Feel the Quantum Functioning: Instantiating Generic Multi-Input Functional Encryption from Learning with Errors (extended version)[*]

Alexandros Bakas, Antonis Michalas, Eugene Frimpong and Reyhaneh Rabbaninejad

Tampere University of Technology, Tampere, Finland
{alexandros.bakas,antonios.michalas,eugene.frimpong,reyhaneh.rabbaninejad}@tuni.fi

Abstract. Functional Encryption (FE) allows users who hold a specific decryption key, to learn a specific function of encrypted data while the actual plaintexts remain private. While FE is still in its infancy, it is our strong belief that in the years to come, this remarkable cryptograhic primitive will have matured to a degree that will make it an integral part of access-control systems, especially cloud-based ones. To this end, we believe it is of great importance to not only provide theoretical and generic constructions but also concrete instantiations of FE schemes from well-studied cryptographic assumptions. Therefore, in this paper, we undertake the task of presenting two instantiations of the generic work presented in [8] from the Decisional Diffie-Hellman (DDH) problem that also satisfy the property of verifiable decryption. Moreover, we present a novel multi-input FE (MIFE) scheme, that can be instantiated from Regev's cryptosystem, and thus remains secure even against quantum adversaries. Finally, we provide a multi-party computation (MPC) protocol that allows our MIFE construction to be deployed in the multi-client model.

Keywords: Functional Encryption · Learning With Errors · Multi-Party Computation · Verifiable Decryption

1 Introduction

In contrast to traditional cryptographic techniques, Functional Encryption (FE) is an emerging cryptographic primitive enabling selective computations over encrypted data. FE has been described [11] as generalized public-key cryptography that offers modern encryption capabilities such as cryptographic access control through Attribute-Based Encryption (ABE) [23] where the decryption algorithm results vary according to the decryption key used each time. Each decryption key sk_f is connected to a function f. Unlike traditional public-key cryptography,

[*] This work was funded from the Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates, for the project ARROWSMITH: Living (Securely) on the edge.

the use of sk_f on a ciphertext $\mathsf{Enc}(x)$ does *not* recover x but a function $f(x)$. In this way, the actual value x remains private. A more recent work [19] has introduced the general and promising notion of multi-input FE (MIFE). Here, when ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$ are provided, sk_f can be used to recover $f(x_1, \ldots, x_n)$. MIFE appears to be a perfect match for real-life applications, particularly cloud-based, as multiple users store large volumes of data in remote and possibly corrupted entities. However, the majority of research in the topic revolves around building *generic* schemes that do not support specific functions except for sums [8], inner-products [1–3] and quadratic polynomials [24]. Though practically FE could lead to innovative, hands-on and creative applications, it still fails to meet those goals. We are persuaded that FE is of great importance and that only a group of modern encryption schemes can lead us into an uncharted technological terrain. We hence attempted to narrow inconsistencies between theory and practice.

Functional Encryption in Access Control FE is a cryptographic primitive providing high-level access control and selective computations on encrypted data. In traditional cryptography access to encrypted data is *all-or-nothing*. With FE there is partial access to a function of the encrypted data and hence, limited access to actual plaintexts. Medical records are typical examples that require cryptographic access control, where the release of sensitive data involves ecnrypted statistical databases and data mining. Additionally, the computed function does not necessarily have to be a strict mathematical function. We could also consider a scenario in which decrypting encrypted images with a cropping key, results in a cropped version of the original image and nothing else. We are positive that in the years to come, FE will play a significant role in access control systems, especially cloud-based.

Quantum-Secure Constructions Along with the numerous advantages that technological evolution has to offer, it is no secret that we are slowly moving towards quantum age, where quantum computers will eventually replace today's digital, machines. Quantum computers should not be seen simply as *more powerful supercomputers* since they represent a whole new paradigm in computing. In particular, quantum computers are, at least in theory, capable of performing computations not attainable by traditional computer, no matter their processing power. To this end, it is important to start replacing quantum-vulnerable mathematical problems, used in traditional public-key encryption, with mathematical problems that are believed to be intractable from both classical and quantum machines. As a result, over the past years, researchers have begun showing great interest in designing encryption schemes based on such problems. This is particularly evident in the case of Homomorphic Encryption [18], where state-of-the-art constructions rely on the hardness assumption of the *"Learning with Errors"* (LWE) problem and its variation over rings of polynomials in finite fields (RLWE) [12, 14]. With this in mind, in this paper we present a detailed construction of a MIFE scheme that can be instantiated from a public-key encryption scheme that remains secure under the LWE hardness assumption.

1.1 Contributions

Our contribution can be summarized as follows:

C1: The first contribution of this paper is an instantiation of the MIFE scheme presented in [8] from the Decisional Diffie-Hellman Problem (DDH). Our instantiation is based on a variation of the ElGamal cryptosystem [15], called additively homomorphic ElGamal, whose IND-CPA security is proven under the hardness assumption of DDH.

C2: We prove that our instantiation from DDH satisfies the property of verifiable decryption in a zero-knowledge fashion. In particular, using our DDH instantiation, a user can verify that the functional decryption was honestly computed without having access to the decryption key. This is extremely important in cases where the decryptor is an untrusted third party such as a Cloud Service Provider (CSP). Satisfying the property of verifiable decryption in FE, is an important step towards assuming stronger threat models, by removing trust from, traditionally, fully trusted entities.

C3: The main contribution of our work is a modified version of the generic scheme from [8] that can be instantiated from LWE. More precisely, similarly to [8], the main building block of our modified scheme is an IND-CPA secure public-key encryption scheme PKE. However, our work shows that Regev's cryptosystem can be used as the PKE. While this modification may seem like a trivial extension, it needs to be treated carefully, because, in contrast to [8], where the multiplication of the public keys (and the ciphertexts) is feasible, in Regev's cryptosystem the public keys are non-squared matrices of equal dimensions, therefore multiplication is not possible. To overcome this challenge, we provide novel definitions and properties in the place of *Linear Key Homomorphism* (LKH) and *Linear Ciphertext Homomorphism* (LCH) used in [8]. Finally, we present a formal proof of security for our novel construction.

C4: We also present a secure Multi-Party Computation (MPC) that can be used as a generic compiler that turns a single-client MIFE, whose functional decryption key is formed as a linear combination of n different secret keys, to a multi-client one. The idea is that instead of having one user generating n public/private key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$, to distribute the process to n different users where each user generates a unique $(\mathsf{pk}_i, \mathsf{sk}_i)$. Then, using our MPC construction, users cooperate with each other and send a masked version of the functional decryption key to an untrusted third party, responsible for executing functional decryption. Upon reception the untrusted party can recover the functional decryption key without learning anything about each individual sk_i, assuming that at least two users are honest.

1.2 Organization

The rest of the paper is organized as follows: In section 2, we discuss important published works in the field of FE and MIFE. In section 3, we present all the important background required for this work. Moreover, we briefly recall Regev's LWE-based cryptosystem on which we base our quantum-secure MIFE construction. In section 4, we present the generic construction from [8] for which we give an instantiation from DDH in section 5, and we further prove that our instantiation satisfies the important property of verifiable decryption. In section 6 we present a novel construction, based on [8], a formal proof of security, and an instantiation from the quantum-secure Regev cryptosystem. Subsequently, in section 7, we provide a generic MPC-based transformation that transforms our MIFE construction from the single-client to the multi-client model along with a proof of security. section 8 comprises of our experimental results and finally, section 9 concludes the paper.

2 Related Work

The most direct ancestor of FE is undoubtedly ABE. In [23], Sahai and Waters presented a ground-breaking idea according to which the data owner could express how they wish to share data directly in the encryption algorithm. In particular, the data owner would provide a predicate f[1] describing how the data will be shared. Moreover, each user would receive a list of credentials L. Then, decryption of a ciphertext encrypted with predicate f is possible if and only if $f(L) = 1$. Another *"relative"* of FE can be found in the field of Searchable Encryption (SE)(both in the symmetric and asymmetric cases) [5,6,10,16]. In an SE scheme, users can search directly on encrypted data for specific keywords, withouht decrypting them. Functional encryption was then formalized as a generalization of public-key encryption and [23] in [11]. Since then, numerous studies with general definitions and generic constructions of FE have been proposed [19,20,22,25]. Despite the promising works published, there is a clear lack of research proposing FE schemes to support specific functions. This step would be necessary to allow FE to transcend its limitations and provide the foundations to reach its full potential. To the best of our knowledge, currently the number of supported functionalities is limited to sums [4,7], inner products [1–3] and quadratic polynomials [24]. In this work, we fist present an instantiation from DDH for the generic scheme presented in [8] and then we propose a MIFE scheme for the sum of a vector's components that can be instantiated from LWE.

3 Preliminaries

In this section, we present the necessary notation and definitions needed to follow this paper. The section is divided into five parts: We start by describing the basic notations, then we give definitions about Public-Key Encryption, Functional Encryption, Homomorphic Encryption and differential privacy.

[1] In mathematical logic, a predicate is a a function that tests for some condition involving its arguments and returns 1 if the condition is true and 0 otherwise.

Notation If $\mathcal{Y}$ is a set, we use $y \xleftarrow{\$} \mathcal{Y}$ if y is chosen uniformly at random from $\mathcal{Y}$. The cardinality of a set $\mathcal{Y}$ is denoted by $|\mathcal{Y}|$. For a positive integer m, $[m]$ denotes the set $\{1, \ldots, m\}$. Vectors are denoted in bold as $\mathbf{x} = [x_1, \ldots, x_n]$. A PPT adversary $\mathcal{ADV}$ is a randomized algorithm for which there exists a polynomial $p(z)$ such that for all input z, the running time of $\mathcal{ADV}(z)$ is bounded by $p(|z|)$. A function $negl(\cdot)$ is called negligible if $\forall\, c \in \mathbb{N}, \exists\, \epsilon_0 \in \mathbb{N}$ such that $\forall\, \epsilon \geq \epsilon_0$: $negl(\epsilon) < \epsilon^{-c}$.

Definition 1 (Inner Product). *The inner product (or dot product) of* $\mathbb{Z}^n$*, for two vectors* $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ *is a function* $\langle \cdot, \cdot \rangle$ *defined by:*

$$f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + \cdots + x_n y_n$$

Definition 2 (ℓ_2 norm). *The* ℓ_2 *norm of* $\mathbb{Z}^n$ *for a vector* $\mathbf{x} \in \mathbb{Z}^n$ *is a function* $\|\cdot\|_2$ *defined by:*

$$f(x) \ = \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{i=n} x_i^2}$$

3.1 Public-Key Encryption

Definition 3 (Public-Key Encryption scheme). *A public-key encryption scheme* PKE *for a message space* $\mathcal{M}$*, consists of three algorithms* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*. A* PKE *scheme is said to be correct if:*

$$Pr[\mathsf{Dec}(\mathsf{sk}, c) \neq m \mid [(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setip}(1^\lambda)] \wedge [m \in \mathcal{M}] \wedge [c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)]] = negl(\lambda)$$

To formalize the security of a PKE scheme, we follow the IND-CPA paradigm.

Definition 4 (Indistinguishability-Based Security). *Let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. We define the following experiments:*

$Exp^{s-IND-CPA-\beta}(\mathcal{ADV})$

```
Initialize(λ, x_0, x_1)                    Finalize(β')
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$      $\beta' = \beta$
    Return pk
Challenge()
    $c_\beta \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_\beta)$
```

The advantage ϵ *of* $\mathcal{ADV}$ *is defined as:*

$$\epsilon = \Big| Pr[Exp^{s-ind-CPA-0}(\mathcal{ADV}) = 1 \\ - Pr[Exp^{s-ind-CPA-1}(\mathcal{ADV}) = 1] \Big|$$

We say that PKE *is s-IND-CPA-*β *secure if*

$$\epsilon = negl(\lambda)$$

Definition 5 (Linear Ciphertext Homomorphism (LCH)). *We say that a* PKE *scheme has linear ciphertext homomorphism if:*

$$\prod_{i\;=1}^{n} \mathsf{Enc}(\mathsf{pk}_i, x_i) = \mathsf{Enc}\left(\prod_{i=1}^{n} \mathsf{pk_i}, \sum_{i=1}^{n} x_i\right)$$

Definition 6 (Linear Key Homomorphism (LKH)). *Let* $(\mathsf{pk}_1, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_2, \mathsf{sk}_2)$ *be two public/private key pairs that have been generated using* PKE.Gen*. We say that* PKE *has linear key homomorphism if* $\mathsf{sk}_1 + \mathsf{sk}_2$ *is a private key to a public key computed as* $\mathsf{pk}_1 \cdot \mathsf{pk}_2$*.*

A direct result of definitions 5 and 6 is that if a PKE scheme is linear ciphertext and key homomorphic, then the public keys of PKE live in multiplicative group $\mathbb{G}_{pub} = (\mathbb{G}, \cdot, 1_{\mathbb{G}_{pub}})$ and the private keys in an additive group $\mathbb{H}_{priv} = (\mathbb{H}, +, 0_{\mathbb{H}_{priv}})$.

3.2 Multi-Input Functional Encryption

Definition 7 (Multi-Input Functional Encryption).
A Multi-Input Functional Encryption scheme MIFE for a message space $\mathcal{M}$ *is a tuple MIFE = (*Setup, Enc, KeyGen, Dec*) such that:*

- $\mathsf{Setup}(1^\lambda)$*: The* Setup *algorithm is a probabilistic algorithm that on input the security parameter* λ*, outputs a master public/private key pair* (mpk, msk)*.*
- Enc(mpk, x)*: The encryption algorithm* Enc *is a probabilistic algorithm that on input the master public key* mpk *and a message* $\mathbf{x} = \{x_1, \ldots, x_n\} \in \mathcal{M}$*, outputs a ciphertext* $\mathbf{c} = \{c_1, \ldots, c_n\}$*.*
- KeyGen(msk, f)*: The key generation algorithm* KeyGen *is a deterministic algorithm that on input the master secret key* msk *and a function* f*, outputs a functional key* sk_f*.*
- $\mathsf{Dec}(\mathsf{sk}_f, c)$*: The decryption algorithm* Dec *is a deterministic algorithm that on input a functional key* sk_f *and a ciphertext* $\mathbf{c}$*, outputs* $f(x_1, \ldots, x_n)$*.*

A MIFE scheme is said to be correct if:

$$\begin{aligned} &Pr[\mathsf{Dec}(\mathsf{sk}_f, \mathbf{c}) \neq f(\mathbf{x}) \mid [(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)] \\ &\quad \wedge [\mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x})] \wedge [\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)]] = negl(\lambda) \end{aligned}$$

Just like in the case of PKE we base our security definition on the selective-IND-CPA formalization:

Definition 8 (MIFE Indistinguishanility-Based Security).
For a MIFE scheme MIFE = (Setup, Enc, KeyGen, Dec) *we define the following experiments:*

$Exp^{s-IND-FE-CPA-\beta}(\mathcal{ADV})$

```
Initialize(λ, x_0, x_1)
$\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$L \leftarrow \emptyset$
Output $\mathsf{mpk}$
Key Generation(f)
$L \leftarrow L \cup \{f\}$
$\mathsf{sk}_f \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}, \mathsf{f})$
Output $\mathsf{sk}_f$

Challenge()
$\mathbf{c}_\beta \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_\beta)$
Finalize($\beta'$)
If $\exists\, f \in L$ :
    $f(\mathbf{x_0}) \neq f(\mathbf{x_1})$
    Output $\perp$
Else
$\beta' = \beta$
```

The advantage ϵ of $\mathcal{ADV}$ is defined as:

$$\epsilon = \Big| Pr[Exp^{s-ind-FE-CPA-0}(\mathcal{ADV}) = 1$$
$$-Pr[Exp^{s-ind-FE-CPA-1}(\mathcal{ADV}) = 1] \Big|$$

We say that PKE *is s-IND-FE-CPA-β secure if*

$$\epsilon = negl(\lambda)$$

3.3 Background on LWE

For a real number $x \in \mathbb{R}$, we denote by $\lfloor x \rfloor$ the largest integer not greater than x and by $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ the integer closest to x with ties broken upward.

Definition 9 (Hermite Normal Form (HNF)). *Let $A \in \mathbb{Z}_q^{n \times m}$. Assume that the leftmost n columns $A_1 \in \mathbb{Z}_q^{n \times n}$ of $A = [A_1|A_2] \in \mathbb{Z}_q^{n \times m}$ form an invertible matrix over $\mathbb{Z}_q$. We can then replace A with its HNF as:*

$$A_1^{-1} \cdot A \ = [I_n|\overline{A} = A_1^{-1} \cdot A_2] \qquad (1)$$

Definition 10 (LWE Distribution). *Let $\mathcal{G}$ be a discrete Gaussian of width $\kappa \cdot q$ for some $\kappa < 1$ (error rate). For a vector $\mathbf{s} \in \mathbb{Z}_q^n$ called the secret, the LWE distribution $\mathcal{D}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow \mathcal{G}$ and outputting $(\mathbf{a}, \mathbf{b} = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod q)$.*

Definition 11 (Decision-LWE). *Given m independent samples $(\mathbf{a_i}, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where every sample is sampled either from $\mathcal{D}$ or from the uniform distribution, distinguish which is the case with non-negligible advantage.*

Regev's cryptosystem Regev's cryptosytem is parametrized by an LWE dimension n, a modulus q and an error distribution $\mathcal{G}$ over $\mathbb{Z}$. Note that public-keys are $\widetilde{O}(n^2)$ bits and ciphertexts $\widetilde{O}(n)$ bits where the $\widetilde{O}$ notation ignores logarithmic factors.

– The private key is a uniformly random LWE secret $s \in \mathbb{Z}^n$, and the public key is m samples $(\overline{\mathbf{a_i}}, b_i = \langle \mathbf{s}, \overline{\mathbf{a_i}} + e_i \rangle) \in \mathbb{Z}_q^{n+1}$ drawn from $\mathcal{D}$ and collected as the columns of a matrix

$$A = \begin{bmatrix} \overline{A} \\ b^T \end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m} \tag{2}$$

By definition, the private and the public keys satisfy the relation:

$$(-\mathbf{s}, 1)^T \cdot A = \mathbf{e}^T \approx 0(\mod q) \tag{3}$$

– To encrypt a bit $m \in \mathbb{Z}_2$ using the public key A, a user first chooses a uniformly random $\mathbf{r} \in \{0,1\}^m$ and outputs:

$$\mathbf{c} = A \cdot \mathbf{r} + \left(0, m \cdot \left\lfloor \frac{q}{2} \right\rceil\right) \in \mathbb{Z}_q^{n+1} \tag{4}$$

– To decrypt a ciphertext using a secret key $\mathbf{s}$ the user computes:

$$\begin{aligned} (-\mathbf{s}, 1)^T \cdot \mathbf{c} &= (-\mathbf{s}, 1)^T \cdot A \cdot r + m \cdot \left\lfloor \frac{q}{2} \right\rceil \\ &= \mathbf{e}^T \cdot r + m \cdot \left\lfloor \frac{q}{2} \right\rceil \\ &\approx m \cdot \left\lfloor \frac{q}{2} \right\rceil (\mod q) \end{aligned} \tag{5}$$

where the relations holds because $\mathbf{s}, \mathbf{e}^T$ and $\mathbf{r}$ are small. Upon calculating $m \cdot \left\lfloor \frac{q}{2} \right\rceil$ the user simply checks whether the result is close to 0 or 1.

In [21] Regev proved that his cryptosystem is semantically secure, assuming that the decision LWE is hard. More specifically, Regev proved that the hardness of decision LWE is implied by the worst-case quantum hardness of lattice problems.

4 Base Construction

In this Section, we briefly recall the generic construction from [8].

Construction Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA secure cryptosystem, that also fulfils the LCH and LKE properties. Then define MIFE as $\mathsf{MIFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ where:

1. $\mathsf{Setup}(1^\lambda, n)$: The setup algorithm invokes the PKE's key generation algorithm Gen and generates n public/private key pairs as $(\mathsf{pk_1}, \mathsf{sk_1}), (\mathsf{pk_2}, \mathsf{sk_2}) \ldots, (\mathsf{pk_n}, \mathsf{sk_n})$. The public keys are then used to create and output a master public/private key pair $(\mathsf{mpk}, \mathsf{msk})$, where $\mathsf{mpk} = (\mathsf{params}, \mathsf{pk_1}, \ldots, \mathsf{pk_n})$ and $\mathsf{msk} = (\mathsf{sk_1}, \ldots, \mathsf{sk_n})$[2].

[2] The public parameters params depend on the choice of the PKE scheme.

2. $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$: The encryption algorithm Enc, takes as input the master public key mpk and a vector $\mathbf{x}$ and outputs $\mathbf{c} = \{c_1, \ldots, c_n\}$, where $c_i = \mathsf{Enc}(\mathsf{pk_i}, x_i)$.
3. $\mathsf{KeyGen}(\mathsf{msk})$: The key generation algorithm, takes as input the master secret key msk and outputs a functional key sk as $\mathsf{sk} = \sum_1^n \mathsf{sk_i}$[3].
4. $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: The decryption algorithm takes as input the functional key sk and an encrypted vector $\mathbf{c}$ and outputs $\mathsf{PKE.Dec}(\mathsf{sk}, \prod_{i=1}^{n} \mathbf{c})$.

Correctness Correctness follows directly since:

$$\begin{aligned}\mathsf{MIFE.Dec}\left(\mathsf{sk}, \mathbf{c}\right) &= \mathsf{PKE.Dec}\left(\mathsf{sk}, \prod_{i=1}^{n} \mathsf{PKE.Enc}(\mathsf{pk_i}, x_i)\right) \\ &= \mathsf{PKE.Dec}\left(\mathsf{sk}, \mathsf{PKE.Enc}(\prod_{i=1}^{n} \mathsf{pk_i}, \sum_{i=1}^{n} x_i)\right) = \sum_{i=1}^{n} x_i\end{aligned}$$

where we used the LCH property. Since the LKE property holds, we know that sk is a valid secret key that decrypts $\prod_{i=1}^{n} \mathbf{c}$.

Theorem 1. *Let* PKE *be an IND-CPA secure public key cryptosystem that is additive key and additive ciphertext homomorphic. Moreover, let* MIFE *be the base Multi-Input Functional Encryption construction from [8] which is obtained though* PKE*. Then* MIFE *is s-IND-FE-CPA secure.*

A detailed proof can be found in [8].

5 Instantiation from DDH

We are now ready to present an instantiation of the generic construction from [8] from DDH. In particular, we will present an instantiation using the Additively Homomorphic El Gamal cryptosystem as the public-key encryption scheme PKE. For the needs of our proof, we rely on the fact that El Gamal remains secure under randomness reuse, as proven in [9].

Theorem 2. *Let* MIFE *be the generic construction from Section 4. Then* MIFE *can be instantiated from El Gamal's cryptosystem.*

Proof. It suffices to prove that El Gamal satisfies the LCH and LKH properties defined in definitions 5 and 6 respectively.

Let q be a prime and $\mathbb{G}$ a group of order q where the DHH assumption is hard. Moreover, let g be a generator of $\mathbb{G}$. Then we have that the private key

[3] We omit the description of the function since in this case we are only focusing on the sum

space is the group $(\mathbb{Z}_q, +, 0_{\mathbb{Z}})$ while the public key space is the group $(\mathbb{G}, \times, 1_{\mathbb{G}})$. Then, an El Gamal ciphertext for a message x is:

$$c = (g^r, \mathsf{pk}^r \cdot g^x)$$

where r is a random value used to ensure that the encryption algorithm is probabilistic.

– **LCH:** If Enc is the Encryption algorithm of El Gamal, then we have:

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

– **LKH** In the El Gamal cryptosystem we have that for a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ the following condition holds:

$$\mathsf{pk} = g^{\mathsf{sk}}$$

Let $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2)$ be two public/private key pairs for an El Gamal instantiation such that $\mathsf{pk}_1, \mathsf{pk}_2 \in (\mathbb{G}, \cdot, 0_{\mathbb{G}})$ and $\mathsf{sk}_1, \mathsf{sk}_2 \in (\mathbb{Z}, +, 0_{\mathbb{Z}})$. Then we have:

$$\mathsf{pk}_1 \cdot \mathsf{pk}_2 = g^{\mathsf{sk}_1} \cdot g^{\mathsf{sk}_2} g^{(\mathsf{sk}_1+\mathsf{sk}_2)}$$

Moreover, since the groups $(G, \cdot, 0_G)$ and $(Z, +, 0_{\mathbb{Z}})$ are closed with respect to multiplication and addition operations respectively, we conclude that $(\mathsf{pk}_1\mathsf{pk}_2, \mathsf{sk}_1 + \mathsf{sk}_2)$ is a valid public/private key pair.

5.1 Verifiable Decryption

As already mentioned, instantiating the generic MIFE construction from Section 4 using DDH, allows users to verify the decryption result in an zero-knowledge manner. This is extremely important as it allows us to considers a stronger threat models. In particular, assuming a malicious curator of a cloud database colludes with the CSP, could result to publishing modified statistics in an attempt to mislead data analysts. Hence, we show that an analyst that only possess a function $f(\mathbf{x})$ along with the public parameters of the encryption scheme, can verify that $f(\mathbf{x})$ is indeed the decryption of $\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x} = x_1, \ldots, x_n))$ under the function f, without having access neither to the master secret key, nor the functional decryption key for the underlying function. This is done by simply calculating and verifying the equality of two discrete logarithms. More precisely, and given that the final ElGamal ciphertext is given by:

$$(u, v) = (g^r, \mathsf{pk}^r \cdot g^{f(x)}), \tag{6}$$

the analyst needs to verify that:

$$\log_{\mathsf{pk}}\left[\left(g^{f(x)}\right)^{-1}\cdot \mathsf{pk}^r\cdot g^{f(x)}\right]=\log_g(g^r) \tag{7}$$

as follows:

$$\log_{\mathsf{pk}}\left[\left(g^{f(x)}\right)^{-1}\cdot \mathsf{pk}^r\cdot g^{f(x)}\right]=\log_g(g^r)\Rightarrow \log_{\mathsf{pk}}\left[\cancel{\frac{1}{g^{f(x)}}\cdot g^{f(x)}}\cdot \mathsf{pk}^r\right]=\log_g(g^r)\Rightarrow$$
$$\log_{\mathsf{pk}}\left(\mathsf{pk}^r\right)=\log_g(g^r)\Rightarrow r=r \tag{8}$$

Indeed, it can be seen that if a malicious party tampers with the result $f(x)$ and replaces it with $f(x)'$, then the term $\left(g^{f(x)'}\right)^{-1}$ will not cancel out along with $g^{f(x)}$ and hence, the equality will not hold.

5.2 Instantiation from CL Framework

In the previous section, we presented an additively homomorphic instantiation which was obtained by encoding the message in the exponent of an ElGamal encryption $(g^r, \mathsf{pk}^r\cdot g^{f(x)})$ where g is a generator of a cyclic group $\mathbb{G}$. However, this instantiation only supports a limited (*logarithmic*) number of additions. The reason is that one has to recover $f(x)$ from $g^{f(x)}$ and since Discrete Logarithm (DL) problem must be intractable in $\mathbb{G}$, it is essential to limit the size of message in the exponent to ensure an efficient decryption.

To enable an *unbounded* number of additions modulo prime q from the ciphertexts, Castagnos and Laguillaumie proposed the CL framework [13]. They assume a group $\mathbb{G}$ where the Decisional Diffie-Hellman (DDH) assumption holds and there exists a subgroup $\mathbb{H}$ of $\mathbb{G}$ in which the DL problem is easy. In the introduced cryptosystem, the message space is $\mathbb{Z}_q$, where the prime q can be *scaled* to meet the application needs, as its size is independent of the security parameter.

5.2.1 Background on CL Framework

CL framework is based on a DDH group with an easy DL subgroup which is defined as a pair of algorithms $(\mathsf{Gen}, \mathsf{Solve})$ described as:

- $\mathsf{CL.Gen}(1^\lambda, q)$: this algorithm inputs security parameter λ and q as a μ-bit prime for $\lambda \le \mu$, and outputs public parameters as $\mathsf{pp} = (q, \widehat{\mathbb{G}}, \mathbb{G}, \mathbb{G}^q, \mathbb{H}, \widetilde{s}, g, h, g_q)$, where
  - $\widehat{\mathbb{G}}$ is a finite group of order $\widehat{n} = q\cdot\widehat{s}$, such that $gcd(q, \widehat{s}) = 1$. Besides, $\widetilde{s}$ is defined to be an upper bound for $\widehat{s}$ with the condition that the distribution of $\{g^r, r \xrightarrow{\$} \{0, \ldots \widetilde{s}\cdot q\}\}$ is computationally indistinguishable from uniform distribution on $\widehat{\mathbb{G}}$.
  - $\mathbb{G}$ is a cyclic group of order $n = q\cdot s$ with generator g, such that s divides $\widehat{s}$.

  - $\mathbb{G}^q := \{x^q | x \in \mathbb{G}\}$ is a cyclic group of order s with generator g_q.
  - $\mathbb{H}$ is a unique cyclic group of order q with generator h, where $g = g_q \cdot h$.
- $\mathsf{CL.Solve}(q, \mathsf{pp}, X)$: this is a deterministic polynomial time algorithm that solves the DL problem in subgroup $\mathbb{H}$:

$$\mathsf{Pr}\left[x = x' \;\middle|\; \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda, q), x \xleftarrow{\$} \mathbb{Z}_q, X \leftarrow h^x, x' \leftarrow \mathsf{Solve}(q, \mathsf{pp}, X)\right] = 1,$$

.

Definition 12 (Hard Subgroup Membership Problem (HSM)). *Let λ be a positive integer and q be a μ-bit prime for $\lambda \leq \mu$. Also* $(\mathsf{Gen}, \mathsf{Solve})$ *generates a DDH group with an easy DL subgroup. The advantage of a PPT adversary $\mathcal{A}$ is defined as:*

$$\epsilon = |\mathsf{Pr}\left[b = b' \;\middle|\; \begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Gen}(1^\lambda, q), x \xleftarrow{\$} \mathcal{D}, x' \xleftarrow{\$} \mathcal{D}_q \\ b \xleftarrow{\$} \{0,1\}, X_0 \leftarrow g^x, X_1 \leftarrow g_q^{x'} \\ b' \leftarrow \mathcal{A}(q, \mathsf{pp}, X_b, \mathsf{Solve}(.)) \end{array}\right] - 1/2|,$$

where distribution $\mathcal{D}$ is such that the distribution of $\{g^x, x \xleftarrow{\$} \mathcal{D}\}$ is computationally indistinguishable from uniform distribution on $\mathbb{G}$ (same for $\mathcal{D}_q$,$\{g_q^x, x \xleftarrow{\$} \mathcal{D}_q\}$, and $\mathbb{G}^q$). $HSM - CL$ assumption holds for $(\mathsf{Gen}, \mathsf{Solve})$*, if for all PPT adversay $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathsf{HSM-CL}} \leq negl(\lambda)$, where $negl(\lambda)$ is a negligible function.*

- $\mathsf{CL.KeyGen}(\mathsf{pp})$ this algorithm samples $\alpha \xleftarrow{\$} \mathcal{D}_q$ and computes g_q^α. It then returns the key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk} := \alpha$ and $\mathsf{pk} := g_q^\alpha$.
- $\mathsf{CL.Enc}(\mathsf{pk}, m)$ on input m from the message space $\mathbb{Z}_q$ and the public key, this algorithm samples randomness $r \xleftarrow{\$} \mathcal{D}_q$ and returns $(c_1, c_2) = (g_q^r, h^m \cdot \mathsf{pk}^r)$.
- $\mathsf{CL.Dec}(\mathsf{sk}, (c_1, c_2))$ this algorithm computes $M \leftarrow c_2 / c_1^{\mathsf{sk}}$. It then runs $m \leftarrow \mathsf{Solve}(q, \mathsf{pp}, M)$, and returns m.

The CL PKE scheme described above is proven to be semantically secure under chosen plaintext attacks (IND-CPA) under the $HSM - CL$ assumption [13].

Theorem 3. *Let* MIFE *be the generic construction from Section 4. Then* MIFE *can be instantiated from* CL.PKE *scheme.*

Proof. Similarly to our initial naive instantiation from DDH, we prove that CL satisfies the LCH and LKH properties.

Let λ be a positive integer and q be a μ-bit prime for $\lambda \leq \mu$. Also $(\mathsf{Gen}, \mathsf{Solve})$ generates a DDH group with an easy DL subgroup where the $HSM - CL$ assumption holds. Moreover, let g_q, h be generators of groups $\mathbb{G}^q, \mathbb{H}$, respectively.

- **LCH:** If $\mathsf{CL.Enc}$ is the encryption algorithm of CL cryptosystem, then we have:

$$\begin{aligned}\mathsf{CL.Enc}(\mathsf{pk}_1, x_1) \cdot \mathsf{CL.Enc}(\mathsf{pk}_2, x_2) &= g_q^{r\mathsf{sk}_1} h^{x_1} \cdot g_q^{r\mathsf{sk}_2} h^{x_2} \\ &= g_q^{r(\mathsf{sk}_1+\mathsf{sk}_2)} \cdot h^{x_1+x_2} \\ &= \mathsf{CL.Enc}(\mathsf{pk}_1\mathsf{pk}_2, x_1 + x_2).\end{aligned}$$

- **LKH** In the CL cryptosystem we have that for a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ the following condition holds:

$$\mathsf{pk} = g_q^{\mathsf{sk}}$$

  Let $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2)$ be two public/private key pairs for a CL instantiation such that $\mathsf{pk}_1, \mathsf{pk}_2 \in (\mathbb{G}^q, \cdot, 1_{\mathbb{G}^q})$ and $\mathsf{sk}_1, \mathsf{sk}_2 \in (\mathcal{D}_q, +, 0_{\mathcal{D}_q})$. Then we have:

$$\mathsf{pk}_1 \cdot \mathsf{pk}_2 = g_q^{\mathsf{sk}_1} \cdot g_q^{\mathsf{sk}_2} = g_q^{(\mathsf{sk}_1+\mathsf{sk}_2)}$$

  Since the groups $(\mathbb{G}^q, \times, 1_{\mathbb{G}^q})$ and $(\mathcal{D}_q, +, 0_{\mathcal{D}_q})$ are closed with respect to multiplication and addition operations respectively, we conclude that $(\mathsf{pk}_1 \cdot \mathsf{pk}_2, \mathsf{sk}_1 + \mathsf{sk}_2)$ is a valid public/private key pair.

5.2.2 Verifiable Decryption Similarly to our previous instantiation, an analyst can verify the correctness of the computations (i.e. verify that $f(\mathbf{x})$ is the correct decrypted plaintext) by checking if the following equality holds:

$$e(u \cdot h^{f(x)}, \mathsf{pk}) \stackrel{?}{=} e(v, g_q) \tag{9}$$

Where $e(\cdot, \cdot)$ is a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ over multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$. The verification works as follows:

$$\begin{aligned}&e(u \cdot h^{f(x)}, \mathsf{pk}) \\ &= e(g_q^r \cdot h^{f(x)}, g_q^{\mathsf{sk}}) \\ &= e(g_q^{r\cdot\mathsf{sk}} \cdot h^{f(x)}, g_q) \\ &= e(\mathsf{pk}^r \cdot h^{f(x)}, g_q) \\ &= e(v, g_q)\end{aligned}$$

More information on bilinear maps can be found in [17].

6 MIFE for sums

We will now present a modified version of the base construction from Section 4 that can be instantiated from LWE. In particular, our construction will use Regev's cryptosystem as the public-key encryption scheme PKE. For the purposes of our version we first need to define new properties in the place of definitions 5 and 6. The reason for this is that in Regev's cryptosystem, the public keys (and the ciphertexts) are non-squared matrices of equal dimensions and hence, we cannot define multiplication between two public keys or ciphertexts respectively.

Definition 13 (Additive ciphertext homomorphism (ACH)). *We say that a* PKE *scheme has additive ciphertext homomorphism if:*

$$\sum_{i\ =1}^{n} \mathsf{Enc}(\mathsf{pk}_i, x_i) = \mathsf{Enc}\left(\sum_{i=1}^{n} \mathsf{pk_i}, \sum_{i=1}^{n} x_i\right)$$

Definition 14 (Additive Key Homomorphism (AKH)). *Let* $(\mathsf{pk}_1, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_2, \mathsf{sk}_2)$ *be two public/private key pairs that have been generated during* PKE.Gen. *We say that* PKE *has additive key homomorphism if* $\mathsf{sk}_1 + \mathsf{sk}_2$ *is a private key to a public key computed as* $\mathsf{pk}_1 + \mathsf{pk}_2$.

Definition 15 (Modified MIFE for the summation of a vector's components (mMIFE)). *Let* PKE = (Gen, Enc, Dec) *be an IND-CPA secure cryptosystem, that also fulfils the ACH and AKE properties. Then we define our* mMIFE *as* mMIFE = (Setup, Enc, KeyGen, Dec) *where:*

1. $\mathsf{Setup}(1^\lambda, n)$*: The* Setup *algorithm invokes the* PKE*'s* Gen *algorithm and generates* n *public/private key pairs as* $(\mathsf{pk_1}, \mathsf{sk_1}), \ldots, (\mathsf{pk_n}, \mathsf{sk_n})$. *It outputs a master public/private key pair as* mpk, msk, *where* $\mathsf{mpk} = (\mathsf{params}, \mathsf{pk_1}, \ldots, \mathsf{pk_n})$ *and* $\mathsf{msk} = (\mathsf{sk_1}, \ldots, \mathsf{sk_n})$.
2. $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$*: The Encryption algorithm* Enc*, takes as input the master public key* mpk *and a vector* $\mathbf{x}$ *and outputs* $\mathbf{c} = \{c_1, \ldots, c_n\}$, *where* $c_i = \mathsf{Enc}(\mathsf{pk_i}, x_i)$.
3. $\mathsf{KeyGen}(\mathsf{msk})$*: The Key Generation algorithm, takes as input the master secret key* msk *and outputs a functional key* sk *as* $\mathsf{sk} = \sum_1^n \mathsf{sk_i}$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$*: The Decryption Algorithm takes as input the functional key* sk *and an encrypted vector* $\mathbf{c}$ *and outputs* $\mathsf{PKE.Dec}(\mathsf{sk}, \sum_{i=1}^{n} \mathbf{c})$

Correctness The correctness of our construction follows directly since:

$$\mathsf{Dec}\left(\mathsf{sk}, \mathbf{c}\right) = \mathsf{PKE.Dec}\left(\mathsf{sk}, \sum_{i=1}^{n} \mathbf{c}\right) = \mathsf{PKE.Dec}\left(\mathsf{sk}, \sum_{i=1}^{n} \mathsf{PKE.Enc}(\mathsf{pk_i}, \mathsf{x_i})\right)$$
$$= \mathsf{PKE.Dec}\left(\mathsf{sksk}, \mathsf{PKE.Enc}\left(\sum_{i=1}^{n} \mathsf{pk_i}, \sum_{1}^{n} x_i\right)\right) = \sum_{1}^{n} x_i$$

Where we used the ACH property of PKE. Moreover, since the AKH property holds, we know that $\mathsf{sk}_{\|\cdot\|}$ is a valid secret key that decrypts $\sum\limits_{i=1}^{n} \mathbf{c}$.

Theorem 4. *Let* PKE *be an IND-CPA secure public key cryptosystem that is additive key and additive ciphertext homomorphic. Moreover, let* mMIFE_{ℓ_1} *be our modified Multi-Input Functional Encryption scheme for the* ℓ_1 *norm of a vector space which is obtained though* PKE*. Then* mMIFE_{ℓ_1} *is s-IND-FE-CPA secure.*

Proof Sketch: To prove Theorem 4 we will rely on a combination of the games presented in definitions 4 and 8. In particular, we assume two algorithms $\mathcal{A}$ and $\mathcal{B}$ are executed simultaneously but independently in which $\mathcal{B}$ is the adversary in game 4 and $\mathcal{A}$ is the both the adversary in game 8 and the challenger in game 8. Hence, $\mathcal{A}$ needs to simulate a perfect view of game 8. We prove that the advantage of $\mathcal{B}$ is bounded by the advantage of $\mathcal{A}$ and hence, if $\mathcal{B}$ wins then $\mathcal{A}$ also wins. However, this contradicts with the assumption that PKE is IND-CPA.

Proof. The proof begins with $\mathcal{B}$ sending $(0, z)$ to the challenger $\mathcal{C}$, where z is an element sampled at random from the message space of PKE. Upon receiving $(0, \mu)$, $\mathcal{C}$ generates a public/private key pair $(\mathsf{pk}_\mathcal{C}, \mathsf{sk}_\mathcal{C})$, flips a truly random coin b and encrypts either 0 or μ under $\mathsf{pk}_\mathcal{C}$ according to the result of the random coin to produce c_b. Finally, $\mathcal{C}$ forwards the pair $(\mathsf{pk}_\mathcal{C}, c_b)$ back to $\mathcal{B}$. Upon reception, $\mathcal{B}$ invokes $\mathcal{A}$ and as a result, receives two messages $\mathbf{x_0}$ and $\mathbf{x_1}$ such that $\|x_0\|_1 = \|x_1\|_1$[4]. To make sure that $\mathcal{B}$ only issues functional decryption keys queries for vectors such that $\|\mathbf{x_0}\|_1 = \|\mathbf{x_1}\|_1$, we impose the restriction that $\mathcal{B}$ only issues queries to a vector space $\mathcal{V} \subset \mathcal{M}$ of dimension n such that $\forall \mathbf{x} \in \mathcal{V}$, $\|\mathbf{x}\|_1 = 0$ and is not able to decrypt in other vector spaces. As a next step, $\mathcal{B}$ produces a basis of $\mathcal{V}$ as $(\mathbf{x_1} - \mathbf{x_0}, r_1, \ldots, r_{n-1})$.

Key Generation The first thing $\mathcal{B}$ needs to do, is to generate the master public key mpk. To do so, $\mathcal{B}$ samples $n-1$ linearly independent vectors $\mathbf{r_1}, \ldots, \mathbf{r_{n-1}}$ such that $\forall i \in [1, n-1] : r_i \in \mathcal{V}$ and each $\mathbf{r_i}$ is also linearly independent to $\mathbf{x_1} - \mathbf{x_0}$. The canonical vectors of the basis are then $\mathbf{e} = \Big[\boldsymbol{\alpha} \cdot (\mathbf{x_1} - \mathbf{x_0}) + \sum_1^{n-1} z_j\Big]$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\alpha_i = \frac{x_{1,i} - x_{0,i}}{\|x_{1,i-x_{0,i}}\|_2^2}$. Subsequently, $\mathcal{B}$ executes $(\mathsf{pk}_{z_j}, \mathsf{sk}_{s_j}) \leftarrow$ $\mathsf{PKE.Gen}, \forall j \in [1, n-1]$ and sets:

$$\mathsf{pk_i} = \alpha_i \cdot \mathsf{pk}_C + \sum_{j}^{n-1} \mathsf{pk}_{z_j} \quad \text{and} \quad \mathsf{mpk} = (\mathsf{pk_1}, \ldots, \mathsf{pk_n}). \tag{10}$$

Note that while $\mathsf{sk}_\mathcal{C}$ is not known to $\mathcal{B}$, due to the AKH property of PKE $\mathcal{B}$ is unknowingly setting $\mathsf{sk}_i = \alpha_i \cdot \mathsf{sk}_\mathcal{C} + \sum_1^{n-1} \mathsf{sk_{z_j}}$

[4] Note here that we abuse the notation of the ℓ_1 norm to denote the sum $\sum_1^n x_i$ where $\mathbf{x} = (x_1, \ldots, x_n)$

Functional Decryption Keys $\mathcal{B}$ receives queries for functional decryption keys from $\mathcal{A}$. To reply to such a query, all $\mathcal{B}$ has to do, is set $\mathsf{sk} = \sum_1^{n-1} \mathsf{sk}_{z_j}$.

Challenge Ciphertexts At some point $\mathcal{A}$ outputs two messages $\mathbf{x_0}$ and $\mathbf{x_1}$ such that $\|\mathbf{x_0}\|$ to $\mathcal{B}$. According to the game in the definition 8, $\mathcal{B}$ is supposed to flip a random coin $\beta \in \{0, 1\}$, and reply to $\mathcal{A}$ with c_β. However, recall that $\mathcal{B}$ not only needs to simulate a perfect view for $\mathcal{A}$, but also extract as much information as possible in order to win its own indistinguishability game of the public-key encryption scheme PKE. To do so, $\mathcal{B}$ flips the truly random coin β but instead of replying with c_β, sets the challenge ciphertext to be:

$$c = \alpha \cdot c_b + \mathsf{PKE.Enc}\left(\sum_{i=1}^{n-1} \mathsf{pk}_{z_j}, 0\right) + \mathsf{PKE.Enc}(0_{el}, \mathbf{x}_\beta) \tag{11}$$

where 0_{el} in equation 11 denotes the zero element of the space in which the public keys live.

Finally, $\mathcal{A}$ outputs a guess for β. If $\mathcal{A}$ correctly guess $\odot$, then $\mathcal{B}$ guesses that $\mathcal{C}$ encrypted 0. Otherwise, if $\mathcal{A}$ fails to guess β, then $\mathcal{B}$ guesses that $\mathcal{C}$ encrypted μ. For a clearer presentation, we distinguish between two cases based on $\mathcal{C}$'s choice.

C1: $\mathcal{C}$ encrypted 0: Assuming that $\mathcal{C}$ encrypted 0, then equation 11 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_{\mathcal{C}}, 0) + \mathsf{PKE.Enc}\left(\sum_{i=1}^{n-1} \mathsf{pk}_{z_j}, 0\right) + \mathsf{PKE.Enc}(0_{el}, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_{\mathcal{C}} + \sum_{i=1}^{n-1} \mathsf{pk}_{z_j} + 0_{el}, 0 + 0 + \mathbf{x}_\beta) = \mathsf{PKE.Enc}(\mathsf{pk}_i, \mathbf{x}_\beta) \end{aligned}$$

It is evident, that in this case,
mathcalB simulates a perfect view of the environment for $\mathcal{A}$, and hence, if $\mathcal{A}$ can guess β with advantage $\epsilon_{\mathcal{A}}$, then the advantage of $\mathcal{B}$, $\epsilon_{\mathcal{B}}$ in guessing that $\mathcal{C}$ wil be exaclty the same. Thus:

$$\epsilon_{\mathcal{A}} = \epsilon_{\mathcal{B}} \tag{12}$$

C2: $\mathcal{C}$ encrypted μ: Following the same procedure as in the previous case, if $\mathcal{C}$ encrypted μ instead of 0, then the challenge ciphertext from equation 11 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_{\mathcal{C}}, \alpha \cdot \mu) + \mathsf{PKE.Enc}\left(\sum_{i=1}^{n-1} \mathsf{pk}_{z_j}, 0\right) + \mathsf{PKE.Enc}(0_{el}, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_{\mathcal{C}} + \sum_{i=1}^{n-1} \mathsf{pk}_{z_j} + 0_{el}, \alpha \cdot \mu + 0 + \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\mathsf{pk}_i, \alpha \cdot \mu + \mathbf{x}_\beta) = \mathsf{PKE.Enc}(\mathsf{pk}_i, \mathbf{x}') \end{aligned}$$

However, recall that α is defined as: $\alpha = \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}$

Hence, $\mathbf{x}'$ is:

$$\begin{aligned}\mathbf{x}' = \mathbf{x}_\beta + \alpha \cdot \mu &= \frac{\mu}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}(\mathbf{x}_1 - \mathbf{x}_0) + \mathbf{x}_\beta \\ &= \frac{\mu}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}(\mathbf{x}_1 - \mathbf{x}_0) + \mathbf{x}_0 + \beta(\mathbf{x}_1 - \mathbf{x}_0)\end{aligned}$$

If we now set $v = \frac{\mu}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2} + \beta$, we see that the challenge message $\mathbf{x}'$ becomes:

$$\mathbf{x}' = v \cdot \mathbf{x}_1 + (1 - v)\mathbf{x}_0 \tag{13}$$

which is exactly the message that corresponds the the challenge ciphertext. Note that $\mathbf{x}' \in V$ since it is a linear combination of elements that live in V and whose coefficients sum up to one. Hence, $\mathbf{x}'$ is well defined. Finally, β is information theoretically hidden as the distributions of u is independent of β. Hence, in this case we have that:

$$\epsilon_{\mathcal{B}} = 0 \tag{14}$$

Combining equations 12 and 14 we end up with $\epsilon_{\mathcal{B}} = \epsilon_{\mathcal{A}}$. Hence, the best advantage one can get against the CPA security of our construction presented in definition 15, is bounded by the best advantage one can get against the IND-CPA security of the public key encryption scheme PKE. In other words, we proved that if $\mathcal{A}$ breaks our MIFE construction, then there exists a PPT algorithm $\mathcal{B}$ that that wins the IND-CPA game of PKE and hence, PKE cannot be IND-CPA secure, which contradicts with our initial assumption that PKE is IND-CPA secure. □

Functional Keys for Vectors in Different Vector Spaces: As mentioned, $\mathcal{A}$ is only allowed to request functional keys for vectors living in a vector space $V \subset M$, where $\forall \mathbf{x} \in V : \|\mathbf{x}\|_1 = 0$. Notice that by allowing $\mathcal{A}$ to obtain functional decryption keys for vectors $x \notin V$, our scheme can be trivially broken. However, this would imply that $\mathcal{B}$ can generate such functional decryption keys, which is *impossible* since $\mathcal{B}$ does not know $\mathsf{sk}_{\mathcal{C}}$. Hence, the generated functional keys can only decrypt ciphertexts whose plaintexts are elements of V. This is a valid assumption since otherwise, we would demand security in a scenario where the master secret key is known to the adversary.

6.1 Instantiation from LWE

What remains to be done to show that our novel construction can be instantiated from LWE is proove the following theorem:

Theorem 5. *Let* mMIFE *be our modified construction for the summation of a vector's components. Then our construction can be instantiated using Regev's cryptosystem.*

Proof. To prove our theorem, it suffices to show that Regev's cryptosystem is both additive-ciphertext and key homomorphic. Just like in the case of the DDH instantiation, we rely on the fact that the randomness is shared between users, and hence, two users can use the same randomness to produce their ciphertexts.

ACH A ciphertext from Regev's cryptosystem is of the form $c = Ar + (0, x \cdot \lfloor \frac{q}{2} \rceil)$. For visual clarification, we can write the above relation as $c = Ar + g(x)$, where $g(x) = x \cdot (\lfloor \frac{q}{2} \rceil)$. Hence, for two ciphertexts $\mathbf{c_1}, \mathbf{c_2}$ we get:

$$\mathbf{c}_1 + \mathbf{c}_2 = A_1 r + g(x_1) + A_2 r + g(x_2) = (A_1 + A_2) r + g(x_1 + x_2) \tag{15}$$

And thus, Regev's cryptosystem is additive ciphertext homomorphic.

AKH The secret keys on Regev's cryptosystem are samples uniformly from $\mathbb{Z}_q^n$. Since $\mathbb{Z}_q^n$ is closed under addition, we know that $\sum_1^n \mathsf{sk}_i = \sum_1^n \mathbf{s}_i \in \mathbb{Z}_q^n$. What remains to be done, is to show that $\sum_1^n \mathsf{sk}_i$ is a valid private key for a public key of the form $\sum_1^n (A\mathsf{sk_i} + \mathbf{e}_i)$, which is true as long as $\sum_1^n \mathbf{e}_i$ remains small. This can be seen from the fact that a private/public key pair needs to satisfy the following:

$$\left(-\sum_1^n \mathsf{sk}_i, 1 \right)^T \cdot A = \sum_1^n \mathbf{e}^T \approx 0(\mod q) \tag{16}$$

Hence, we see that Regev's cryptosystem satisfies both the ACH and AKH properties and as a result, it can be used to instantiate our modified construction mMIFE.

Security of the instantiation A direct result of theorems 4 and 5 is that by instantiating our construction using Regev's cryptosystem, then our scheme is quantum-secure.

7 From Single-Client to Multi-Client MIFE

In this section, we present a generic tranformation that tranforms our construction from being solely multi-input to also being multi-client. The main challenge in this setting is the generation of the functional decryption key. Our solution relies on an MPC in which all users input a masked version of their secret key sk_i. Then, they send the masked keys to a central authority, that outputs the functional decryption key as the sum of all sk_i's without learning anything about each individual sk_i. More precisely, the problem we are trying to solve is formally described below:

Probelm Statement 1 (MIFE$_{\ell_1}$ **with Multi-Client Support)** *Let* $\mathcal{U} = \{u_1, \ldots, u_n\}$ *be a set of users. Each user* $u_j \in \mathcal{U}$ *generates a public/private key pair* $(\mathsf{pk}_j, \mathsf{sk}_j)$ *for a public-key encryption scheme satisfying the properties defined in definitions 5 and 6, and uses* pk_j *to encrypt a message* x_j*. Additionally, assume that all generated ciphertexts are outsourced and stored in a remote location operated by an untrusted (i.e. possible malicious) CSP. Furthermore, we assume that an analyst* $\mathbf{A}$ *(e.g. a user from* $\mathcal{U}$*) wishes to perform statistics on the data stored on the CSP. Our multi-client construction shows how a legitimate analyst can do this without learning any valuable information about the individual values* x_j*.*

MPC Upon request of $\mathbf{A}$, each user $u_i \in \mathcal{U}$ generates a random number r_i and breaks it into n shares as $r_i = r_{i,1} + \cdots + r_{i,n}$. Each share will be sent to a different user from the set $\mathcal{U} = \{u_1, \ldots, u_n\}$. Upon receiving $n-1$ different shares, each user u_i mask her private key sk_i as $b_i = \mathsf{sk}_i + r_i - \sum_{j=1}^{n} r_{j,i}$, and sends the masked key to $\mathbf{A}$. When $\mathbf{A}$ has gathered all the masked keys, she computes the functional decryption key sk as $\mathsf{sk} = \sum_1^n \mathsf{b_i}$. The MPC is illustrated in figure 1.

```
MPC
$\mathbf{A}$ generates $r_A \xrightarrow{\$} \mathbb{Z}$
$\mathbf{A}$ writes $r_A$ as $r_A = r_{i,1} + \cdots + r_{i,n}$
for $j \in [1, n-1]$ do:
    Send $r_{i,j}$ to $u_j$
for $u_j \in \mathcal{U}/\{\mathbf{A}\}$ do:
    $u_j$ generates $r_j \xrightarrow{\$} \mathbb{Z}$
    Express $r_j$ as $r_j = r_{j,1} + \cdots + r_{j,n}$
    Send $r_{j_i}$ to $u_i, \forall u_i \in \mathcal{U}/\{\mathbf{A}\}$
    Compute masked version of the key as $b_j = \mathsf{sk}_j + r_j - \sum_{\ell=1}^{n} r_{k_i}$ after having received every share from the rest of the users
    Send $b_j$ to $\mathbf{A}$
$\mathbf{A}$ computes the functional decryption key as $\sum_1^n b_j = \sum_1^n \mathsf{sk}_j = \mathsf{sk}$
```

Fig. 1: Functional Decryption Key Generation in the Multi-Client Setting

It is important to highlight that splitting and distributing the random numbers to the different users, allows the users to work in parallel for the MPC and hence, we overcome the limitations that would emerge by using a ring topology.

Theorem 6. *Let $\mathcal{ADV}$ be an adversary that corrupts at most $n-2$ users out of those in $\mathcal{U}$. Then, $\mathcal{ADV}$ cannot infer any information about the secret keys of the legitimate users.*

Proof. Recall that each user receives $n-1$ shares from the remaining users. Assuming that $\mathcal{ADV}$ has colluded with $n-2$ users, we conclude that $\mathcal{ADV}$ will know the $n \cdot (n-2)$ shares of the compromised users. Moreover, $\mathcal{ADV}$ will also know the $n-4$ shares sent from the legitimate users u_l and u_ℓ to the compromised ones. In other words, $\mathcal{ADV}$ knows all the exchanged shares except from the ones that u_l and u_ℓ keep for themselves as well as the ones exchanged between u_l and u_ℓ. More specifically, the shares $r_{l,l}$ and $r_{\ell,\ell}$ are kept with u_l and u_ℓ respectively, while the shares $r_{\ell,l}$ and $r_{l,\ell}$ are exchanged between u_l and u_ℓ. We notice that:

$$s_l = \underline{\mathsf{sk}_l} + \underline{r_l} - (r_{1,l} + \cdots + \underline{r_{l,l}} + \cdots + \underline{r_{\ell,l}} + \cdots + r_{n,l}) \tag{17}$$

and

$$s_\ell = \underline{\mathsf{sk}_\ell} + \underline{r_\ell} - (r_{1,\ell} + \cdots + \underline{r_{l,\ell}} + \cdots + \underline{r_{\ell,\ell}} + \cdots + r_{n,\ell}) \tag{18}$$

Where the underlined terms are the ones that $\mathcal{ADV}$ does not know. Equations 17 and 18 can also be written as:

$$s_l = \underline{\mathsf{sk}_l} + \sum_{j \neq l,\ell}^{n} (r_{l,j} - r_{j,l}) + \underline{r_{l,\ell} - r_{l,\ell}} \quad \text{and} \quad s_\ell = \underline{\mathsf{sk}_\ell} + \sum_{j \neq \ell,l}^{n} (r_{\ell,j} - r_{j,\ell}) + \underline{r_{\ell,l} - r_{\ell,l}}$$

We see that for $\mathcal{ADV}$ to find the the secret keys sk_l and sk_ℓ, she needs to solve a system of two equations with four unknown terms. Hence, we conclude that even in the extreme scenario where $n-2$ users are corrupted, $\mathcal{ADV}$ cannot infer any information about the keys of the legitimate users. □

8 Experimental Results

In this section, we present a brief evaluation of an implementation of the instance of the generic construction described in Section 5. To this end, we implement the core functions of the construction on a standalone Linux machine, and measure the performance when applied to a real dataset. Our evaluations focused on the Setup, Encryption, and KeyGen functions. The experiments described in this section were conducted on an Intel Core i5-8279U CPU @ 2.40 GHz x2 Ubuntu 20.04 Desktop with 2GB RAM. Additionally, we utilized an Additive EC-Elgamal C library[5] to implement the basic cryptographic operations needed to implement the proposed construction. To closely mimic the multi-party nature of our construction,we perform our experiments on a real-world dataset obtained from the European Centre for Disease Prevention and Control[6], and conduct each experiment 50 times to find an average.

Setup Phase: Experiments in this phase focused on analyzing the processing time for generating a unique pair of keys for each row, and encrypting a specific chosen value on that row. In this instance, we chose the number of *deaths* as our value of interest. In total, the selected dataset has 61901 rows, however to provide a comprehensive evaluation of the setup process, we performed our experiments on a varying number of rows from 10000 to the maximum number of rows. The number of rows directly corresponds to the number of unique key pair and ciphertexts being computed. The processing time for generating 10,000 unique keypairs and computing 10,000 ciphertexts was 2.353 seconds while the processing time was 12.791 seconds for 60,000 ciphertexts (Figure 2).

Functional Key Generation: In this phase, we measured the execution time for computing the Functional Decryption key in the KeyGen function and time taken to compute the final ciphertext in the Encryption algorithm. As with the Setup function, we evaluated both functions for a varying number of keys and ciphertexts from 10,000 to 60,000 based on the unique keys and ciphertexts generated. For 10,000 keys, it took 0.226 seconds to compute the function decryption key (Figure 3a) and 0.415 seconds to generate the final ciphertext (Figure 3b). While for 60,000 keys and ciphertexts, it took 1.316 seconds and 2.379 seconds respectively (Figure 3a and Figure 3b).

[5] https://github.com/lubux/ecelgamal
[6] https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide

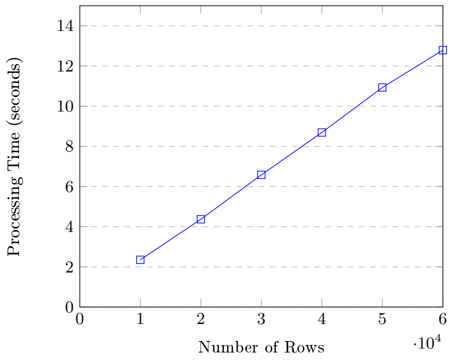
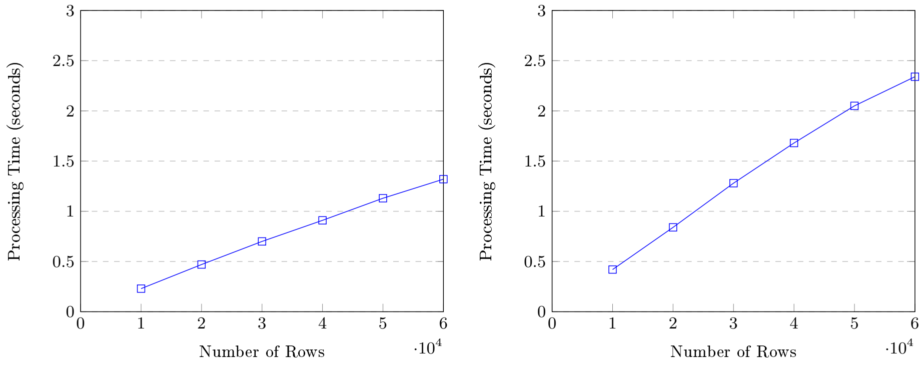


Fig. 2: Setup Phase



(a) KeyGen Function (b) Encryption Function

Science & Reproducible Research: To support open science and reproducible research, and provide other researchers with the opportunity to use, test, and hopefully extend our scheme, we have anonymized our source code and made it publicly available online[7].

9 Conclusion

The future will inevitably bring to the fore the need to exploit the power and functionality of a modern cryptographic technique such as FE. It is our firm belief that future access-control systems, especially cloud-based ones will rely less on traditional encryption and more on computations over encrypted data. To this end, it is of great significance, to start designing schemes that remain secure even against quantum attackers, an imminent threat closer than ever before.

[7] https://anonymous.4open.science/r/FeelQuantum-A4C4

References

1. Abdalla, M., , D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Advances in Cryptology – CRYPTO 2018 (2018)
2. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: IACR International Workshop on Public Key Cryptography. pp. 733–751. Springer (2015)
3. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer
4. Bakas, A., Michalas, A.: Multi-input functional encryption: efficient applications from symmetric primitives. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1105–1112. IEEE (2020)
5. Bakas, A., Michalas, A.: Power range: Forward private multi-client symmetric searchable encryption with range queries support. In: 2020 IEEE Symposium on Computers and Communications (ISCC). pp. 1–7. IEEE (2020)
6. Bakas, A., Michalas, A.: Nowhere to leak: A multi-client forward and backward private symmetric searchable encryption scheme. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 84–95. Springer (2021)
7. Bakas, A., Michalas, A., Ullah, A.: (f) unctional sifting: A privacy-preserving reputation system through multi-input functional encryption. In: Nordic Conference on Secure IT Systems. pp. 111–126. Springer (2020)
8. Bakas, A., Michalas, A., Dimitriou, T.: Private lives matter: A differential private functional encryption scheme. In: The 12th ACM Conference on Data and Application Security and Privacy (2022)
9. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemeas. In: International Workshop on Public Key Cryptography. pp. 85–99. Springer (2003)
10. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. pp. 506–522. Springer (2004)
11. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Theory of Cryptography Conference. pp. 253–273. Springer (2011)
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS '12 (2012)
13. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Cryptographers' Track at the RSA Conference. pp. 487–505. Springer (2015)
14. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 409–437. Springer International Publishing, Cham (2017)
15. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985)
16. Frimpong., E., Bakas., A., Dang., H., Michalas., A.: Do not tell me what i cannot do! (the constrained device shouted under the cover of the fog): Implementing symmetric searchable encryption on constrained devices. In: Proceedings of the 5th International Conference on Internet of Things, Big Data and Security - IoTBDS,. pp. 119–129. INSTICC, SciTePress (2020). https://doi.org/10.5220/0009413801190129

17. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (sep 2008). https://doi.org/10.1016/j.dam.2007.12.010, `https://doi.org/10.1016/j.dam.2007.12.010`
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 169–178. STOC '09, Association for Computing Machinery, New York, NY, USA (2009)
19. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 578–602. Springer (2014)
20. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Annual Cryptology Conference. pp. 536–553. Springer (2013)
21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6) (Sep 2009), `https://doi.org/10.1145/1568318.1568324`
22. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 463–472 (2010)
23. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. pp. 457–473. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
24. Sans, E.D., Gay, R., Pointcheval, D.: Reading in the dark: Classifying encrypted digits with functional encryption. IACR Cryptology ePrint Archive **2018**, 206 (2018)
25. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Annual Cryptology Conference. pp. 678–697. Springer (2015)