Better Security for Functional Encryption for Inner Product Evaluations

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

January 28, 2016

Abstract

Functional encryption is a new public key paradigm that solves, in a non-interactive way, most of the security challenges raised by cloud computing. A recent paper by Abdalla, Bourse, De Caro, and Pointcheval shows a functional encryption scheme for evaluations of inner products whose security can be proven under simple assumptions. Inner product evaluation is a simple, but quite powerful functionality, that suffices for many concrete applications. We analyze the different security notions for functional encryption for inner product evaluation and propose a new generic construction that achieves security against adaptive adversaries. We show 3 instantiations based on the ElGamal encryption (plain DDH assumption), Paillier/BCP encryption (DCR assumption), and Regev encryption (LWE assumption). All of them have different advantages and drawbacks, but with acceptable trade-offs, and rely on standard assumptions.

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

1 Introduction 1

2 Basic Tools 4
2.1 Notation and Conventions 4
2.2 Public-Key Encryption 4
2.3 Functional Encryption 5

3 Relations between Simulation and Indistinguishability 7

4 A Generic Fully Secure Functional Inner-Product Encryption Scheme 9
4.1 Inner-Product Modulo p 11
4.2 Inner-Product over the Integers 12

5 Instantiation from ElGamal 17

6 Instantiation from BCP 18

7 Instantiation from Regev 20

A Tools and Assumptions 24

B Security properties of Regev's PKE 25

1 Introduction

Functional Encryption. In traditional encryption schemes, decryption is always all-or-nothing: either you can recover all data or you can recover none. *Functional encryption* (FE) [SW05, BSW11, O'N10] is an emerging paradigm that allows fine-grained control of the information revealed by a combination of a ciphertext and a secret key. In a *functional encryption* scheme for a functionality $F : K \times X \to \Sigma \cap \{\bot\}$, a master authority generates secret keys sk_k for values k in the key space K. When using sk_k on a ciphertext, the key holder only learns $F(k, x)$ and nothing else, where $x \in X$ is the encrypted data.

The concept finds numerous applications. In cloud computing, a user could store his encrypted data on an untrusted remote server, and then give secret keys to other users, sharing with each user a precise amount of the information contained in the data, or he could give a key to the server to delegate computation on the data while ensuring minimum leakage of information.

The traditional security requirement is resistance to collusions. Intuitively, an adversary knowing secret keys sk_{k_i} for a set of values $\{k_i\}$ cannot learn any information other than $\{F(k_i, x)\}$ if given an encryption of x. There are two security definitions that capture this idea of collusion-resistance:

***indistinguishability-based* security (IND).** Informally, it requires that an adversary cannot tell apart which of two message x_0, x_1 of its choice has been encrypted given the public key and any secret key sk_k such that $F(k, x_0) = F(k, x_1)$. This models the idea that an individual's message is still secure even if an arbitrary number of other users of the system collude against that user.

***simulation-based* security (SIM).** Informally, it requires that the "view" of the adversary can be simulated by a simulator that is given neither ciphertexts nor keys but only the corresponding outputs of the functionality on the underlying plaintexts.

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. They also show 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.

Functional Encryption for Inner Product Evaluations. A recent line of work opened by Abdalla, Bourse, De Caro, and Pointcheval [ABDP15] aims at constructing functional encryption schemes based on standard assumptions such as the plain decisional Diffie-Hellman assumption. They proposed a framework to construct IND-secure FE scheme to evaluate inner products from any PKE scheme with good properties.

More precisely, one knowing a secret key for a vector $\mathbf{y}$ can recover $\langle \mathbf{x}, \mathbf{y} \rangle$ given a ciphertext for the vector $\mathbf{x}$ and nothing else. This is different from prior notions of inner product encryption in [AAB+15, KSW08, LOS+10, AFV11], because here you compute an actual value, instead of all-or-nothing decryption.

Evaluating inner products is an interesting functionality, because it is sufficient to compute any polynomial evaluation on the data, by expanding the ciphertext with all monomials appearing in the desired family of polynomials. In addition, functional inner product encryption also finds direct applications:

Distance between two points The inner product between two unit vectors is exactly the cosine of the angle between the two vectors. This can be used as a distance between two points. Private evaluation of two vectors of same length allows for computation of distance from a known vector to an unknown vector. One could think of applications such as online dating, where a user would know his distance to all users without learning any personal information about them.

Weighted means The inner product corresponds to weighted means, which is one of the main tools used in statistics. This is a way to derive useful information on a big set of personal data while leaking no information about the personal data of any particular user. One could think of application such as course grades online, where teachers, students, other administrative staff members could access different statistics about the grades while preserving privacy of others.

This Work. Our work builds on the scheme proposed by [ABDP15] that only reaches security against a selective adversary, and does not try to hide the function computed with a certain key. In this paper, we analyze stronger notions of security for a functional encryption scheme for inner products. We are interested in improving the framework of [ABDP15] to provide security against adaptive adversary. We also study the relation between indistinguishability-based security and the stronger simulation-based security. For general functionalities, SIM security (the above simulation-based security) implies IND-CPA security (indistinguishability under chosen-plaintext attacks), but nothing can be said about NA-SIM (non-adaptive simulation-based security) and IND-CPA (indistinguishability under chosen-plaintext attacks). We prove that those two notions are equivalent in the case of functional encryption for inner products, if the scheme allows key delegation. This result gives a better understanding of the leakage implied by giving inner product of a vector with a known vector: a selective adversary has almost the same power as an adaptive one, given the strong malleability (even linearity) of the functionality.

Next, we proceed with the description of our framework, which constructs an FE scheme secure against adaptive adversaries, using a PKE having simple properties (the same properties as those defined by [ABDP15]). The resulting scheme is almost as efficient as the existing one, adding only one group element. Our proof of security is different from the one in [ABDP15] in that we do not need to compute a change of basis in the secret key space. This allows us to present a new instantiation based on the BCP PKE [BCP03], in which the secret key space order is unknown, and whose security relies on the DCR assumption (Decisional Composite Residuosity), which is a special case of the EDDH assumption.

Relation to [ALS15, BJK15]. [ALS15][version 20150629:075619] recently gave two constructions for FE schemes for inner product evaluations secure against adaptive adversaries. The first one is based on DDH and the second one on LWE. Our generic construction is inspired by their construction relying on the DDH assumption and our instantiation based on the ElGamal cryptosystem gives the same FE scheme. Our approach was to extract the properties needed from a PKE scheme to obtain a FE scheme using this method. This allowed us to prove the security of two new schemes, one based on LWE and one based on DCR. However, our construction based on LWE is restricted to inner products over $\mathbb{Z}$ while the construction of [ALS15] based on the LWE assumption aimed at evaluating modular inner product. In a new version, they concurrently proposed one more construction for fully secure functional encryption for inner product evaluations whose security is based on the DCR assumption, getting 3 different constructions. Thus, they achieve similar results as we do, and they even construct modular

	ElGamal	BCP	Regev
mpk	$(\ell+1)\log p$	$2(\ell+1)\log N$	$m(\ell+1)\log q$
msk	$2\ell\log p$	$\ell(2\log N+\lambda+\log T)$	$n(\ell)\log q+\ell\log T$
$\mathsf{Ct}_{\mathbf{x}}$	$(\ell+2)\log p$	$2(\ell+2)\log N$	$(n+\ell+1)\log q$
$\mathsf{Sk}_{\mathbf{y}}$	$2\log p$	$2\log \ell M_y N+\lambda+\log T$	$n\log q+\log \ell M_y+\log T$
output space	computation of DL	N	$\ell M_x M_y$

Table 1: Parameter Sizes. Here, ℓ is the length of the vectors encrypted in the ciphertexts and encoded in the secret keys, T is a parameter that allows some trade-off between security and efficiency. In ElGamal, p is the size of a group element. In BCP, N is an RSA modulus, the size of a group element, λ is the security parameter. In Regev, q is the size of a group element, p is the order of $\ell M_x M_y$, M_y is a bound on the key space, and M_x is a bound on the message space.

inner product functionality when we only get it over the integers. In fact, their construction also gives slightly better parameters for the LWE instantiation because they do not need to use a superpolynomial modulus. However, the biggest advantage of our work is that our approach is generic. Indeed, the proof can easily be adapted to new schemes, because we extracted the useful properties from the 3 different constructions, so the result is more flexible.

Another construction that improves the selective security of [ABDP15]'s scheme has been given by [BJK15]. They construct a private key FE scheme for inner product evaluations and prove its security against adaptive adversaries, and even hide the function provided by a key. This security notion can only be achieved in private key setting, so we cannot have a chance at meeting their security. On the other hand, their construction uses asymmetric pairings, which reduce the efficiency of the scheme. Finally, as in our ElGamal-based instantiation, their construction also needs the computation of a discrete logarithm during the decryption.

Parameter Sizes For Our Constructions. In Table 1, we compare the size of the parameters and ciphertexts for each concrete functional encryption scheme. Each column refers to an instantiation of our generic construction and is indexed by its underlying public key encryption scheme. 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 $(g, N, \mathbf{A})$. Output space is the number of different values of inner products that can be decrypted using this instantiation, it gives a bound on message and key space. The ElGamal instantiation can give short ciphertexts and keys using elliptic curves. It also allows for modular inner product evaluations, but requires the computation of a discrete logarithm in the decryption, which makes it usable for small message space only. The instantiation relying on BCP fixes this problem by having the message space exponential in the security parameter, but uses RSA moduli, which leads to larger ciphertexts and keys. Regev 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. It is also believed to be secure against adversaries using quantum computers. BCP and Regev are also unable to evaluate modular inner products, and are only proven secure when used to compute the inner product over the integers.

Relation to Predicate Encryption. As mentioned above, the notion of functional inner-product encryption which we use is different from those used in predicate encryption schemes, since the user in possession of a secret key for a function f in our case actually obtains the value of $f(m)$, where m is the message being encrypted.

In contrast, *predicate encryption* (PE) 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 $\mathsf{m} \in M$.

Some examples of predicate encryption schemes in which the index ind is kept private are anonymous identity-based encryption (AIBE) [BF01, ABC$^+$05, Gen06], hidden vector encryption (HVE) [BW07], and predicate inner-product encryption [KSW08, LOS$^+$10, OT12]. On the other hand, the cases where the index ind is easily readable from the ciphertext includes identity-based encryption (IBE) [Sha84, BF01, Coc01], attribute-based encryption (ABE) [SW05, GPSW06], and functional encryption for regular languages [Wat12].

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 $x \overset{R}{\leftarrow} S$ 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 $y \overset{R}{\leftarrow} \mathcal{A}(x)$ 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.

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:

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

proc Initialize($\lambda$)
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    Return $\mathsf{pk}$

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

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

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

proc Initialize($\lambda, \mathsf{m}_0^*, \mathsf{m}_1^*$)
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{Setup}(1^\lambda)$
    Return $\mathsf{pk}$

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

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

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* ($\mathsf{IND\text{-}CPA}$ *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 ($\mathsf{IND\text{-}CPA}$ secure, for short) if

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

We also define *selective security against chosen-plaintext attacks* ($\mathsf{s\text{-}IND\text{-}CPA}$ *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** could 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

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

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

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

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

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 $(b' = b)$
```

Figure 2: Game $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa\text{-}}b}_{\mathcal{FE},\lambda}(\mathcal{A})$ define IND-FE-CPA security of $\mathcal{FE}$.

a special string not contained in Σ. Notice that the functionality is undefined when 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 $F(k, m) \neq \perp$[1], except with negligible probability.

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

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

Non-Adaptive Simulation-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 *non-adaptive simulation security* (NA-SIM *security*, for short) via the security game depicted on Figure 3. Then, we say that $\mathcal{FE}$ is simulation secure against non-adaptive adversaries (NA-SIM secure, for short) if there exists a PPT simulator $\mathcal{S}$ such that

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

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

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

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

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

proc LR($\mathsf{m}^*$)
    $\mathsf{Ct}^*_0 \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathsf{m}^*)$
    $\mathsf{Ct}^*_1 \leftarrow \mathcal{S}(mpk, \{F(k, \mathsf{m}^*), k\}_{k \in V})$
    Return $\mathsf{Ct}^*_b$

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

Figure 3: Game $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}}b}_{\mathcal{FE},\lambda}(\mathcal{A})$ define $\mathsf{NA\text{-}SIM}$ security of $\mathcal{FE}$. The procedure **KeyDer** can only be invoked before invoking **LR**.

Inner-Product Functionality. In this paper we are interested in two different functionalities:

- The *inner-product functionality* over $\mathbb{Z}_p$ (IP_p, 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 $\mathsf{IP}_{p,\ell}(k, x) = \langle k, x\rangle \bmod p$. When it is clear from the context we remove the reference to the length ℓ.
- And the *inner-product functionality* over $\mathbb{Z}$ (IP, for short), which is the same functionality but the result is not reduced modulo p. It is a family of functionalities with key space $K_\ell = \{0 \dots M_y\}^\ell$ and message space $X_\ell = \{0 \dots M_x\}^\ell$ consisting of integer vectors of length ℓ: for any $k \in K_\ell, x \in X_\ell$ the functionality $\mathsf{IP}_\ell(k, x) = \langle k, x\rangle$. When it is clear from the context we remove the reference to the length ℓ.

3 Relations between Simulation and Indistinguishability

In this section, we compare the two a-priori incomparable security notions of $\mathsf{NA\text{-}SIM}$ and $\mathsf{IND\text{-}FE\text{-}CPA}$. It is a known result that simulation security implies indistinguishability for any functionality. The next theorem shows that $\mathsf{IND\text{-}FE\text{-}CPA}$ security notion lies somewhere between SIM security and $\mathsf{NA\text{-}SIM}$ security for the inner-product functionality. Although we prove this result from scratch for completeness, we remark that it follows from the work of [O'N10], because the inner-product functionality is preimage sampleable, as noted in [ALS15].

Theorem 3.1 *Let $\mathcal{FE}$ be a functional encryption scheme for the inner-product functionality. If $\mathcal{FE}$ is* $\mathsf{IND\text{-}FE\text{-}CPA}$ *secure, then it is* $\mathsf{NA\text{-}SIM}$ *secure.*

Proof: We prove this statement by exhibiting a simulator $\mathcal{S}$ and showing that if $\mathcal{FE}$ is $\mathsf{IND\text{-}FE\text{-}CPA}$ secure, then $\mathcal{S}$ satisfies the properties needed to prove $\mathsf{NA\text{-}SIM}$ security of $\mathcal{FE}$.
$\mathcal{S}$ is given as input mpk and the set of values $\{\langle \mathbf{x}, \mathbf{y}\rangle, \mathbf{y}\}_{\mathbf{y} \in V}$ for some unknown $\mathbf{x}$. It then finds a vector $\mathbf{x}'$ such that $\langle \mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}', \mathbf{y}\rangle$ for all $\mathbf{y} \in V$ and encrypts it using mpk. $\mathcal{S}$ returns this new formed ciphertext.
If an adversary $\mathcal{A}$ wins Game $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}}b}_{\mathcal{FE},\lambda}(\mathcal{A})$, it wins Game $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa\text{-}}b}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenge messages $\mathbf{x}$ and $\mathbf{x}'$ and without key queries after the challenge.

▮

The next question is whether NA-SIM security implies IND-FE-CPA security or not. We answer this question in a non-black box manner: We require that the functional encryption scheme supports key delegation in order to prove equivalence of NA-SIM security and IND-FE-CPA security.

Definition 3.2 [Key delegation] We say that a *functional encryption scheme for the inner-product functionality* $\mathcal{FE}$ supports key delegation if $\mathsf{sk}_{\mathbf{y}}$ can be obtained from any set $\{\mathsf{sk}_{\mathbf{z}}\}_{\mathbf{z}\in V}$ where $\mathbf{y} \in \mathrm{Span}(V)$.

Note that with this definition, any functional encryption scheme for the inner-product functionality with a deterministic key derivation algorithm supports key delegation.

Theorem 3.3 Let $\mathcal{FE}$ be a *functional encryption scheme for the inner-product functionality.* If $\mathcal{FE}$ is NA-SIM secure and supports key delegation, then it is IND-FE-CPA secure.

Proof: If there weren't any secret key queries after the challenge ciphertext has been received, we could use the same argument as for proving that SIM security implies IND-FE-CPA security. This is a simple argument on games defining the security: the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa\text{-}0}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenges $\mathbf{x}_0$ and $\mathbf{x}_1$ is the same experiment as $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}0}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenge $\mathbf{x}_0$, which is indistinguishable from the experiment $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}1}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenge $\mathbf{x}_0$, which is exactly the same as $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}1}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenge $\mathbf{x}_1$, which is in turn indistinguishable from $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}fe\text{-}0}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenge $\mathbf{x}_1$, or equivalently $\mathsf{Exp}^{\mathsf{ind\text{-}fe\text{-}cpa\text{-}1}}_{\mathcal{FE},\lambda}(\mathcal{A})$ with challenges $\mathbf{x}_0$ and $\mathbf{x}_1$. From which IND-FE-CPA security follows.

We now show that given $\mathbf{x}_0$ and $\mathbf{x}_1$, one can compute a basis $\{\mathbf{z}_i\}_{i\in[\ell]}$ of the orthogonal $(\mathbf{x}_1-\mathbf{x}_0)^{\perp}$. Given secret keys for this basis, key delegation allows to compute a secret key for any $\mathbf{y}$ such that $\langle \mathbf{x}_0, \mathbf{y}\rangle = \langle \mathbf{x}_1, \mathbf{y}\rangle$.

This is easy for IP_p because $\mathbb{Z}_p$ is a field. For IP, we want to find a basis $\{\mathbf{z}_i\}_{i\in[\ell-1]}$ of the lattice orthogonal to a given vector $\mathbf{x}$. We construct one recursively:
First, if the gcd of all coordinates of $\mathbf{x}$ is not 1, set $\mathbf{x} = \frac{1}{\gcd(x_i)}\mathbf{x}$.

- If $\ell = 2$: $\mathbf{z} = (-x_2, x_1)$ is a basis of $(x_1, x_2)^{\perp}$
- If $\ell > 2$: Let $\{\mathbf{z}_i\}_{i\in[\ell-2]}$ be a basis of $(x_1, \cdots, x_{\ell-1})^{\perp}$. We set $\mathbf{z}_{\ell-1} = (-x_\ell \times a_1, -x_\ell \times a_2, \ldots, -x_\ell \times a_{\ell-1}, \gcd(x_1, \ldots, x_{\ell-1}))$ where the a_i's come from Bezout's Identity.

This basis does not generate a sub-lattice of $\mathbf{x}^{\perp}$. Indeed, a vector $\mathbf{y}$ orthogonal to $\mathbf{x}$ has a last coordinate which is a multiple of $\gcd(x_1, \ldots, x_{\ell-1})$, because $x_\ell y_\ell = -\sum_{i\in[\ell-1]} x_i y_i$.

To conclude the proof, suppose there is an adversary $\mathcal{A}$ that breaks the IND-FE-CPA security. A simulator $\mathcal{S}$ can use this adversary to break the NA-SIM security: Upon reception of $\mathbf{x}_0$ and $\mathbf{x}_1$, $\mathcal{S}$ queries secret keys for the $\mathbf{z}_i$ before asking for a challenge ciphertext, thus avoiding the non-adaptive queries. $\mathcal{S}$ can still answer $\mathcal{A}$'s adaptive queries using key delegation.

▮

4 A Generic Fully Secure Functional Inner-Product Encryption Scheme

Our framework constructs functional encryption scheme for the inner-product functionality $\mathsf{IP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ from a public-key encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$. In order to prove the correctness and security of the new scheme, we need some structural and homomorphic properties on $\mathcal{E}$ – for correctness – and some security properties – for security. These properties are similar to the one defined in [ABDP15], and a bit more permissive, thus allowing for a new instantiation. We define those properties below:

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 the message space is $\mathcal{M}_x \subseteq \mathbb{Z}$. The secret key group G does not have to be public, as our proof only requires the operation $+$ to be publicly available. 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.
We also split the Setup algorithm for convenience in the two following algorithms to sample secret keys, and to sample 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.

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 $\mathsf{E}(\mathsf{pk}_1\mathsf{pk}_2, x_1 + x_2; r) = \mathsf{E}(\mathsf{pk}_1, x_1; r) \cdot \mathsf{E}(\mathsf{pk}_2, x_2; r)$.

And for security, we define two properties via security games. These differ from the one in [ABDP15] in that the α_i's are taken in $\mathcal{T}$ instead of the message space.

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

```
Game $\mathsf{Exp}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\lambda}(\mathcal{A})$
proc Initialize$(\lambda, \mathcal{M})$
$(\mathsf{sk}, (\alpha_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{D}(1^\lambda)$
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 $\mathcal{D}$ samples tuples of the form $(\mathsf{sk}, (\alpha_i, \mathsf{sk}_i)_{i\in[\ell]})$ where sk and the sk_i's are sampled from SKGen, and the α_i's are in $\mathcal{T}$.

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

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

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

```
Game $\mathsf{Exp}^{\ell\text{-}\mathsf{ct}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\lambda}(\mathcal{A})$
proc Initialize$(\lambda, \mathcal{M})$
$(a, (\alpha_i, x_i, \mathsf{sk}_i)_{i\in[\ell]}) \xleftarrow{R} \mathcal{D}(1^\lambda)$
$\mathsf{sk} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$; $\mathsf{pk} \leftarrow \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}, \tau')$; $(\mathsf{pk}_i \leftarrow \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}_i, \tau_i))_{i\in[\ell]}$
$\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r)$; $\mathsf{ct} = \mathcal{E}.\mathsf{E}(\mathsf{pk}, a; r)$
if $b = 0$ then $\mathsf{ct}_i = \mathsf{ct}^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)$
else $\mathsf{ct}_i = \mathsf{ct}^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}'(\mathsf{sk}_i, x_i, \mathsf{ct}_0, \tau_i)$
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

- $\mathcal{D}$ samples tuples of the form $(a, (\alpha_i, x_i, , \mathsf{sk}_i)_{i\in[\ell]})$, where sk_i's are sampled from SKGen, α_i's are in $\mathcal{T}$ and a and the x_i's are in $\mathcal{M}_x$.
- 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 the parameters needed to generate public keys, 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

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

We now show two constructions, one for the IP_p functionality that requires a public-key encryption scheme $\mathcal{E}$ that has a message space and a key space that have the same order p, and one for the IP functionality. We only sketch the proof in the first case and detail the proof of the latter because it is trickier.

4.1 Inner-Product Modulo p

Here we describe a construction of a functional encryption scheme for the functionality IP_p.

Construction 4.1 [Adaptive-PKE-IP-Mod Scheme] Let us consider a PKE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the properties defined above. We define our *functional encryption scheme for the inner-product functionality* over $\mathbb{Z}_p$ $\mathsf{IP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows. We set $\mathcal{T} = \mathbb{Z}_p$ in this case.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$ calls $\mathcal{E}$'s key generation algorithm to generate $\ell + 1$ independent secret keys pairs $s_1, \ldots, s_\ell$ and sk sharing the same public parameters params and $\mathbf{t} = (t_1, \ldots, t_\ell) \in \mathcal{T}^\ell$. Then, the algorithm sets $\mathsf{pk} = \mathsf{PKGen}(\mathsf{sk})$ and $\mathsf{pk}_i = \mathsf{PKGen}(s_i + t_i \mathsf{sk})$, and returns $mpk = (\mathsf{params}, \mathsf{pk}, \mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and $msk = (s_1, \ldots, s_\ell, t_1, \ldots, t_\ell)$.
- $\mathsf{KeyDer}(msk, \mathbf{y})$ on input master secret key msk and a vector $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathcal{M}_y^\ell$, computes $\mathsf{sk}_\mathbf{y}$ as $\mathsf{sk}_\mathbf{y} = (\sum_{i\in[\ell]} y_i \cdot s_i, \sum_{i\in[\ell]} y_i \cdot t_i)$. The second part is computed over the integers.
- $\mathsf{Encrypt}(mpk, \mathbf{x})$ on input master public key mpk and message $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathcal{M}_x^\ell$, chooses shared randomness r in the randomness space of $\mathcal{E}$, and computes $\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r)$, $\mathsf{ct}_1 = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r)$ and $\mathsf{ct}_{2,i} = \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)$. Then the algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1, (\mathsf{ct}_{2,i})_{i\in[\ell]})$.
- $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_\mathbf{y})$ on input master public key mpk, ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1, (\mathsf{ct}_{2,i})_{i\in[\ell]})$, and secret key $\mathsf{Sk}_\mathbf{y} = (\mathsf{Sk}_{\mathbf{y},0}, \mathsf{Sk}_{\mathbf{y},1})$ for vector $\mathbf{y} = (y_1, \ldots, y_\ell)$, returns the output of $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y},0}, (\mathsf{ct}_0, (\prod_{i\in[\ell]} \mathsf{ct}_{2,i}^{y_i}) \cdot \mathsf{ct}_1^{-\mathsf{Sk}_{\mathbf{y},1}}))$.

Theorem 4.2 If the underlying PKE $\mathcal{E}$ has message space and secret key space of same order p, if it is $\mathsf{s\text{-}IND\text{-}CPA}$, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, ℓ-public-key-reproducible, ℓ-ciphertext-reproducible with coefficients $\alpha_i \in \mathcal{T} = \mathbb{Z}_p$, then Construction 4.1 is $\mathsf{IND\text{-}FE\text{-}CPA}$.

Proof Sketch. The proof of security can be informally reduced to 2 different ideas:

- First, each master public key corresponds to multiple different master secret keys. This is due to the fact that we generate $\ell + 1$ public keys from $\ell + 1$ secret keys and ℓ random values in $\mathbb{Z}_p$.
- Second, we show that we can generate ill-formed ciphertext that encrypts vector $\mathbf{x} + \mathbf{t}$ with $\mathsf{ct}_1 = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 1; r)$ that is indistinguishable from a well-formed ciphertext that encrypts $\mathbf{x}$. We create it using reproducibility and homomorphic properties, and the indistinguishability directly comes from the $\mathsf{s\text{-}IND\text{-}CPA}$security of $\mathcal{E}$.

Now we argue that a public key for $(s_i, t_i)_{i\in[\ell]}$ is also a public key for $(s'_i, t'_i)_{i\in[\ell]}$, with $t'_i = t_i + x_{b,i} - x_{1-b,i}$ and $s'_i = s_i + (x_{1-b,i} - x_{b,i})\mathsf{sk}$ for all $\mathbf{x}_b$ and $\mathbf{x}_{1-b}$. So we can show that a well-formed ciphertext for a vector $\mathbf{x}_b$ is indistinguishable from an ill-formed ciphertext for the vector $\mathbf{x}_b + \mathbf{t} = \mathbf{x}_{1-b} + \mathbf{t}'$ which is in turn indistinguishablee from a well-formed ciphertext for the vector $\mathbf{x}_{1-b}$. Moreover, it is easy to see that the secret keys query by the adversary are the same whether $msk = (s_i, t_i)_{i\in[\ell]}$ or $msk = (s'_i, t'_i)_{i\in[\ell]}$. So the adversary has the same view in both cases, which concludes the proof.

A more detailed version a the proof can be obtain by reading the proof of the next theorem, replacing $\mathcal{T}$ by $\mathbb{Z}_p$. Note that in this case, the simulator never needs to abort, so the proof is tight.

4.2 Inner-Product over the Integers

If the message space and the secret key space do not have the same order, we can still construct a functional encryption scheme for the functionality IP if all the others properties are satisfied. Note that this functionality is not as simple to implement because security is proven only for vectors $\mathbf{x}$ and $\mathbf{y}$ respectively in $\mathcal{M}_x$ and $\mathcal{M}_y$. Nothing is known about the security if one uses this scheme with too large elements. The construction is basically the same as Construction 4.1, but $\mathcal{T}$ is set to be $[0,\dots,T]$ and the computations related to $\mathbf{t}$ are done over the integers.

Construction 4.3 [Adaptive-PKE-IP Scheme] Let us consider a PKE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the properties defined previously. We define our *functional encryption scheme for the inner-product functionality* over $\mathbb{Z}$ $\mathsf{IP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows. We set $\mathcal{T} = [0,\dots,T]$ in this case, where T will be set according to the security properties needed. (T/M_x superpolynomial is needed for security against polynomially bounded adversaries. T/M_x exponential provides security against sub exponetially bounded adversaries)

- $\mathsf{Setup}(1^\lambda, 1^\ell, M_x, M_y, T)$ calls $\mathcal{E}$'s key generation algorithm to generate $\ell+1$ independent secret keys pairs $s_1,\dots,s_\ell$ and sk sharing the same public parameters params and $\mathbf{t} = (t_1,\dots,t_\ell) \in \mathcal{T}^\ell$. Then, the algorithm sets $\mathsf{pk} = \mathsf{PKGen}(\mathsf{sk})$ and $\mathsf{pk}_i = \mathsf{PKGen}(s_i + t_i\mathsf{sk})$, and returns $mpk = (\mathsf{params}, \mathsf{pk}, \mathsf{pk}_1,\dots,\mathsf{pk}_\ell)$ and $msk = (s_1,\dots,s_\ell,t_1,\dots,t_\ell)$.
- $\mathsf{KeyDer}(msk, \mathbf{y})$ on input master secret key msk and a vector $\mathbf{y} = (y_1,\dots,y_\ell) \in \mathcal{M}_y^\ell$, computes $\mathsf{sk}_\mathbf{y}$ as $\mathsf{sk}_\mathbf{y} = (\sum_{i\in[\ell]} y_i \cdot s_i, \sum_{i\in[\ell]} y_i \cdot t_i)$. The second part is computed over the integers.
- $\mathsf{Encrypt}(mpk, \mathbf{x})$ on input master public key mpk and message $\mathbf{x} = (x_1,\dots,x_\ell) \in \mathcal{M}_x^\ell$, chooses shared randomness r in the randomness space of $\mathcal{E}$, and computes $\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r)$, $\mathsf{ct}_1 = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r)$ and $\mathsf{ct}_{2,i} = \mathcal{E}.\mathsf{E}(\mathsf{pk}_i, x_i; r)$. Then the algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1, (\mathsf{ct}_{2,i})_{i\in[\ell]})$.
- $\mathsf{Decrypt}(mpk, \mathsf{Ct}, \mathsf{sk}_\mathbf{y})$ on input master public key mpk, ciphertext $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1, (\mathsf{ct}_{2,i})_{i\in[\ell]})$, and secret key $\mathsf{Sk}_\mathbf{y} = (\mathsf{Sk}_{\mathbf{y},0}, \mathsf{Sk}_{\mathbf{y},1})$ for vector $\mathbf{y} = (y_1,\dots,y_\ell)$, returns the output of $\mathcal{E}.\mathsf{Decrypt}(\mathsf{Sk}_{\mathbf{y},0}, (\mathsf{ct}_0, (\prod_{i\in[\ell]} \mathsf{ct}_{2,i}^{y_i}) \cdot \mathsf{ct}_1^{-\mathsf{Sk}_{\mathbf{y},1}}))$.

Theorem 4.4 If the underlying PKE $\mathcal{E}$ is $\mathsf{s\text{-}IND\text{-}CPA}$, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, ℓ-public-key-reproducible and ℓ-ciphertext-reproducible with coefficients $\alpha_i \in \mathcal{T}$, then Construction 4.3 is $\mathsf{IND\text{-}FE\text{-}CPA}$.

Proof: We prove security via a sequence of hybrid experiments, and then we show they are indistinguishable. The techniques used to switch public keys and ciphertext values with reproducibility properties closely follows the work of [ABDP15].

Hybrid H_1: This is the $\mathsf{IND\text{-}FE\text{-}CPA}$ game:

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

proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

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

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_2: This is like H_1, except that the simulator raises an abort flag if $\mathbf{t}+\mathbf{x}_b+\mathbf{x}_{1-b} \notin \mathcal{T}^\ell$.

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

proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

proc LR($\mathbf{x}_0, \mathbf{x}_1$)
    ┌──────────────────────────────────────────────┐
    │ if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$
    │     then bad $\leftarrow$ true
    └──────────────────────────────────────────────┘
    $\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
    Return $\mathsf{Ct}^*$

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_1 and H_2 give exactly the same view to the adversary: $H_2 = H_1$.

Hybrid H_3: This is like H_2, except that the simulator do not use bad instances.

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

proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

proc LR($\mathbf{x}_0, \mathbf{x}_1$)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then bad $\leftarrow$ true
    $\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
    Return $\mathsf{Ct}^*$

proc Finalize($b'$)
    ┌──────────────────────────────────────────────┐
    │ if bad $\vee$ ($\exists \mathbf{y} \in V$ such that
    │     $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$)
    │     then return false
    └──────────────────────────────────────────────┘
    Return $(b' = b)$
```

If the adversary has advantage ϵ in H_2, then it has advantage $\epsilon - \Pr[\mathsf{bad}]$ in H_3[2]. Note that $\Pr[\mathsf{bad}] < 1 - (1 - M_x/T)^\ell$, and it is negligible if T/M_x is set to be superpolynomial in λ. Also note that in the case of the IP_p functionality, $\Pr[\mathsf{bad}] = 0$ because $\mathcal{T} = \mathbb{Z}_p$, so $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \in \mathcal{T}$.

[2]In a previous version, $\epsilon / \Pr[\mathsf{bad}]$ was claimed, but this requires the set $\mathcal{T}$ to be chosen such that there is no element $\mathbf{t} \in \mathcal{T}$, together with a vector $\mathbf{x} \in [-M_x, M_x]^\ell$ such that neither $\mathbf{t} + \mathbf{x}$ nor $\mathbf{t} - \mathbf{x}$ is in $\mathcal{T}$

Hybrid H_4: This is like H_3 except that the master public key is generated by invoking the algorithm $H_4.\mathsf{Setup}$ defined as follows:

$H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$: The algorithm samples $\mathsf{sk} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$, for $i \in [\ell]$, PKE secret key $s_i \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\lambda)$ and uniformly random scalar $t_i \xleftarrow{R} \mathcal{T}$. Finally, the algorithm sets:

$$\mathsf{pk} = \mathcal{E}.\mathsf{PKGen}(\mathsf{sk}, \tau) \qquad \mathsf{sk}_i = s_i + t_i \cdot \mathsf{sk}$$
$$\mathsf{pk}_{s_i} = \mathcal{E}.\mathsf{PKGen}(s_i, \tau_i) \qquad \mathsf{pk}_i = \mathsf{pk}^{t_i} \cdot \mathsf{pk}_{s_i}$$

where τ is the same as used in the Setup algorithm, and τ_i is such that $\mathsf{pk}_{s_i} \cdot \mathsf{pk}^{t_i}$ is close to $\mathcal{E}.\mathsf{PKGen}(\mathsf{sk}_i)$. The algorithm returns $mpk = (\mathsf{pk}, (\mathsf{pk}_i)_{i\in[\ell]})$ and $msk = (s_i, t_i)_{i\in[\ell]}$.

```
proc Initialize($\lambda$)
    $\boxed{(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)}$
    $V \leftarrow \emptyset$
    Return $mpk$
proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$
proc LR($\mathbf{x}_0, \mathbf{x}_1$)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then bad $\leftarrow$ true
    $\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
    Return $\mathsf{Ct}^*$
proc Finalize($b'$)
    if bad $\vee$ ($\exists \mathbf{y} \in V$ such that
        $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$)
        then return false
    Return ($b' = b$)
```

Under the ℓ-public-key-reproducibility of $\mathcal{E}$, H_3 and H_4 are indistinguishable.

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:

$H_5.\mathsf{Encrypt}(msk, \mathsf{pk}, \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r) \qquad \mathsf{ct}_1 = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0; r) \qquad \mathsf{ct}_{2,i} = \mathsf{ct}_1^{t_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_{s_i}, x_i; r)$$

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

```
proc Initialize($\lambda$)
    $(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$
    $V \leftarrow \emptyset$
    Return $mpk$
proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$
proc LR($\mathbf{x}_0, \mathbf{x}_1$)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then bad $\leftarrow$ true
    $\boxed{\mathsf{Ct}^* \xleftarrow{R} H_5.\mathsf{Encrypt}(msk, \mathsf{pk}, \mathbf{x}_b)}$
    Return $\mathsf{Ct}^*$
proc Finalize($b'$)
    if bad $\vee$ ($\exists \mathbf{y} \in V$ such that
        $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$)
        then return false
    Return ($b' = b$)
```

By linear ciphertext-homomorphism of $\mathcal{E}$, $H_4 = H_5$.

Hybrid H_6: This is like H_5 except that the challenge ciphertext is generated by invoking the algorithm $H_6.\mathsf{Encrypt}$ defined as follows:

$H_6.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x})$: Let $\mathsf{Ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$. Then, the algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$\mathsf{ct}'_0 = \mathsf{ct}_0 \qquad \mathsf{ct}'_1 = \mathsf{ct}_1 \qquad \mathsf{ct}'_{2,i} = \mathsf{ct}_1^{t_i} \cdot \mathcal{E}.\mathsf{E}'(s_i, x_i, \mathsf{ct}_0; \tilde{r})$$

where $\mathcal{E}.\mathsf{E}'$ is the alternative encryption algorithm defined in the ℓ-ciphertext-reproducibility game, $\tilde{r}$ is some randomness shared among all the invocation of $\mathcal{E}.\mathsf{E}'$.

```
proc Initialize(λ)
    $(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$
    $V \leftarrow \emptyset$
    Return $mpk$

proc KeyDer(y)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

proc LR(x_0, x_1)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then $abort \leftarrow \mathsf{true}$
    $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 0)}$
    $\boxed{\mathsf{Ct}^* \xleftarrow{R} H_6.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x}_b)}$
    Return $\mathsf{Ct}^*$

proc Finalize(b')
    if $\mathsf{bad} \vee (\exists \mathbf{y} \in V$ such that
        $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1))$
        then return false
    Return $(b' = b)$
```

Under the ℓ-ciphertext-reproducibility of $\mathcal{E}$, H_5 and H_6 are indistinguishable.

Hybrid H_7: This is like H_6 except that Ct encrypts a random value $a \in \mathbb{Z}_p$.

```
proc Initialize(λ)
    $(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$
    $V \leftarrow \emptyset$
    Return $mpk$

proc KeyDer(y)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

proc LR(x_0, x_1)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then $abort \leftarrow \mathsf{true}$
    $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 1)}$
    $\mathsf{Ct}^* \xleftarrow{R} H_6.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x}_b)$
    Return $\mathsf{Ct}^*$

proc Finalize(b')
    if $\mathsf{bad} \vee (\exists \mathbf{y} \in V$ such that
        $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1))$
        then return false
    Return $(b' = b)$
```

Under the $\mathsf{s\text{-}IND\text{-}CPA}$ security of $\mathcal{E}$, H_6 and H_7 are indistinguishable.

Hybrid H_8: This is like H_7 except that the challenge ciphertext is generated by invoking the algorithm $H_8.\mathsf{Encrypt}$ defined as follows:

```
$H_8$.Encrypt$(msk, \mathsf{pk}, \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$
in the following way:
    $\mathsf{ct}_0 = \mathcal{E}.\mathsf{C}(r) \quad \mathsf{ct}_1 = \mathcal{E}.\mathsf{E}(\mathsf{pk}, 1; r) \quad \mathsf{ct}_{2,i} = \mathsf{ct}_1^{t_i} \cdot \mathcal{E}.\mathsf{E}(\mathsf{pk}_{s_i}, x_i; r),$
where $r$ is some randomness in the random space of $\mathcal{E}$.
```

```
proc Initialize($\lambda$)
$(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$
$V \leftarrow \emptyset$
Return $mpk$

proc KeyDer($\mathbf{y}$)
$V \leftarrow V \cup \{\mathbf{y}\}$
$\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
Return $\mathsf{sk}_\mathbf{y}$

proc LR($\mathbf{x}_0, \mathbf{x}_1$)
if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
    then $abort \leftarrow$ true
$\boxed{\mathsf{Ct}^* \xleftarrow{R} H_8.\mathsf{Encrypt}(msk, \mathsf{pk}, \mathbf{x}_b)}$
Return $\mathsf{Ct}^*$

proc Finalize($b'$)
if bad $\vee$ ($\exists \mathbf{y} \in V$ such that
    $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$)
    then return false
Return $(b' = b)$
```

Under the ℓ-ciphertext-reproducibility of $\mathcal{E}$, H_7 and H_8 are indistinguishable.

Hybrid H_9: This is like H_8 except that the challenge ciphertext is generated by invoking the algorithm Encrypt.

```
proc Initialize($\lambda$)
$(mpk, msk) \xleftarrow{R} H_4.\mathsf{Setup}(1^\lambda, 1^\ell)$
$V \leftarrow \emptyset$
Return $mpk$

proc KeyDer($\mathbf{y}$)
$V \leftarrow V \cup \{\mathbf{y}\}$
$\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
Return $\mathsf{sk}_\mathbf{y}$

proc LR($\mathbf{x}_0, \mathbf{x}_1$)
if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
    then bad $\leftarrow$ true
$\boxed{\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b + \mathbf{t})}$
Return $\mathsf{Ct}^*$

proc Finalize($b'$)
if bad $\vee$ ($\exists \mathbf{y} \in V$ such that
    $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1)$)
    then return false
Return $(b' = b)$
```

By linear ciphertext-homomorphism of $\mathcal{E}$, $H_8 = H_9$.

Hybrid H_{10}: This is like H_9 except that the master public key is generated by invoking the algorithm Setup.

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

proc KeyDer($\mathbf{y}$)
    $V \leftarrow V \cup \{\mathbf{y}\}$
    $\mathsf{sk}_\mathbf{y} \xleftarrow{R} \mathsf{KeyDer}(msk, \mathbf{y})$
    Return $\mathsf{sk}_\mathbf{y}$

proc LR($\mathbf{x}_0, \mathbf{x}_1$)
    if $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \notin \mathcal{T}^\ell$:
        then $\mathsf{bad} \leftarrow \mathsf{true}$
    $\mathsf{Ct}^* \xleftarrow{R} \mathsf{Encrypt}(mpk, \mathbf{x}_b + \mathbf{t})$
    Return $\mathsf{Ct}^*$

proc Finalize($b'$)
    if $\mathsf{bad} \vee (\exists \mathbf{y} \in V$ such that
        $F(\mathbf{y}, \mathbf{x}_0) \neq F(\mathbf{y}, \mathbf{x}_1))$
        then return $\mathsf{false}$
    Return $(b' = b)$
```

Under the ℓ-public-key-reproducibility of $\mathcal{E}$, H_9 and H_{10} are indistinguishable.

Advantage of any Adversary in H_{10}. As long as $\mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b} \in \mathcal{T}^\ell$, which is always the case unless bad, the advantage of the adversary in this game is 0. Because there exists $(s_i', t_i')_{i\in[\ell]}$ equally likely as $(s_i, t_i)_{i\in[\ell]}$ that gives exactly the same view by replacing $\mathbf{x}_b$ by $\mathbf{x}_{1-b}$:

$$\mathbf{t}' = \mathbf{t} + \mathbf{x}_b - \mathbf{x}_{1-b}, \quad s_i' = s_i + (\mathbf{x}_{1-b} - \mathbf{x}_b)_i \mathsf{sk}$$

Moreover, for any vector $\mathbf{y}$ satisfying the security game constraints, i.e. $\langle \mathbf{y}, \mathbf{x}_b \rangle = \langle \mathbf{y}, \mathbf{x}_{1-b} \rangle$, it holds that, since $\mathbf{x}_b + \mathbf{t} = \mathbf{x}_{1-b} + \mathbf{t}'$,

$$\sum_{i\in[\ell]} y_i t_i = \sum_{i\in[\ell]} y_i t_i' \quad \text{and} \quad \sum_{i\in[\ell]} y_i s_i = \sum_{i\in[\ell]} y_i s_i'.$$

∎

5 Instantiation from ElGamal

In this section, we show that the ElGamal encryption scheme [ElG85] can be used to instantiate Construction 4.1. Its security is based on the plain DDH assumption. This gives a very simple functional encryption scheme for the IP_p functionality. However, the message space has to be limited due to the decryption requiring the computation of a discrete logarithm. We propose in the next sections two other instantiations to overcome this limitation, but they achieve only IP functionality.

Definition 5.1 [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 the scalars $a, b, c \in \mathbb{Z}_p$ are chosen independently and uniformly at random.

Construction 5.2 [ElGamal PKE Scheme] We recall the *ElGamal public-key encryption scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\lambda)$ samples $(\mathbb{G}, q, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, and $s \leftarrow \mathbb{Z}_q$ and returns the pair $(\mathsf{pk} = g^s, \mathsf{sk} = s)$.

- $\mathsf{Encrypt}(\mathsf{pk}, m)$, on input public key pk, and message $m \in \mathbb{Z}_q$, chooses random $r \leftarrow \mathbb{Z}_q$ and computes
$$\mathsf{ct}_0 = g^r \quad \text{and} \quad \mathsf{ct}_1 = \mathsf{pk}^r g^m \ .$$
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, returns the evaluation
$$y = \log_g(\mathsf{ct}_1 \cdot \mathsf{ct}_0^{-\mathsf{sk}}).$$

Theorem 5.3 Construction 4.1 instantiated with ElGamal is IND-FE-CPA under the DDH assumption.

Proof: We now show that the above scheme possesses the properties stated in Section 4.

Semantic Security. Proof of the semantic security of this encryption scheme under the DDH assumption can be found in [ElG85].

Linear Key Homomorphism. If s_1 and s_2 are two secret keys corresponding to the public keys g^{s_1} and g^{s_2}, then $s_1 + s_2$ is a functional secret key for $g^{s_1} \cdot g^{s_2}$.

Ciphertext Homomorphism Under Shared Randomness. It is easy to verify that:
$$\mathsf{pk}_1^r g^{x_1} \cdot \mathsf{pk}_2^r g^{x_2} = (\mathsf{pk}_1 \cdot \mathsf{pk}_2) g^{x_1+x_2}.$$

ℓ-Public-Key Reproducibility. This property is perfectly satisfied, since we have $\mathsf{PKGen}(s_1 + s_2) = \mathsf{PKGen}(s_1) \cdot \mathsf{PKGen}(s_2)$.

ℓ-Ciphertext Reproducibility. This property is perfectly satisfied as well, since we have $\mathsf{ct}_0^{\mathsf{sk}} = (g^r)^s = (g^s)^r = \mathsf{pk}^r$.

▮

Note that the decryption requires the computation of a discrete logarithm to recover the final result. So this scheme can only be used if the message distribution has low entropy.

6 Instantiation from BCP

In this section, we show that the BCP encryption scheme, by Bresson, Catalano and Pointcheval [BCP03], can also be used to instantiate Construction 4.3. It is a combination of Paillier's encryption scheme and ElGamal encryption scheme. This instantiation avoids the restriction of low entropy messages by using a subgroup in which the discrete logarithm is easy to compute. On the other hand, this construction is only proven secure for the IP functionality and not IP_p.

Definition 6.1 [The Extended DDH Assumption [HO12]] Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time algorithm that takes as input a security parameter 1^λ, and outputs a triplet $(\mathbb{G}, G, \mathbf{H}, K)$ where $\mathbb{G}$ is a group whose order is a λ-bit number, $\mathbf{H}$ is a subgroup of $\mathbb{G}$, $G \subset \mathbb{G}$ and $K \subset \mathbb{Z}$. Then, the *Extended Decisional Diffie-Hellman (EDDH) assumption* states that the tuples (g, g^a, g^b, g^{ab}) and $(g, g^a, g^b, g^{ab}h)$ are computationally indistinguishable, where $(\mathbb{G}, G, \mathbf{H}, K) \leftarrow \mathsf{GroupGen}(1^\lambda)$, and $g \in G$, $a, b \in K$ and $h \in \mathbf{H}$ are chosen independently and uniformly at random.

Construction 6.2 [BCP PKE Scheme] We recall the *BCP public-key encryption scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\lambda)$ picks two safe primes p and q of λ bits each and sets the public parameters $N = pq$ and $g = g_0^{2N} \mod N^2$ with $g_0 \leftarrow \mathbb{Z}_{N^2}$. Then it samples $s \leftarrow \mathbb{Z}_N$ and returns the pair $(\mathsf{pk} = (N, g^s \mod N^2), \mathsf{sk} = s) \in \mathbb{Z}^*_{N^2} \times \mathbb{Z}_N$.
- $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{m})$ on input public key pk, and message $\mathsf{m} \in \mathbb{Z}_N$, chooses random $r \leftarrow \mathbb{Z}_{N^2}$ and computes
$$\mathsf{ct}_0 = g^r \mod N^2, \qquad \mathsf{ct}_1 = \mathsf{pk}^r (1+N)^{\mathsf{m}} \mod N^2 .$$
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, returns the evaluation
$$y = \left(\mathsf{ct}_1 \cdot \mathsf{ct}_0^{-\mathsf{sk}} - 1 \bmod N^2\right)/N.$$

Theorem 6.3 Construction 4.3 instantiated with BCP, $\mathcal{T} = \{0, \ldots, T\}$ where T is superpolynomially (resp. exponentially) bigger than M_x is $\mathsf{IND\text{-}FE\text{-}CPA}$ against polynomially bounded (resp. sub exponentially bounded) adversaries under the DCR assumption.

Proof: We now show that the above scheme possesses the properties stated in Section 4.

Semantic Security. Proof of the semantic security of this encryption scheme can be found in [BCP03]. It relies on the DCR assumption, which is a special case of the EDDH assumption.

Linear Key Homomorphism. If s_1 and s_2 are two secret keys corresponding to the public keys g^{s_1} and g^{s_2}, then $s_1 + s_2$ is a functional secret key for $g^{s_1} \cdot g^{s_2}$.

Ciphertext Homomorphism Under Shared Randomness. It is easy to verify that:
$$\mathsf{pk}_1^r (1+N)^{x_1} \cdot \mathsf{pk}_2^r (1+N)^{x_2} = (\mathsf{pk}_1 \cdot \mathsf{pk}_2)(1+N)^{x_1+x_2}.$$

ℓ-Public-Key Reproducibility. This property is perfectly satisfied, because $\mathsf{PKGen}(s_1+s_2) = \mathsf{PKGen}(s_1) \cdot \mathsf{PKGen}(s_2)$.

ℓ-Ciphertext Reproducibility. This property is perfectly satisfied as well, because we have $\mathsf{ct}_0^{\mathsf{sk}} = (g^r)^s = (g^s)^r = \mathsf{pk}^r$.

▌

Note that the secret keys are elements of a group of unknown order. But taking elements with λ more bits allows for computations over the integers, while falling outside the range of allowed secret keys with negligible probability.

7 Instantiation from Regev

In this section, we show that Regev's public-key encryption scheme [Reg05] can also be used to instantiate Construction 4.3. It has the same advantages as the instantiation from the BCP scheme, but its security relies on assumptions that are supposed to be resistant to quantum attacks.

Definition 7.1 [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 $\mathbb{Z}_q$.

Choosing the Parameters. The message space is $M = \{0, \dots, M_x\} \subseteq \mathbb{Z}_p$ for some integer M_x and prime $p > \ell M_x M_y$. Let $q = \mathsf{poly}(n) > p$ be another prime modulus that satisfies the constraints of the security proof of Section 4. 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:

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

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

Construction 7.2 [Regev PKE Scheme] We recall the *Regev public-key encryption scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\lambda)$: it first 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 $x \in M$, chooses random $\mathbf{r} \leftarrow \{0,1\}^m$ and computes
$$\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r} \in \mathbb{Z}_q^n \quad \text{and} \quad \mathsf{ct}_1 = \langle \mathsf{pk}, \mathbf{r} \rangle + t(x) \in \mathbb{Z}_q$$
where $t(x) = \lfloor v \cdot q/p \rfloor \in \mathbb{Z}_q$. 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 $x \in M$, where each x is such that $d - t(x) \in \mathbb{Z}_q$ is closest to $0 \mod q$.

Theorem 7.3 Construction 4.3 instantiated with Regev, $\mathcal{T} = \{0, \ldots, T\}$ where T is superpolynomially bigger than M_x, is IND-FE-CPA under the LWE assumption with superpolynomial gap between the error parameter and the modulus.

Proof: We now show that the above scheme possesses the properties stated in Section 4.

Semantic Security. Proof of the semantic security of this encryption scheme can be found in [Reg05]. It relies on the LWE assumption and on the leftover hash lemma, which is a statistical argument.

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 the definition of the function t.

ℓ-Public-Key Reproducibility and ℓ-Ciphertext Reproducibility. For the sake of completeness, we add proofs of ℓ-Public-Key Reproducibility and ℓ-Ciphertext Reproducibility in the appendix, but they can be found in [ABDP15]. They are both satisfied under the LWE assumption, with superpolynomial gap between the error parameter and the modulus.

∎

Acknowledgments

We would like to thank anonymous PKC16 reviewers for pointing out a flaw in a previous version of the security proof of Theorem 4.4 and giving us fruitful comments. 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 French ANR-14-CE28-0003 Project EnBiD.

References

[AAB$^+$15] Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 777–798. Springer, Heidelberg, March / April 2015.

[ABC$^+$05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005.

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, February 2010.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011.

[ALS15] Shweta Agrawal, Benoit Libert, and Damien Stehle. Fully secure functional encryption for inner products, from standard assumptions. Cryptology ePrint Archive, Report 2015/608, 2015. http://eprint.iacr.org/2015/608.

[BCP03] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54. Springer, Heidelberg, November / December 2003.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.

[BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In Michel

Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, editors, *CANS 13*, volume 8257 of *LNCS*, pages 218–234. Springer, Heidelberg, November 2013.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, February 2007.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.

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

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, Heidelberg, May / June 2006.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622, 2014. http://eprint.iacr.org/2014/622.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[HO12] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 627–643. Springer, Heidelberg, May 2012.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May 2010.

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

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Heidelberg, April 2012.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012.

[Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014. http://eprint.iacr.org/2014/588.

A 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

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

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 A.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 A.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 $\mathbf{c} \in \operatorname{span}(\Lambda)$, then $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \geq \frac{1-\epsilon}{1+\epsilon} \cdot \rho_{\sqrt{\Sigma}}(\Lambda)$.

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 A.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 A.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,

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

B Security properties of Regev's PKE

ℓ-Public-Key Reproducibility. To show that Construction 7.2 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 B.1 Construction 7.2 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 $\sigma > \sigma'(1 + T\sqrt{\ell})$ because α_i is smaller than T 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 $\sigma > \sigma'(1+T\sqrt{\ell})$. 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 $\forall \mathbf{z}, \epsilon \leftarrow \chi_{\sqrt{\Sigma'}}$

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

▮

ℓ-Ciphertext Reproducibility. We show that Construction 7.2 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 B.2 Under the LWE assumption, Construction 7.2 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.

```
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 $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

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

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

```
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 $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

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

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

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.

```
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 $(\mathsf{pk}, (\alpha_i, \mathsf{pk}_i, \mathsf{sk}_i)_{i\in[\ell]}, \mathsf{ct}_0, (\mathsf{ct}_i)_{i\in[\ell]})$

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 $\mathsf{ct}_0 = \mathbf{A}^\top \mathbf{r}$.
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 $(\langle \mathbf{e}_1, \mathbf{r}\rangle, \dots, \langle \mathbf{e}_\ell, \mathbf{r}\rangle)$, where $\mathbf{e}_i \leftarrow \chi^m$, $\mathbf{r} \in \{0,1\}^m$.

Then, by Lemma A.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 A.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.

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 $(b' = b)$
```

These are exactly the same distribution as $\mathbf{r}$ is used nowhere else.

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 $(b' = b)$
```

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 A.3 gives us that the statistical distance between the two views is at most ϵ.

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

```
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 $(b' = b)$
```

Once again, the hardness of LWE guarantees that pk looks pseudo-random to the adversary.

▌