \ .$$

as required. This is enough to show that $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\lambda}(\mathcal{A})$ is indistinguishable from $\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{E},\lambda}(\mathcal{A})$.

Theorem 6.7 Under the LWE assumption, Construction 6.5 has ℓ-*ciphertext-reproducibility.*

Proof: We prove the theorem via a sequence of hybrid experiments.

Hybrid H_1: This is the $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\mathcal{M},\lambda}(\mathcal{A})$, with the algorithms unfold.

$\underline{\textbf{proc Initialize}}(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\mathsf{sk} \xleftarrow{R} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi_{\sigma'}^m, \mathsf{pk} = \mathbf{A}\mathsf{sk} + \mathbf{e}$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{R} \{0,1\}^m$
$\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}, \mathsf{ct} = \mathsf{E}(\mathsf{pk}, a; \mathbf{r})$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \mathsf{E}(\mathsf{pk}_i, x_i; \mathbf{r})$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

$\underline{\textbf{proc Finalize}}(b')$
Return $(b' = b)$

Hybrid H_2: This is like H_1 except that pk is taken uniformly random in $\mathbb{Z}_q^m$.

$\underline{\textbf{proc Initialize}}(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\boxed{\mathsf{pk} \xleftarrow{R} \mathbb{Z}_q^m}$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{R} \{0,1\}^m$
$\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}, \mathsf{ct} = \mathsf{E}(\mathsf{pk}, a; \mathbf{r})$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \mathsf{E}(\mathsf{pk}_i, x_i; \mathbf{r})$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

$\underline{\textbf{proc Finalize}}(b')$
Return $(b' = b)$

Hybrid H_3: This is like H_2 except that ct_0 and $\langle \mathsf{pk}, \mathbf{r} \rangle$ are replaced with uniformly random values.

$\underline{\textbf{proc Initialize}}(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\mathsf{pk} \xleftarrow{R} \mathbb{Z}_q^m, \boxed{u \xleftarrow{R} \mathbb{Z}_q}$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{R} \{0,1\}^m$
$\boxed{\mathsf{ct}_0 \xleftarrow{R} \mathbb{Z}_q^m, \mathsf{ct} = u + t(a)}$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \boxed{\mathsf{E}'((\mathsf{sk}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r})}$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

$\underline{\textbf{proc Finalize}}(b')$
Return $(b' = b)$

Notice that $\mathsf{E}'((\mathsf{sk}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r}) = \mathsf{E}(\mathsf{pk}_i, x_i; \mathbf{r})$ if $\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}$.

Hybrid H_4: This is like H_3 except that $\mathbf{r}$ is replaced by another random value $\mathbf{r}'$.

proc Initialize$(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\mathsf{pk} \xleftarrow{R} \mathbb{Z}_q^m, u \xleftarrow{R} \mathbb{Z}_q$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\boxed{\mathbf{r}' \xleftarrow{R} \{0,1\}^m}$
$\mathsf{ct}_0 \xleftarrow{R} \mathbb{Z}_q^m, \mathsf{ct} = u + t(a)$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \boxed{\mathsf{E}'((\mathsf{sk}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r}')}$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

Hybrid H_5: This is like H_4 except that ct_0 is generated as $\mathbf{A}^\top \mathbf{r}$ and u is replaced by $\langle \mathsf{pk}, \mathbf{r} \rangle$.

proc Initialize$(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\boxed{\mathsf{pk} \xleftarrow{R} \mathbb{Z}_q^m}$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\boxed{\mathbf{r} \xleftarrow{R} \{0,1\}^m, \mathbf{r}' \xleftarrow{R} \{0,1\}^m}$
$\boxed{\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}, \mathsf{ct} = \mathsf{E}(\mathsf{pk}, a; \mathbf{r})}$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \mathsf{E}'((\mathsf{sk}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r}')$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

Hybrid H_6: This is the $\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{E},\mathcal{M},\lambda}(\mathcal{A})$.

proc Initialize$(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{M}(1^\lambda)$
$\boxed{\mathsf{sk} \xleftarrow{R} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi_{\sigma'}^m, \mathsf{pk} = \mathbf{A}\mathsf{sk} + \mathbf{e}}$
$\mathsf{pk}_i = \mathbf{A}\mathsf{sk}_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{R} \{0,1\}^m, \mathbf{r}' \xleftarrow{R} \{0,1\}^m$
$\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}, \mathsf{ct} = \mathsf{E}(\mathsf{pk}, a; \mathbf{r})$
$\mathsf{ct}_i = \alpha_i \mathsf{ct} + \mathsf{E}'((\mathsf{sk}_i, \mathbf{e}_i), x_i, \mathsf{ct}_0; \mathbf{r}')$
Return $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

We now show that the relevant distinguishing probabilities between adjacent hybrids are negligible, which completes the proof.

Indistinguishability of H_1 and H_2: The hardness of LWE guarantees that pk looks pseudo-random to the adversary. Moreover notice that sk is never used.

Indistinguishability of H_2 and H_3: Let us define the following random variables:

- X is the random variable that takes uniform values of the form $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{b} \in \mathbb{Z}_q^n)$.
- W is the random variable that takes uniform values of the form $\mathbf{r} \in \{0,1\}^m$.
- I is the random variable that takes values of the form $(\langle \mathbf{e}_1, \mathbf{r} \rangle, \ldots, \langle \mathbf{e}_\ell, \mathbf{r} \rangle)$, where $\mathbf{e}_i \leftarrow \chi^m$, $\mathbf{r} \in \{0,1\}^m$.

Then, by Lemma 6.3, we have that $\tilde{\mathbf{H}}_\infty(W|I) \geq \mathbf{H}_\infty(W) - (\ell - 1)\log q = m - (\ell - 1)\log q$. Now, notice that $H_X(W) = H_{(\mathbf{A},\mathbf{b})}(\mathbf{r}) = (\mathbf{A}^\top \mathbf{r}, \langle \mathbf{b}, \mathbf{r} \rangle)$ is a universal hash function and by applying the generalized leftover hash lemma (Lemma 6.4), we have that:

$$\mathbf{SD}((H_X(W), X, I), (U, X, I)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\mathbf{H}}_\infty(W|I)} q^{n+1}} \ .$$

Then, if $m \geq (n + \ell + 1)\log q + 2\log \frac{1}{\epsilon} + \Omega(1)$, the statistical distance between the two views is at most ϵ.

Indistinguishability of H_3 and H_4: These are exactly the same distribution as $\mathbf{r}$ is used nowhere else.

Indistinguishability of H_4 and H_5: The change from H_5 to H_4 is the same as the change from H_2 to H_3, except that no information about $\mathbf{r}$ is leaked. So Lemma 6.3 gives us that the statistical distance between the two views is at most ϵ.

Indistinguishability of H_5 and H_6: Once again, the hardness of LWE guarantees that pk looks pseudo-random to the adversary. ∎

Acknowledgments

We thank the anonymous reviewers for their fruitful comments and for pointing out an issue with an instantiation based on Paillier's encryption scheme. We are also grateful to Damien Stehlé for comments on previous versions of this paper.

This work was supported in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud) and by the *Chaire France Telecom*.

References

[ABN10] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In *TCC 2010*, *LNCS* 5978, pages 480–497. Springer, February 2010. (Page 6.)

[BBS03] M. Bellare, A. Boldyreva, and J. Staddon. Randomness re-use in multi-recipient encryption schemeas. In *PKC 2003*, *LNCS* 2567, pages 85–99. Springer, January 2003. (Pages 3 and 10.)

[BCP14] E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In *TCC 2014*, *LNCS* 8349, pages 52–73. Springer, February 2014. (Pages 1 and 4.)

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, *LNCS* 2139, pages 213–229. Springer, August 2001. (Page 1.)

[BO13] M. Bellare and A. O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS 13*, *LNCS* 8257, pages 218–234. Springer, November 2013. (Page 6.)

[BR06] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2006*, *LNCS* 4004, pages 409–426. Springer, May / June 2006. (Page 5.)

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, *LNCS* 6597, pages 253–273. Springer, March 2011. (Pages 1 and 6.)

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, *LNCS* 4392, pages 535–554. Springer, February 2007. (Page 1.)

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, *LNCS* 2260, pages 360–363. Springer, December 2001. (Page 1.)

[DRS04] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, *LNCS* 3027, pages 523–540. Springer, May 2004. (Page 21.)

[ElG84] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, *LNCS* 196, pages 10–18. Springer, August 1984. (Page 3.)

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. (Page 20.)

[Gen06] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT 2006*, *LNCS* 4004, pages 445–464. Springer, May / June 2006. (Page 1.)

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Pages 1 and 4.)

[GGHZ14] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622, 2014. http://eprint.iacr.org/2014/622. (Pages 1 and 4.)

[GLW12] S. Goldwasser, A. B. Lewko, and D. A. Wilson. Bounded-collusion IBE from key homomorphism. In *TCC 2012*, *LNCS* 7194, pages 564–581. Springer, March 2012. (Page 3.)

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Page 1.)

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, *LNCS* 4965, pages 146–162. Springer, April 2008. (Pages 1, 2, and 4.)

[Kur02] K. Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *PKC 2002*, *LNCS* 2274, pages 48–63. Springer, February 2002. (Page 10.)

[LOS+10] 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 2010*, *LNCS* 6110, pages 62–91. Springer, May 2010. (Pages 1 and 2.)

[O'N10] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556. (Page 1.)

[OT12] T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, *LNCS* 7237, pages 591–608. Springer, April 2012. (Pages 1 and 2.)

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93. ACM Press, May 2005. (Page 21.)

[Rot11] R. Rothblum. Homomorphic encryption: From private-key to public-key. In *TCC 2011*, *LNCS* 6597, pages 219–234. Springer, March 2011. (Page 3.)

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, *LNCS* 196, pages 47–53. Springer, August 1984. (Page 1.)

[SW05] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, *LNCS* 3494, pages 457–473. Springer, May 2005. (Page 1.)

[TW14] S. Tessaro and D. A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *PKC 2014*, *LNCS* 8383, pages 257–274. Springer, March 2014. (Page 3.)

[Wat12] B. Waters. Functional encryption for regular languages. In *CRYPTO 2012*, *LNCS* 7417, pages 218–235. Springer, August 2012. (Page 1.)

[Wat14] B. Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014. http://eprint.iacr.org/2014/588. (Pages 1 and 4.)