

Simulation Secure Multi-Input Quadratic Functional Encryption: Applications to Differential Privacy

Ferran Alborch Escobar^{1,2,3*†}, Sébastien Canard^{2†}
and Fabien Laguillaumie^{3†}

¹Applied Crypto Group, Orange Innovation, Caen, 14000, France.

²LTCI, Télécom Paris, Institut Polytechnique de Paris,
Palaiseau, 91120, France.

³LIRMM, Université de Montpellier, CNRS, Montpellier, 34095,
France.

*Corresponding author(s). E-mail(s): ferran.alborch@gmail.com;
Contributing authors: sebastien.canard@telecom-paris.fr;
fabien.laguillaumie@lirmm.fr;

†These authors contributed equally to this work.

Abstract

Multi-input functional encryption is a primitive that allows for the evaluation of an ℓ -ary function over multiple ciphertexts, without learning any information about the underlying plaintexts. This type of computation is useful in many cases where one has to compute over encrypted data, such as privacy-preserving cloud services, federated learning, or more generally delegation of computation from multiple clients. It has recently been shown by Alborch *et al.* in PETS '24 to be useful to construct a randomized functional encryption scheme for obtaining differentially private data analysis over an encrypted database supporting linear queries. In this work we propose the first secret-key multi-input quadratic functional encryption scheme satisfying simulation security. Current constructions supporting quadratic functionalities, proposed by Agrawal *et al.* in CRYPTO '21 and TCC '22, only reach indistinguishability-based security. Our proposed construction is generic, and for a concrete instantiation, we propose a new function-hiding inner-product functional encryption scheme proven simulation secure against one challenge ciphertext in the standard model, which is of independent interest.

We then use these two results to construct an efficient randomized quadratic functional encryption scheme, from which we obtain differentially private data analysis over an encrypted database supporting quadratic queries. Finally, we give and fully benchmark an implementation of the randomized scheme. This work is an extended version of the paper “Simulation Secure Multi-Input Quadratic Functional Encryption” at SAC ’24, where the multi-input quadratic functional encryption scheme and function-hiding inner-product functional encryption schemes were first presented (Section 3 and Section 4).

Keywords: Functional Encryption, Multi-input, Randomized, Quadratic Functions, Differential Privacy

1 Introduction

Functional encryption is a generalization of public key encryption first formalized in [1, 2], allowing for more control over the access on encrypted data. In general terms, this means that there is an authority who generates functional keys related to a function f . When combined with a ciphertext c_x of a plaintext x during the decryption process, $f(x)$ is recovered. The security guarantees that no other information about x is leaked apart from $f_i(x)$ for all functions f_i queried.

The versatility of this definition has allowed the concept to be applied to many different privacy preserving problems. For example, predicate encryption such as identity-based encryption [3] and attribute-based encryption [4] have been used to regulate fine-grained access of users to data. The way they achieve this is, at a high-level, by defining a function that outputs the message only if a certain predicate is verified. In the case of identity-based encryption this predicate is linked to the identity of the user, while in attribute-based encryption it is linked to some attributes of the user.

Another route taken with functional encryption for privacy purposes was introduced by Goyal *et al.* in [5], where they consider the case of outputs following a randomized function applied to the plaintext instead of a deterministic one. This is made with the objective to eventually mix functional encryption with differential privacy, another privacy enhancing technologies. More recently, Alborch *et al.* [6] and Zalonis *et al.* [7] studied the particular case of linear queries, for which they give efficient constructions. The objective of this work is to continue this line of work and give a solution for quadratic queries.

On the construction of functional encryption (FE) for some specific functionalities, Abdalla *et al.* in [8] gave the first instantiation for inner product (IPFE) from standard assumptions. Soon a plethora of so called IPFE satisfying different security models or based on different assumptions appeared [9, 10, 11]. Another set of works, first introduced in [12], refer to quadratic functions (QFE), for which papers in [13, 14, 15] propose instantiations in

the standard model. For higher degrees it is known that succinct degree-3 functional encryption implies indistinguishability obfuscation [16] as long as there exist pseudo-random generators of block-wise locality 3, but there are no known schemes based on polynomial assumptions.

The security of functional encryption has been known to be delicate to define from its initial formalization [1, 2]. This is due to the difficulty to incorporate the inherent leakage $f(x)$ into the security definition. There are two different approaches: *indistinguishability*-based security and *simulation*-based security. The first one requires for the adversary to be unable to distinguish between two plaintexts x_0, x_1 given a ciphertext of one of them and several functional keys sk_{f_i} . However, for this definition to make sense, it is required that $f_i(x_0) = f_i(x_1)$ is satisfied. This limits its applicability and makes it inadequate for some functionalities [1]. The second one requires the scheme to be indistinguishable from a simulator taking as inputs the inherent leakage from the scheme, in other words, the information released to the adversary through execution of the scheme like the output. It is known that this definition is strictly stronger than indistinguishability-based security. but also that there are some impossibility results [1, 2, 17]. In either case, there are also two cases that could be treated. If the adversary can only request functional keys after sending all the challenges, it is called *selective*, while if the adversary has no restriction on when it can request functional keys it is called *adaptive*. In this work we focus on selective simulation security against one challenge ciphertext, and multiple functional keys, which makes sense in our real-life use-case, where a database owner outsources their database by encrypting it and then an analyst can perform several differentially private queries over that encrypted data.

Multi-input functional encryption.

Multi-input functional encryption (MIFE) is a generalization of functional encryption first proposed by Goldwasser *et al.* in [18]. The main objective is to divide the plaintext into several parts (called inputs) so that they can be encrypted in independent executions of the encryption algorithm. In this case, the output of the decryption is a function taking all inputs as variables. This models a situation where data to be encrypted may not arrive all at the same time, while still be needed all together to obtain the evaluation of the desired function. This primitive is useful in many real-life use cases related to privacy-preserving cloud services, federated learning, or more generally delegation of computation to a more powerful entity.

Multi-input functional encryption for general purpose is difficult to achieve (as its single-input counterpart) and has strong implications, e.g., indistinguishability obfuscation [18]. However, for concrete families of functions some instantiations have been found. In the case of inner-product, the first proposal was by Abdalla *et al.* [19] with a secret-key multi-input scheme. It was followed by a transformation from single-input IPFE to the multi-input case [20] still in the secret-key setting that requires no other assumptions apart from the one of

the single-input scheme. For quadratic functionalities, only two proposals exist by Agrawal *et al.* [21, 22] in which they give specific instantiations of secret-key multi-input quadratic functional encryption schemes based on function-hiding functional encryption. Such constructions are proven secure in the indistinguishability setting and, as far as we know, it does not exist any construction for a multi-input quadratic functional encryption scheme satisfying simulation security. Our main purpose in this paper is to fill this gap.

The focus on secret key constructions comes from the fact that a public key multi-input functional encryption is easily constructed from a single-input functional encryption [19, 21] (for both linear and quadratic functionalities). However, these instantiations cannot be trivially transformed into secret key by considering the public key as part of the master secret key due to the inherent leakage of the functionalities. For a more detailed analysis of this issue, we refer to [19, Section 1.1] and [21, Section 1.2, Appendix A.2].

Function-hiding functional encryption.

As in [19, 21], our MIQFE is based on a function-hiding IPFE (FH-IPFE). Function-hiding is an additional security property for functional encryption, first proposed by Shen *et al.* in [23]. Analogously to the ciphertext protecting the plaintext x , the functional key is in this setting required to protect the function f . In other words, the adversary *only* learns $f(x)$ and no other information about either x or f . Once again, given the power the adversary has in the public-key setting to encrypt any message, public-key function-hiding schemes are unfeasible. By giving the adversary the possibility of encrypting any plaintext, the function “hidden” in the functional key can be recovered.

In the case of IPFE, several function-hiding constructions exist, most of the time based on bilinear pairings and secure in the standard model [24, 25, 26, 27] (especially because a recent result shows that Learning With Errors based constructions are impossible [28]). Security-wise, as explained above, we need simulation based security for our multi-input quadratic FE. Regarding the literature on the subject, it only remains the scheme by Kim *et al.* in [29] which achieves such security in the Generic Group Model.

Randomized functional encryption.

Randomized functional encryption (RFE) is a generalization of functional encryption first proposed by Goyal *et al.* in [5]. It considers functional encryption schemes whose output is not a deterministic function, but rather a probabilistic function over the plaintext. Importantly, the output distributions should be independent between different ciphertexts and different functional keys. This primitive models situations where the desired output is a distribution based on the data, rather than a deterministic value.

Generic transformations from deterministic to randomized functional encryption exist [30, 31] but require extra assumptions and result in inefficient constructions. As such, the main areas of interest are to give efficient instantiations for some concrete families of randomized functions, that incur in minimal

over cost with respect to deterministic schemes. One of the main applications of randomized functional encryption is to perform differentially private data analysis over encrypted data, with some recent constructions allowing linear queries [6, 7]. Our objective in this paper is to construct a randomized functional encryption scheme for quadratic functionalities.

Differential privacy.

Differential privacy (DP) is a private data mechanism property first proposed by Dwork *et al.* in [32]. The main objective is to release noisy data statistics in such a way that both the privacy and utility loss can be precisely calibrated. To do so, the distribution of the noisy statistics coming from two databases which differ in one individual are verified to be statistically close, with respect to some privacy parameters. This ensures that the information of any individual is protected, while information of the whole database at large remains useful and significant.

This concept and its usefulness have been greatly studied both by academics, as shown by the recent survey by Desfontaines and Péjo [33] where they compile over 200 different variants proposed in literature for different use-cases, as well as by the industry, where deployment of differential privacy includes the US Census Bureau [34] supporting analysis on travel patterns through their *OnTheMap* project [35], Google training next-word prediction models [36], or Microsoft collecting telemetry data privately [37] among many other.

Quadratic queries.

Quadratic queries in the context of statistical analysis are defined for some data points $\mathbf{x} = (x_1, \dots, x_n)$ as

$$q(\mathbf{x}) = \sum_{i,j \in [k]} a_{i,j} \cdot x_i \cdot x_j + \sum_{i=0}^k b_i \cdot x_i + c$$

for some $k > 1$ and fixed query weights $a_{i,j}, b_i, c$. Some notable examples are quadratic regressions, used for example in modeling chemical reactions in function of temperature, and χ^2 testing [38], a well-known hypothesis test where the quadratic form is constructed as follows. Let x_1, \dots, x_n be a set of observations and h_1, \dots, h_n be a null hypothesis an analyst wants to verify. We define the database $\mathbf{x} = (x_1, \dots, x_n, 1)$ and the quadratic function \mathbf{F}

$$\mathbf{F} = \begin{pmatrix} 1/h_1 & & & -2 \\ & \ddots & & -2 \\ & & 1/h_n & -2 \\ & & & \sum h_i \end{pmatrix}.$$

Then the evaluation $\mathbf{x}^\top \mathbf{F} \mathbf{x}$ outputs the χ^2 test of the null hypothesis h_1, \dots, h_n over the observations x_1, \dots, x_n , $\sum (x_i - h_i)^2 / h_i$.

In the context of differential privacy, private χ^2 testing has been studied both by itself [39] and in the broader context of private hypothesis testing [40].

1.1 Contributions

Our first contribution is the transformation from any function-hiding inner-product functional encryption to a multi-input quadratic functional encryption achieving selective simulation security for one ciphertext in the secret key setting, with a ciphertext size of $O(n\ell^2)$ where ℓ is the number of inputs and n the size of these inputs. This is the first instantiation of multi-input quadratic functional encryption scheme satisfying simulation security. To achieve this we rely on the single-input quadratic functional encryption scheme of [15, Section 3], using techniques from the multi-input inner-product functional encryption scheme from [20, Section 3].

This transformation is based on a simulation secure function-hiding inner-product functional encryption for which we give a new instantiation in the standard model, based on the DDH-based inner-product scheme in [8, Section 3] and inspired by the partially function-hiding inner-product scheme in [15, Section 4]. This is the first simulation secure function-hiding inner-product scheme in the standard model. These first two constructions were presented in SAC 2024 [41].

Finally, using our first contribution we construct an efficient secret-key randomized quadratic functional encryption satisfying selective simulation security against one challenge ciphertext. This is the first efficient scheme of this kind with minimal over cost with respect to deterministic quadratic functional encryption. This, due to a recent result in [6], allows us to construct a scheme to respond to differentially private quadratic queries. We use this result to construct a scheme to respond to private χ^2 tests over encrypted observations, for which we provide a differential privacy analyze, implement and fully benchmark.

1.2 State of the Art

In quadratic functional encryption, the function is generally defined by a matrix $\mathbf{F} \in \mathbb{Z}_p^{n \times n}$ with $\mathbf{F}(\mathbf{x}) = \mathbf{x}^\top \mathbf{F} \mathbf{x}$. Based on that, it is trivial to construct a “naive” single-input quadratic functional encryption scheme from any single-input inner-product functional encryption scheme, in which the functions are defined by a vector $\mathbf{y} \in \mathbb{Z}_p^n$ with $\mathbf{y}(\mathbf{x}) = \mathbf{x}^\top \mathbf{y}$. For such a generic construction, we first observe that $\mathbf{x}^\top \mathbf{F} \mathbf{x} = (\mathbf{x} \otimes \mathbf{x})^\top \text{vect}(\mathbf{F})$ where \otimes denotes the Kronecker product and $\text{vect}(\mathbf{F})$ is the vectorization of \mathbf{F} . From that, a naive QFE can be constructed as follows.

Encryption Scheme 1 (Naive QFE)

- **QFE.Enc**(\mathbf{x}) : Compute $c_{\mathbf{x}} \leftarrow \text{IPFE.Enc}(\mathbf{x} \otimes \mathbf{x})$.
- **QFE.KeyGen**(\mathbf{F}) : Compute $sk_{\mathbf{F}} \leftarrow \text{IPFE.KeyGen}(\text{vect}(\mathbf{F}))$.
- **QFE.Dec**($c_{\mathbf{x}}, sk_{\mathbf{F}}$) : Compute $(\mathbf{x} \otimes \mathbf{x})^{\top} \text{vect}(\mathbf{F}) \leftarrow \text{IPFE.Dec}(c_{\mathbf{x}}, sk_{\mathbf{F}})$.

But it is obvious that this leads to quadratic ciphertext sizes. Hence, constructions for single-input quadratic functional encryption are centered on achieving linear-size ciphertexts. However, such a naive approach does not work in the multi-input setting since we would need $\text{Enc}(\mathbf{x}_i \otimes \mathbf{x}_j)$ and $\mathbf{x}_i, \mathbf{x}_j$ to be encrypted independently.

There are only two proposals for multi-input quadratic functional encryption [21, 22]. In both, they adapt the single-input quadratic scheme from Lin [14] to the multi-input setting. The high level idea is to use function-hiding inner-product functional encryption. More specifically, during the encryption, every input is used as an input of both encryption and key generation of the function-hiding inner-product scheme. They achieve selective indistinguishability security for many challenge plaintexts, based on the security of the underlying scheme. In [21] they make use of two extra functionalities, namely predicated inner-product functional encryption and mixed group inner-product functional encryption, while in [22] the instantiation is simplified, while achieving a stronger sense of security allowing corruption of inputs, still in the indistinguishability based setting.

Another work by Gay [15] gives a transformation from “partially” function-hiding inner-product functional encryption to public-key single-input quadratic functional encryption scheme. This way, it achieves semi-adaptive simulation security for one ciphertext. Abdalla *et al.* in [20] gave a transformation from standard inner-product functional encryption to secret-key multi-input inner-product functional encryption scheme, achieving selective simulation security for one ciphertext and adaptive indistinguishability security for many ciphertexts. We will use elements of both these transformations to construct ours.

Efficiency-wise, let us consider an input plaintext of size $n\ell$, either by ℓ inputs of size n or a single input of size $n\ell$. The previous naive construction of a single-input quadratic functional encryption scheme achieves simulation-security with $O(n^2\ell^2)$ -bit size ciphertexts. On the other hand, Gay [15] proposes a simulation secure single-input quadratic functional encryption with ciphertext of size $O(n\ell)$. The ciphertext size of the multi-input inner product functional encryption from [20] is also $O(n\ell)$. In this work, we show that upgrading a single-input quadratic functional encryption to multi-input in simulation based security can be done at a cost linear in ℓ , leading to a ciphertext size of $O(n\ell^2)$. With indistinguishability based security, the scheme in [22] achieves a ciphertext of size $O(n\ell)$. For a summary, see Table 1.

Concerning simulation secure function-hiding inner-product FE, there only exists by Kim *et al.*’s protocol in [29], where they achieve adaptive simulation security against many challenge ciphertexts by constraining themselves to the

Table 1 Relevant functional encryption schemes, where ℓ refers to the number of inputs and n to the size of each input. For a fair comparison we consider the single-input schemes as one input of size $n\ell$. PFH stands for partially function-hiding.

Proposal	Starting building block	Functionality	SIM security	Ciphertext size
Naive (1)	IPFE	QFE	✓	$O(n^2\ell^2)$
[15]	PFH-IPFE	QFE	✓	$O(n\ell)$
[20]	IPFE	MIPFE	✓	$O(n\ell)$
[22]	FH-IPFE	MIQFE	✗	$O(n\ell)$
Our work	FH-IPFE	MIQFE	✓	$O(n\ell^2)$

generic group model (GGM). Since this is an oracle-based model, the impossibility results for adaptive simulation security [1, 2, 17] no longer hold. There is also [42] which claims a function-hiding inner-product functional encryption scheme simulation secure against many challenge ciphertexts in the standard model, which is known to be impossible even in the non function-hiding setting.

Finally, regarding the use of functional encryption to instantiate efficient schemes to provide differential privacy for computation over encrypted data, there are two recent works that give solutions to computing private inner-products. In [7], Zalonis *et al.* use function-hiding inner-product functional encryption to give their instantiation by hiding the differentially private noise in the functional key. They give an instantiation for noisy multi-input inner-product functional encryption secure for one ciphertext which they use to privately perform counting queries over private medical data. In [6], Alborch *et al.* use multi-input inner-product functional encryption to give their instantiation being able to avoid function-hiding and most importantly pairings. They give an instantiation of randomized inner-product functional encryption secure for one ciphertext which they use to privately perform linear queries over private data.

1.3 Technical Overview

Multi-input Quadratic Functional Encryption Scheme.

Our objective is to construct a simulation sound multi-input quadratic functional encryption scheme. One approach could have been to prove that the schemes [21, 22] are simulation sound. However, the way both these schemes are constructed from the single-input quadratic functional encryption scheme by Lin [14] makes it impossible. Indeed, during the encryption of input x_i they run both the encryption and key generation algorithms of a function-hiding inner-product. Then during decryption, they multiply all the results of the decryptions of all the cross terms i, j by the coefficients of the matrix $\mathbf{F}_{i,j}$ to obtain the desired result. Because there are several inputs to be encrypted independently, using both the encryption and key generation algorithms, the selective simulation soundness for one challenge ciphertext in the multi-input scheme would require adaptive simulation soundness for several challenge ciphertexts. But adaptive simulation soundness is hard to achieve

for non function-hiding functional encryption in the standard model [1], and even more in the case of function-hiding functional encryption.

Our idea is hence to start from the single-input quadratic functional encryption scheme in [15] based on partially function-hiding inner-product functional encryption. Function-hiding is said to be partial when decryption keys partially hide their underlying function (see [15] for details). Such a construction is sketched below, in the secret key setting to simplify the reading.

Encryption Scheme 2 (Simplified Figure 4, [15])

- **QFE.Setup**(1^κ) : Sample $\mathbf{a} \in \mathbb{Z}_p^2$, $\mathbf{B} \in \mathbb{Z}_p^{3 \times 2}$, $\mathbf{U} \in \mathbb{Z}_p^{n \times 2}$ and $\mathbf{V} \in \mathbb{Z}_p^{m \times 3}$. Define

$$\mathbf{M} := \left(\begin{array}{c|c} \mathbf{a} \otimes (\text{Id}_m | \mathbf{V}\mathbf{B}) & 0 \\ \hline 0 & \text{Id}_n \otimes \mathbf{B} \end{array} \right),$$

and run $(\text{IPFE.pk}, \text{IPFE.msk}) \leftarrow \text{IPFE.Setup}(1^\kappa, [\mathbf{M}]_1)$. Output $\text{QFE.msk} = (\mathbf{a}, \mathbf{B}, \mathbf{U}, \mathbf{V}, \text{IPFE.pk}, \text{IPFE.msk})$

- **QFE.Enc**(\mathbf{x}) : Sample $r \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^2$ and compute

$$\begin{aligned} [\mathbf{ct}_x^1]_1 &:= [\mathbf{x} + \mathbf{U}\mathbf{a}r]_1, \quad [\mathbf{ct}_x^2]_2 := [\mathbf{x} + \mathbf{V}\mathbf{B}\mathbf{s}]_2, \\ \text{IPFE.c} &\leftarrow \text{IPFE.Enc} \left(\text{IPFE.pk}, \left(r \otimes \begin{pmatrix} \mathbf{x} \\ \mathbf{s} \end{pmatrix} \right) \right). \end{aligned}$$

Output $c_x = ([\mathbf{ct}_x^1]_1, [\mathbf{ct}_x^2]_2, \text{IPFE.c})$.

- **QFE.KeyGen**(\mathbf{F}) : Compute

$$\text{IPFE.sk} \leftarrow \text{IPFE.KeyGen} \left(\text{IPFE.msk}, \begin{pmatrix} \text{vect}(\mathbf{U}^\top \mathbf{F}) \\ \text{vect}(\mathbf{F}\mathbf{V}) \end{pmatrix} \right).$$

Output $sk_{\mathbf{F}} = (\mathbf{F}, \text{IPFE.sk})$.

- **QFE.Dec**($c_x, sk_{\mathbf{F}}$) : Compute

$$[d]_T \leftarrow \text{IPFE.Dec}(\text{IPFE.c}, \text{IPFE.sk}), \quad [v]_T := e([\mathbf{ct}_x^1]_1, \mathbf{F}[\mathbf{ct}_x^2]_2) - [d]_T$$

Output $\log([v]_T)$ if $v \in [0, n^2 \cdot B^3]$ and \perp otherwise.

At a high level, there are two one-time pads $\mathbf{ct}_x^1, \mathbf{ct}_x^2$ which are combined as $\mathbf{ct}_x^1 \top \mathbf{F} \mathbf{ct}_x^2$ to give the desired value $\mathbf{x}^\top \mathbf{F} \mathbf{x}$ plus some extra terms. The inner-product functional encryption scheme is used to compute these extra terms so they can be subtracted in the end. Eventually, the partially function-hiding property is used to ensure that the functional key does not leak too much information about \mathbf{U} and \mathbf{V} (in concrete, $[\mathbf{ct}_x^i]_i$ remain indistinguishable from random).

The first idea that comes to mind to extend this scheme from single-input to multi-input would be to encrypt each input and substitute the (partially function-hiding) inner-product functional encryption scheme for a (partially function-hiding) multi-input one. However, the decryption phase necessitates to eliminate the extra terms coming from $e([\mathbf{ct}_{\mathbf{x}_i}^1]_1, \mathbf{F}_{i,j}[\mathbf{ct}_{\mathbf{x}_j}^2]_2)$:

$$\begin{aligned} e([\mathbf{ct}_{\mathbf{x}_i}^1]_1, \mathbf{F}_{i,j}[\mathbf{ct}_{\mathbf{x}_j}^2]_2) &= [\mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j]_T + [\mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{V} \mathbf{B} \mathbf{s}_i]_T \\ &\quad + [\mathbf{U} \mathbf{a} r_i^\top \mathbf{F}_{i,j} \mathbf{x}_j]_T + [\mathbf{U} \mathbf{a} r_i^\top \mathbf{F}_{i,j} \mathbf{V} \mathbf{B} \mathbf{s}_i]_T. \end{aligned}$$

This elimination could be done using the IPFE encryption. In this case, we would need the ciphertext and functional key to be

$$\begin{aligned} \text{IPFE.}\tilde{c}_{i,j} &\leftarrow \text{IPFE.Enc} \left(\text{IPFE.pk}, \begin{pmatrix} r_i \otimes \begin{pmatrix} \mathbf{x}_j \\ \mathbf{s}_j \end{pmatrix} \\ \mathbf{x}_i \otimes \mathbf{s}_j \end{pmatrix} \right), \\ \text{IPFE.}\tilde{sk}_{i,j} &\leftarrow \text{IPFE.KeyGen} \left(\text{IPFE.msk}, \begin{pmatrix} \text{vect}(\mathbf{U}^\top \mathbf{F}_{i,j}) \\ \text{vect}(\mathbf{F}_{i,j} \mathbf{V}) \end{pmatrix} \right). \end{aligned}$$

However, \mathbf{x}_i and \mathbf{x}_j should be encrypted in different independent instances of the encryption algorithm and therefore such ciphertext cannot be created. To circumvent this issue we use two main properties of the multi-input scheme. Firstly, we use the fact that during decryption, all partial decryptions (for $i, j \in [\ell]$) will be added altogether, and in particular $\text{IPFE.}c_{i,j}$ and $\text{IPFE.}c_{j,i}$. This allows us to “interweave” the ciphertexts, where in input i we compute half of what is needed for $e([\mathbf{ct}_{\mathbf{x}_i}^1]_1, \mathbf{F}_{i,j}[\mathbf{ct}_{\mathbf{x}_j}^2]_2)$ and half of what is needed for $e([\mathbf{ct}_{\mathbf{x}_j}^1]_1, \mathbf{F}_{j,i}[\mathbf{ct}_{\mathbf{x}_i}^2]_2)$, while the other two halves are computed on input j . Therefore, combining the decryptions of $\text{IPFE.}c_{i,j}$ and $\text{IPFE.}c_{j,i}$ we obtain all the extra terms resulting from $e([\mathbf{ct}_{\mathbf{x}_i}^1]_1, \mathbf{F}_{i,j}[\mathbf{ct}_{\mathbf{x}_j}^2]_2)$ and $e([\mathbf{ct}_{\mathbf{x}_j}^1]_1, \mathbf{F}_{j,i}[\mathbf{ct}_{\mathbf{x}_i}^2]_2)$. More specifically, by computing the following where the changes are squared

$$\begin{aligned} \text{IPFE.}c_{i,j} &\leftarrow \text{IPFE.Enc} \left(\text{IPFE.pk}, \begin{pmatrix} r_j \otimes \begin{pmatrix} \boxed{\mathbf{x}_i} \\ \mathbf{s}_i \end{pmatrix} \\ \mathbf{x}_i \otimes \mathbf{s}_j \end{pmatrix} \right), \\ \text{IPFE.}sk_{i,j} &\leftarrow \text{IPFE.KeyGen} \left(\text{IPFE.msk}, \begin{pmatrix} \text{vect}(\mathbf{U}^\top \boxed{\mathbf{F}_{j,i}}) \\ \text{vect}(\mathbf{F}_{i,j} \mathbf{V}) \end{pmatrix} \right), \end{aligned}$$

which given the linearity of inner product, during decryption we get for any $i, j \in [\ell]$,

$$\text{IPFE.Dec}(\text{IPFE.}c_{i,j}, \text{IPFE.}sk_{i,j}) + \text{IPFE.Dec}(\text{IPFE.}c_{j,i}, \text{IPFE.}sk_{j,i})$$

=

$$\text{IPFE.Dec}(\text{IPFE.}\tilde{c}_{i,j}, \text{IPFE.}\tilde{sk}_{i,j}) + \text{IPFE.Dec}(\text{IPFE.}\tilde{c}_{j,i}, \text{IPFE.}\tilde{sk}_{j,i})$$

which is what we need to eliminate the extra terms appearing in both crossed terms i, j and j, i .

Secondly, we use the fact that we are in the *secret-key* setting. As such we can presample the random values we will use in the encryption algorithm during the set up. From the encryption of a specific input, this gives us access to the values of all the rest of the inputs.

Furthermore, this use of secret-key cryptography allows us to directly base ourselves in function-hiding inner-product functional encryption, without the need to use its “partial” version (the scheme from [15] cannot do so since public-key function-hiding functional encryption is impossible). We also simplify the scheme to require less security assumptions. Eventually, the security of our scheme depends solely on the security of the underlying function-hiding inner-product functional encryption scheme.

The final step of our transformation is to use the argument in [20] to make sure that the correct result can only be obtained when all the inputs are taken into account. More precisely, they encrypt a one-time pad $\mathbf{w} + \mathbf{x}$ and then compute an extra $zk_{\mathbf{y}} = \mathbf{w}^\top \mathbf{y}$ during key generation, which one is finally subtracted during decryption to obtain the real value. This holds as long as there are less functional key queries than the size of the vector since \mathbf{w} will still have enough entropy to make $zk_{\mathbf{F}}$ indistinguishable from random.

Function-hiding Inner-product Functional Encryption Scheme.

Similarly to the construction of partially function-hiding functional encryption from [15, Section 4], we construct our function-hiding inner product functional encryption scheme by layering two instances of a non function-hiding scheme (one in and one out) as follows: $\text{Enc}^{\text{FH}}(\mathbf{x}) = \text{Enc}^{\text{out}}(\text{KeyGen}^{\text{in}}(\mathbf{x}))$ and $\text{KeyGen}^{\text{FH}}(\mathbf{y}) = \text{KeyGen}^{\text{out}}(\text{Enc}^{\text{in}}(\mathbf{y}))$. Then, decryption is done through the use of a bilinear pairing. This allows us to protect both the plaintext in the ciphertext and the function in the functional key.

The instantiation in [15, Section 4] is based on the non function-hiding scheme from [9, Section 3], since it intends to achieve simulation security in the public key setting and as such it needs an extra slot to handle this. In our case, given that we are in the secret key setting, basing ourselves in the scheme from [8, Section 3] is enough. As such we use a different approach than [15] to simulate the functional keys, where we use the Q -fold DDH assumption instead of the “1”-fold DDH one. This allows us to have slightly smaller ciphertexts and functional keys (one less slot) than the construction in [15].

This layering approach is very similar to the function-hiding inner product functional encryption scheme given in [14, Section 6.3]. Indeed, the crucial difference is the order in which the layers are set. In their case they have $\text{Enc}^{\text{FH}}(\mathbf{x}) = \text{KeyGen}^{\text{out}}(\text{Enc}^{\text{in}}(\mathbf{x}))$ and $\text{KeyGen}^{\text{FH}}(\mathbf{y}) = \text{Enc}^{\text{out}}(\text{KeyGen}^{\text{in}}(\mathbf{y}))$. However, this set up does not work for proving simulation security. As evidence, let us denote $\text{key}^{\text{out}}, \text{key}^{\text{in}}$ the keys for each layer of non function-hiding inner-product functional encryption. When simulating the function-hiding ciphertext by simulating the encryption in the inner layer, the simulated ciphertext will

still depend on (non-simulated) key^{out} . Then, when simulating the function-hiding functional key by simulating the encryption in the outer layer, the key we need to simulate is both in the ciphertext and functional key, which are in two different groups. As such, trying to use the DDH assumption (in which [8] is based) to simulate this functional key will not work since there will be one element in \mathbb{G}_1 and another in \mathbb{G}_2 : the bilinear pairing trivially breaks the scheme.

In our case, when simulating the function-hiding ciphertext by simulating the encryption in the *outer* layer, the ciphertext no longer depends on key^{in} and as such we are free to use the DDH assumption to simulate the function-hiding functional key, simulating the encryption in the inner layer.

Randomized Quadratic Functional Encryption Scheme.

Our objective is to build a simulation secure randomized quadratic functional encryption scheme, in other words, a scheme for which given an encryption of a plaintext \mathbf{x} and a functional key related to a quadratic function \mathbf{F} and some probability distribution D , the decryption algorithm output is distributed as $\mathbf{x}^\top \mathbf{F} \mathbf{x} + D$. One first approach to build a randomized quadratic functional encryption scheme would be to take a function-hiding functional encryption scheme for quadratic functionalities and hide the noise in the functional key, in a similar fashion as [7]. However, no such scheme is known. On top of that, such a scheme would need to “encrypt” the quadratic function which grows quadratically with respect to the size of the plaintext. This means that the functional keys would grow much faster than the plaintext.

Another approach would be to use a two-input quadratic functional encryption and use one input to encrypt the plaintext during the encryption of the randomized scheme and the other to encrypt the noise during the key generation of the randomized scheme, similarly to the concept of the solution of [6]. However, basing the security of the randomized scheme directly on the two-input scheme as a blackbox would require a two-input scheme which should be simulation secure against many challenge ciphertexts (one per randomized functional key). Furthermore, given that the noise is added linearly at the end (without squaring), encrypting it through a quadratic scheme does not seem optimal.

Therefore, we look more in details our specific instantiation of multi-input quadratic functional encryption. During the encryption, as said before, there are two separate parts, a one-time pad $\mathbf{ct}_x = \mathbf{x} + c \cdot \mathbf{u}$ and a function-hiding inner-product ciphertext IPFE. c . Then during encryption, the one-time pads are applied to the quadratic function to obtain $\mathbf{x}^\top \mathbf{F} \mathbf{x} + \gamma$, where γ are some extra terms we want to eliminate. We do so by using the function-hiding inner-product scheme to compute γ . The objective is to obtain $\mathbf{x}^\top \mathbf{F} \mathbf{x} + e$ for some $e \leftarrow D$. Then, note that changing the one-time pad to incorporate this noise in the first computation obtaining $\mathbf{x}^\top \mathbf{F} \mathbf{x} + e + \gamma$ and then eliminate γ gives

the same output as not modifying the one-time pad and having the function-hiding inner-product scheme output $\gamma - e$. As such, using a function-hiding randomized inner-product scheme should suffice.

This, however, leaves two unanswered questions. Firstly, the known randomized inner-product functional encryption schemes [6] do not consider function-hiding. Secondly, it is unclear if releasing e together with γ gives the adversary more information about e than only obtaining $\mathbf{x}^\top \mathbf{F}\mathbf{x} + e$ (thus compromising simulation security). By using the ideas from our instantiation of multi-input quadratic functional encryption we handle both issues, since the one-time pad \mathbf{w} in the function-hiding inner-product encryption together with a one-time pad for the noise allows us to prove simulation security and maintains the function-hiding properties.

2 Preliminaries

One-dimensional elements will be noted as lower-case letters (x, y, \dots) , while vectors will use bold lower-case letters $(\mathbf{x}, \mathbf{y}, \dots)$ and matrices will use bold upper-case letters $(\mathbf{M}, \mathbf{U}, \mathbf{V}, \dots)$. Let D be a probability distribution, $x \leftarrow D$ means the element x is sampled from the distribution D , while for any set \mathcal{Y} , $y \stackrel{\$}{\leftarrow} \mathcal{Y}$ means that y is sampled uniformly at random from \mathcal{Y} . Finally, a function f is said to be *negligible* over n ($f = \text{negl}(n)$) if for all $k \in \mathbb{N}_{>0}$, there exists $n_0 \in \mathbb{N}_{>0}$ such that for any $n > n_0$ then $|f(n)| < 1/n^k$.

2.1 Pairing Groups

This work makes use of asymmetric pairing groups, inherited from function-hiding functional encryption. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be three additive cyclic groups of order a prime p . Let P_1, P_2 be generators of \mathbb{G}_1 and \mathbb{G}_2 respectively and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be an efficiently computable (non-degenerate) bilinear map. This means that $e(\alpha P_1, \beta P_2) = \alpha \cdot \beta P_T$ for any $\alpha, \beta \in \mathbb{Z}_p$ where we define $P_T := e(P_1, P_2)$.

For $s \in \{1, 2, T\}$ and a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ for any $n, m \geq 1$ we define $[\mathbf{A}]_s$ as the representation of \mathbf{A} in the group \mathbb{G}_s . In other words, $[\mathbf{A}]_s = (a_{ij} P_s) \in \mathbb{G}_s^{n \times m}$. For any two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_p^{n \times m}$, $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T$ and for $s \in \{1, 2, T\}$ we have $[\mathbf{A}]_s + [\mathbf{B}]_s := [\mathbf{A} + \mathbf{B}]_s$.

Finally, we define the PPT algorithm PGGGen that, on input a security parameter κ , outputs a set $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e)$ where p is a 2κ -bit prime.

2.2 The DDH Assumption

Let p be a prime number, we define the following distribution, using the framework from [43]. The DDH distribution over \mathbb{Z}_p^2 samples $t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and outputs $\mathbf{t} := (1, t)^\top$.

Definition 1 Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and $s \in \{1, 2, T\}$. For any PPT adversary \mathcal{A} we define the following advantage

$$\text{Adv}_{\mathbb{G}_s}^{\text{DDH}}(\mathcal{A}) := |\Pr[1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{P}\mathcal{G}, [t]_s, [tr]_s)] - \Pr[1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{P}\mathcal{G}, [t]_s, [w]_s)]|,$$

where the probability is taken over $\mathcal{P}\mathcal{G} \leftarrow \text{PGGen}(1^\kappa)$, $t \leftarrow \text{DDH}$, $r \xleftarrow{\$} \mathbb{Z}_p$ and $w \xleftarrow{\$} \mathbb{Z}_p^2$. We say that the *Decisional Diffie-Hellman (DDH) assumption* holds if for all PPT adversaries \mathcal{A} $\text{Adv}_{\mathbb{G}_s}^{\text{DDH}}(\mathcal{A}) \leq \text{neg}(\kappa)$.

We are also interested in the case where Q independent queries are asked, with same t but different r_i , and its relationship to the base DDH.

Definition 2 Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and $s \in \{1, 2, T\}$. For any PPT adversary \mathcal{A} we define the following advantage

$$\text{Adv}_{\mathbb{G}_s}^{Q\text{-DDH}}(\mathcal{A}) := |\Pr[1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{P}\mathcal{G}, [t]_s, [tr^\top]_s)] - \Pr[1 \leftarrow \mathcal{A}(1^\kappa, \mathcal{P}\mathcal{G}, [t]_s, [\mathbf{W}]_s)]|,$$

where the probability is taken over $\mathcal{P}\mathcal{G} \leftarrow \text{PGGen}(1^\kappa)$, $t \leftarrow \text{DDH}$, $r \xleftarrow{\$} \mathbb{Z}_p^Q$ and $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{2 \times Q}$. We say that the *Q -fold Decisional Diffie-Hellman (Q -DDH) assumption* holds if for all PPT adversaries \mathcal{A} $\text{Adv}_{\mathbb{G}_s}^{Q\text{-DDH}}(\mathcal{A}) \leq \text{neg}(\kappa)$.

More concretely we will use the random self-reducibility of the Q -fold DDH assumption.

Lemma 1 (Random Self-reducibility, [43]) *Let $Q > 1$, and $s \in \{1, 2, T\}$. Then, for any PPT adversary \mathcal{A} there exists a PPT adversary \mathcal{B} such that*

$$\text{Adv}_{\mathbb{G}_s}^{Q\text{-DDH}}(\mathcal{A}) \leq \text{Adv}_{\mathbb{G}_s}^{\text{DDH}}(\mathcal{B}) + \frac{1}{p-1},$$

where the probability is taken over $\mathcal{P}\mathcal{G} \leftarrow \text{PGGen}(1^\kappa)$, $t \leftarrow \text{DDH}$, $r \xleftarrow{\$} \mathbb{Z}_p^Q$ and $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{2 \times Q}$.

2.3 Functional Encryption

Functional encryption is a generalization of encryption first formalized by Boneh *et al.* [1] and O’Neill [2], in which the decryption algorithm no longer outputs necessarily the plaintext, but a function applied to this plaintext. This is achieved through the generation of functional keys related to the specific function wanted to be applied. Such schemes are defined as follows in the secret-key setting.

Definition 3 (Functional Encryption Scheme) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and \mathcal{F} be a family of functions. A function $f \in \mathcal{F}$ is defined as $f : \mathcal{X} \rightarrow \mathcal{S}$. We define a *secret-key functional encryption scheme* the following tuple of PPT algorithms:

- $\text{FE.Setup}(1^\kappa, \mathcal{F})$: given the security parameter κ and the family of functions \mathcal{F} as input, it outputs some public parameters FE.param and a master secret key FE.msk . We will assume the public parameters as inputs in all other algorithms.
- $\text{FE.Enc}(\text{FE.msk}, x)$: given the master secret key FE.msk and a plaintext $x \in \mathcal{X}$ as inputs, it outputs a ciphertext c_x .
- $\text{FE.KeyGen}(\text{FE.msk}, f)$: given the master secret key FE.msk and a function $f \in \mathcal{F}$ as inputs, it outputs a functional key sk_f .
- $\text{FE.Dec}(c_x, sk_f)$: given a ciphertext c_x and a functional key sk_f as inputs, it outputs a value in \mathcal{S} or \perp if it fails.

The correctness notion for such schemes is as follows.

Definition 4 (Correctness of Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and $\text{FE} = (\text{FE.Setup}, \text{FE.Enc}, \text{FE.KeyGen}, \text{FE.Dec})$ be a secret-key functional encryption scheme. We say it is *correct* if for any $x \in \mathcal{X}$ and $f \in \mathcal{F}$ we have

$$\Pr [\text{FE.Dec}(c_x, sk_f) \neq f(x)] = \text{negl}(\kappa)$$

where the distribution is taken over $\text{FE.msk} \leftarrow \text{FE.Setup}(1^\kappa, \mathcal{F})$, $c_x \leftarrow \text{FE.Enc}(\text{FE.msk}, x)$ and $sk_f \leftarrow \text{FE.KeyGen}(\text{FE.msk}, f)$.

2.3.1 Multi-input Functional Encryption

As mentioned in Section 1, multi-input functional encryption is a generalisation of functional encryption which divides the plaintext into ℓ inputs to be encrypted independently. For this case, the standard definitions are as follows.

Definition 5 (Multi-input Functional Encryption Scheme) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter, $\ell \in \mathbb{N}_{>0}$ be the number of inputs and \mathcal{F} be a family of ℓ -ary functions. A function $f \in \mathcal{F}$ is defined as $f : \mathcal{X}_1 \times \dots \times \mathcal{X}_\ell \rightarrow \mathcal{S}$. We define a *secret-key multi-input quadratic functional encryption scheme* as the following tuple of PPT algorithms:

- $\text{MIFE.Setup}(1^\kappa, \mathcal{F})$: given the security parameter 1^κ and a family of ℓ -ary functions \mathcal{F} , it outputs some public parameters MIFE.param a master secret key MIFE.msk . We will assume the public parameters as inputs in all other algorithms.
- $\text{MIFE.Enc}(\text{MIFE.msk}, i, x_i)$: given the master secret key MIFE.msk , an input number $i \in [\ell]$ and $x_i \in \mathcal{X}_i$, it outputs a ciphertext c_{x_i} .
- $\text{MIFE.KeyGen}(\text{MIFE.msk}, f)$: given the master secret key MIFE.msk and a function $f \in \mathcal{F}$ as inputs, it outputs a functional decryption key sk_f .
- $\text{MIFE.Dec}(c_{x_1}, \dots, c_{x_\ell}, sk_f)$: a deterministic algorithm that given ciphertexts $c_{x_1}, \dots, c_{x_\ell}$ and a functional key sk_f as inputs, it outputs a value in \mathcal{S} , or \perp if it fails.

The correctness notion for these schemes goes as follows.

Definition 6 (Correctness of Multi-input Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter, $\ell \in \mathbb{N}_{>0}$ be the number of inputs and $\text{MIFE} = (\text{MIFE.SetUp}, \text{MIFE.Enc}, \text{MIFE.KeyGen}, \text{MIFE.Dec})$ be a secret-key multi-input functional encryption scheme. We say it is *correct* if for any $x_1 \in \mathcal{X}_1, \dots, x_\ell \in \mathcal{X}_\ell$ and $f \in \mathcal{F}$ we have

$$\Pr [\text{MIFE.Dec}(c_{x_1}, \dots, c_{x_\ell}, sk_f) \neq f(x_1, \dots, x_\ell)] = \text{negl}(\kappa)$$

where the distribution is taken over $\text{MIFE.msk} \leftarrow \text{MIFE.SetUp}(1^\kappa, \mathcal{F})$, $c_{x_i} \leftarrow \text{MIFE.Enc}(\text{MIFE.msk}, i, x_i)$ for all $i \in [\ell]$ and $sk_f \leftarrow \text{MIFE.KeyGen}(\text{MIFE.msk}, f)$.

As explained in Section 1, there are two main ways to classify security definitions for functional encryption: indistinguishability based or simulation based and selective or adaptive. In this work we are interested in selective simulation security for one challenge ciphertext, for which we give the definition below.

Definition 7 (Simulation Security for Multi-input Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter, $\ell \in \mathbb{N}_{>0}$ be the number of inputs and $\text{MIFE} = (\text{MIFE.SetUp}, \text{MIFE.Enc}, \text{MIFE.KeyGen}, \text{MIFE.Dec})$ be a secret-key multi-input functional encryption scheme. For any PPT simulator $\mathcal{S} := (\text{MIFE.SetUpSim}, \text{MIFE.EncSim}, \text{MIFE.KeyGenSim})$ and any PPT adversary \mathcal{A} we define the experiments in Table 2 where the oracles are described as follows.

1. **Real Experiment:** $\mathcal{O}_{\text{MIFE.KeyGen}}(\text{MIFE.msk}, \cdot)$ takes as input a function $f \in \mathcal{F}$ and outputs $sk_f \leftarrow \text{MIFE.KeyGen}(\text{MIFE.msk}, f)$.
2. **Ideal Experiment:** $\widetilde{\mathcal{O}}_{\text{MIFE.KeyGen}}(\text{MIFE.msk}, x_1, \dots, x_\ell, \cdot)$ takes as input a function $f \in \mathcal{F}$, computes $v = f(x_1, \dots, x_\ell)$ and outputs $\widetilde{sk}_f \leftarrow \text{MIFE.KeyGen Sim}(\text{MIFE.msk}, v, f)$.

We say MIFE is *one selective multi-input simulation secure* if there exists a PPT simulator $\mathcal{S} := (\text{MIFE.SetUpSim}, \text{MIFE.EncSim}, \text{MIFE.KeyGenSim})$ such that for all PPT adversary \mathcal{A} the following inequality holds.

$$\text{Adv}_{\text{MIFE}}^{\text{MI-SIM}}(\mathcal{A}) = |\Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa)] - \Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{\text{ideal}}(1^\kappa)]| \leq \text{negl}(\kappa)$$

2.3.2 Function-hiding Functional Encryption

As mentioned in Section 1, function-hiding functional encryption is a restriction of functional encryption which guarantees privacy for the function from the functional key, as well as the standard privacy for the message from the ciphertext. The security definition in the simulation security setting is then as follows.

Definition 8 (Simulation Security of Function-hiding Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter, $n \in \mathbb{N}_{>0}$ be the dimension, and $\text{FE} = (\text{FE.SetUp}, \text{FE.Enc}, \text{FE.KeyGen}, \text{FE.Dec})$ be a secret-key functional encryption scheme. For any PPT simulator $\mathcal{S} := (\text{FE.SetUpSim}, \text{FE.EncSim}, \text{FE.KeyGen Sim})$ and any PPT

Table 2 Real and ideal experiments in SEL-SIM security for MIQFE.

- $\text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa)$:
 1: $\text{MIFE.msk} \leftarrow \text{MIQFE.Setup}(1^\kappa, \mathcal{F})$
 2: $(\{x_i\}_{i \in [\ell]}, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x_i \in \mathcal{X}_i$
 3: For all $i \in [\ell]$, $c_{x_i} \leftarrow \text{MIFE.Enc}(\text{MIFE.msk}, i, x_i)$
 4: $\gamma \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{MIFE.KeyGen}}(\text{msk}, \cdot)}(\{c_{x_i}\}_{i \in [\ell]}, \text{st}_1)$
- $\text{Exp}_{\mathcal{A}, \mathcal{S}}^{\text{ideal}}(1^\kappa)$:
 1: $\text{MIFE.msk} \leftarrow \text{MIFE.SetupSim}(1^\kappa, \mathcal{F})$
 2: $(\{x_i\}_{i \in [\ell]}, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x_i \in \mathcal{X}_i$
 3: For all $i \in [\ell]$, $\tilde{c}_{x_i} \leftarrow \text{MIFE.EncSim}(\text{MIFE.msk}, i)$
 4: $\gamma \leftarrow \mathcal{A}_2^{\tilde{\mathcal{O}}_{\text{MIFE.KeyGen}}(\text{MIFE.msk}, \{x_i\}_{i \in [\ell]}, \cdot)}(\{\tilde{c}_{x_i}\}_{i \in [\ell]}, \text{st}_1)$

Table 3 Real and ideal experiments in function-hiding SEL-SIM security for FE.

- | | |
|--|--|
| $\text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa)$:
1: $\text{FE.msk} \leftarrow \text{FE.Setup}(1^\kappa, \mathcal{F})$
2: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$
3: $c_x \leftarrow \text{FE.Enc}(\text{FE.msk}, x)$
4: $\gamma \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{FE.KeyGen}}(\text{FE.msk}, \cdot)}(c_x, \text{st}_1)$ | $\text{Exp}_{\mathcal{A}, \mathcal{S}}^{\text{ideal}}(1^\kappa)$:
1: $\text{FE.msk} \leftarrow \text{FE.SetupSim}(1^\kappa, \mathcal{F})$
2: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$
3: $\tilde{c}_x \leftarrow \text{FE.EncSim}(\text{FE.msk})$
4: $\gamma \leftarrow \mathcal{A}_2^{\tilde{\mathcal{O}}_{\text{FE.KeyGen}}(\text{FE.msk}, x, \cdot)}(\tilde{c}_x, \text{st}_1)$ |
|--|--|

adversary \mathcal{A} we define the experiments in Table 3 where the oracles are described as follows.

1. **Real Experiment:** $\mathcal{O}_{\text{FE.KeyGen}}(\text{FE.msk}, \cdot)$ takes as input a function $f \in \mathcal{F}$ and outputs $sk_f \leftarrow \text{FE.KeyGen}(\text{FE.msk}, f)$.
2. **Ideal Experiment:** $\tilde{\mathcal{O}}_{\text{FE.KeyGen}}(\text{FE.msk}, x, \cdot)$ takes as input a function $f \in \mathcal{F}$, computes $v = f(x)$ and outputs $\tilde{sk}_f \leftarrow \text{FE.KeyGenSim}(\text{FE.msk}, v)$.

We say FE is *one selective function-hiding simulation secure* if there exists a PPT simulator $\mathcal{S} := (\text{FE.SetupSim}, \text{FE.EncSim}, \text{FE.KeyGenSim})$ such that for all PPT adversary \mathcal{A} the following inequality holds.

$$\text{Adv}_{\text{FE}}^{\text{FH-SIM}}(\mathcal{A}) = |\Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa)] - \Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{\text{ideal}}(1^\kappa)]| \leq \text{negl}(\kappa)$$

The reason why this security definition captures function-hiding is the fact that FE.KeyGenSim takes as inputs *only* the simulated keys and the output of the function applied to the challenge. If this is satisfied, then the only information leaked from the ciphertext and functional key is the output of the function, since they both can be simulated only knowing this information.

2.3.3 Randomized Functional Encryption

Randomized functional encryption is a generalization of functional encryption first formalized by Goyal *et al.* [5], in which the decryption algorithm no longer

outputs a deterministic function applied to the plaintext, but a randomized function, in such a way that the distributions are independent for different functional keys. We define a randomized function as $\hat{f} : \mathcal{X} \times \mathcal{R} \rightarrow \mathcal{S}$, where $r \in \mathcal{R}$ is understood as the seed for the probabilistic sampling of the randomized function \hat{f} and as such, as true randomness completely unknown to the adversary. Such schemes are defined as follows in the secret-key setting.

Definition 9 (Randomized Functional Encryption Scheme) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and $\hat{\mathcal{F}}$ be a family of randomized functions. We define a *secret-key functional encryption scheme* the following tuple of PPT algorithms:

- $\text{RFE.Setup}(1^\kappa, \mathcal{F})$: given the security parameter κ and the family of randomized functions $\hat{\mathcal{F}}$ as input, it outputs some public parameters RFE.param and a master secret key RFE.msk . We will assume the public parameters as inputs in all other algorithms.
- $\text{RFE.Enc}(\text{RFE.msk}, x)$: given the master secret key RFE.msk and a plaintext $x \in \mathcal{X}$ as inputs, it outputs a ciphertext c_x .
- $\text{RFE.KeyGen}(\text{RFE.msk}, f)$: given the master secret key RFE.msk and a description of the randomized function $\hat{f} \in \hat{\mathcal{F}}$ as inputs, it outputs a functional key $sk_{\hat{f}}$.
- $\text{RFE.Dec}(c_x, sk_{\hat{f}})$: given a ciphertext c_x and a functional key $sk_{\hat{f}}$ as inputs, it outputs a value in \mathcal{S} or \perp if it fails.

The correctness notion for such schemes is differs from Definition 10 due to the probabilistic nature of the output. The definition is as follows, for a one ciphertext scheme.

Definition 10 (Correctness of Randomized Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter and $\text{RFE} = (\text{RFE.Setup}, \text{RFE.Enc}, \text{RFE.KeyGen}, \text{RFE.Dec})$ be a secret-key randomized functional encryption scheme supporting the family of randomized functions $\hat{\mathcal{F}}$. We say it is *correct* if for any plaintext x and any set of randomized functions $\hat{f}^1, \dots, \hat{f}^Q \in \hat{\mathcal{F}}$ the following distributions are computationally indistinguishable:

- $\text{Real}(1^\kappa, \hat{\mathcal{F}}) := \{v^i \leftarrow \text{Dec}(c_x, sk_{\hat{f}^i})\}_{i \in [Q]}$, where
 $(\text{RFE.param}, \text{RFE.msk}) \leftarrow \text{RFE.Setup}(1^\kappa)$
 $c_x \leftarrow \text{RFE.Enc}(\text{RFE.msk}, x)$
 $sk_{\hat{f}^i} \leftarrow \text{RFE.KeyGen}(\text{RFE.msk}, \hat{f}^i)$ for all $i \in [Q]$.
- $\text{Ideal}(1^\kappa, \hat{\mathcal{F}}) := \{\hat{f}^i(x; r^i)\}_{i \in [Q]}$ where $r^i \leftarrow \mathcal{R}$.

Concerning the security definition, we take the slightly stronger one given in [6] instead of the one given in [5], since it is the security definition that reduces to computational differential privacy (as shown in [6]). We change the notation to be more coherent with the rest of this work. Where they define the oracle $\hat{\mathcal{O}}_{\text{FE.KeyGen}}$ with access to another oracle KeyIdeal it can query to receive

Table 4 Real and ideal experiments in function-hiding SEL-SIM security for FE.

$\text{Exp}_{\mathcal{A}}^{real}(1^\kappa):$ 1: $\text{RFE.msk} \leftarrow \text{RFE.Setup}(1^\kappa, \hat{\mathcal{F}})$ 2: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$ 3: $c_x \leftarrow \text{RFE.Enc}(\text{RFE.msk}, x)$ 4: $\gamma \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{RFE.KeyGen}}(\text{RFE.msk}, \cdot)}(c_x, \text{st}_1)$	$\text{Exp}_{\mathcal{A}, \mathcal{S}}^{ideal}(1^\kappa):$ 1: $\text{RFE.msk} \leftarrow \text{RFE.SetupSim}(1^\kappa, \mathcal{F})$ 2: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$ 3: $\tilde{c}_x \leftarrow \text{RFE.EncSim}(\text{RFE.msk})$ 4: $\gamma \leftarrow \mathcal{A}_2^{\tilde{\mathcal{O}}_{\text{RFE.KeyGen}}(\text{RFE.msk}, x, \cdot)}(\tilde{c}_x, \text{st}_1)$
--	--

for each randomized function \hat{f} the value $v = \hat{f}(x; r)$ with some $r \leftarrow \mathcal{R}$, we assume v as an input to the simulated oracle.

Definition 11 (Simulation Security of Randomized Functional Encryption) Let $\kappa \in \mathbb{N}_{>0}$ be a security parameter, $n \in \mathbb{N}_{>0}$ be the dimension, and $\text{RFE} = (\text{RFE.Setup}, \text{RFE.Enc}, \text{RFE.KeyGen}, \text{RFE.Dec})$ be a secret-key randomized functional encryption scheme. For any PPT simulator $\mathcal{S} := (\text{RFE.SetupSim}, \text{RFE.EncSim}, \text{RFE.KeyGenSim})$ and any PPT adversary \mathcal{A} we define the experiments in Table 4 where the oracles are described as follows.

1. **Real Experiment:** $\mathcal{O}_{\text{RFE.KeyGen}}(\text{RFE.msk}, \cdot)$ takes as input a randomized function $\hat{f} \in \hat{\mathcal{F}}$ and outputs $sk_{\hat{f}} \leftarrow \text{RFE.KeyGen}(\text{RFE.msk}, \hat{f})$.
2. **Ideal Experiment:** $\tilde{\mathcal{O}}_{\text{RFE.KeyGen}}(\text{RFE.msk}, x, \cdot)$ takes as input a randomized function $v = \hat{f} \in \hat{\mathcal{F}}$, computes $\hat{f}(x; r)$ for some $r \leftarrow \mathcal{R}$ and outputs $\tilde{sk}_{\hat{f}} \leftarrow \text{RFE.KeyGenSim}(\text{RFE.msk}, v)$.

We say RFE is *one selective simulation secure* if there exists a PPT simulator $\mathcal{S} := (\text{RFE.SetupSim}, \text{RFE.EncSim}, \text{RFE.KeyGenSim})$ such that for all PPT adversary \mathcal{A} the following inequality holds.

$$\text{Adv}_{\text{RFE}}^{\text{SIM}}(\mathcal{A}) = |\Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{real}(1^\kappa)] - \Pr[1 \leftarrow \text{Exp}_{\mathcal{A}}^{ideal}(1^\kappa)]| \leq \text{negl}(\kappa)$$

2.4 Differential Privacy

Differential privacy is a private data mechanism property first proposed by Dwork *et al.* [32], which allows for release of data analytics over private data in such a way that the privacy loss is easily analyzable. This property concerns the so called privacy mechanisms, randomized functions which on input a database and a query output a noisy response to the query applied to the database. In other words, we define a privacy mechanism over a family of queries \mathcal{F} as $\mathcal{M} : \mathcal{X} \times \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{S}$, where \mathcal{S} is contained in the multidimensional real numbers. This property only compares outputs of the mechanism applied to *neighbouring* databases, which are databases differing in only one individual, since it wants to verify that the privacy of the individual (opposing to groups) is not compromised. In this work we formally define neighbouring databases as follows.

Definition 12 (Neighbouring Databases) Two databases x, x' are said to be neighbouring

$$\|x - x'\|_1 \leq 1.$$

This means that for two databases to be neighbouring they must differ at most in one entry and at most by a difference of one. Having defined neighbouring databases, the differential privacy property goes as follows.

Definition 13 (Differential Privacy) Let ϵ, δ be two real numbers and \mathcal{F} a family of queries. A privacy mechanism for \mathcal{F} , $\mathcal{M} : \mathcal{X} \times \mathcal{F} \times \mathcal{R} \rightarrow \mathcal{S}$ is said to be (ϵ, δ) -differential private $((\epsilon, \delta)$ -DP) if for all $S \subseteq \mathcal{S}$, any query $f \in \mathcal{F}$, every pair of neighbouring databases $x, x' \in \mathcal{X}$ and $r, r' \leftarrow \mathcal{R}$

$$\Pr[\mathcal{M}(x, f; r) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(x', f; r') \in S] + \delta.$$

Note that this definition only considers mechanisms where only one single query is asked. If we want to handle multiple queries selectively, the same definition works, one can just consider the set of queries f_1, \dots, f_Q as a new bigger query F to analyze with the output being contained in \mathcal{S}^Q . However if the queries are adaptive we can use the following property of differential privacy.

Proposition 2 (Sequential Composition, [44]) *Let \mathcal{M} be a mechanism allowing Q adaptive queries to a mechanism $\mathcal{M}' : \mathcal{X} \times \mathcal{F} \times \mathcal{R} \rightarrow \mathcal{S}$ satisfying (ϵ, δ) -DP. Then \mathcal{M} satisfies $(Q \cdot \epsilon, Q \cdot \delta)$ -DP.*

Finally, two additional concepts that are useful to calibrate the added noise and then to evaluate how good the mechanism is, are as follows.

Definition 14 (Sensitivity) Let $x, x' \in \mathcal{X}$ be two neighbouring databases. The ℓ_1 -sensitivity of a function f is

$$\Delta_f := \max_{x, x' \text{ neighbouring}} \|f(x) - f(x')\|_1.$$

This can be naturally extended to the ℓ_1 -sensitivity of a family of queries \mathcal{F} by taking the maximum over the family, which we will denote as $\Delta_{\mathcal{F}}$.

Definition 15 (Utility) Let $\mathcal{M} : \mathcal{X} \times \mathcal{F} \times \mathcal{R} \rightarrow \mathcal{S}$, $\mathcal{M}(x, f; r) = f(x) + e(r)$ be a differentially private mechanism. We say \mathcal{M} is (α, β) -useful if

$$\Pr[|\mathcal{M}(x, f; r) - f(x)| \leq \alpha] \geq 1 - \beta$$

for any $x \in \mathcal{X}$, $f, \in \mathcal{F}$ and $r \leftarrow \mathcal{R}$.

For our instantiation we will use the well-known geometric mechanism as proposed by [45], where they show that it is an optimal mechanism. The exact parameters and utility of the geometric mechanism is as follows.

Lemma 3 (Privacy of Geometric Mechanism) *Let \mathcal{X} be a database space, $\mathcal{S} = \mathbb{Z}^Q$, \mathcal{F} be a family of queries, let D be a random variable, $D_\epsilon \sim \text{Geo}(\exp(-\epsilon/\Delta_{\mathcal{F}}))$, where $\Delta_{\mathcal{F}}$ is as in Definition 14. Then the geometric mechanism defined as*

$$\mathcal{M}(x, f; r) := f(x) + e(r_f)$$

where $x \in \mathcal{X}$ and $e(r_f) \leftarrow D_\epsilon$ is $(\epsilon, 0)$ -DP.

Lemma 4 (Utility of Geometric Mechanism) *The Geometric mechanism as described in Lemma 3 is $\left(O\left(\frac{1}{\epsilon}\right) \cdot \Delta_{\mathcal{F}} \cdot \log\left(\frac{\beta}{2}\right), \beta\right)$ -useful.*

3 Multi-input Quadratic Functional Encryption Scheme

In this section we describe our multi-input quadratic functional encryption scheme for bounded-norm quadratic functionalities. We first describe the family of functions we want to cover. Let $\mathcal{F}_{Q,B}^{n,\ell} : ([0, B]^n)^\ell \rightarrow [0, (n\ell)^2 \cdot B^3]$ be the family of ℓ -ary functions such that a function $\mathbf{F} \in \mathcal{F}_{Q,B}^{n,\ell}$ is defined by a matrix in $[0, B]^{n\ell \times n\ell}$ which we note as

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{1,1} & \cdots & \mathbf{F}_{1,\ell} \\ \vdots & \ddots & \vdots \\ \mathbf{F}_{\ell,1} & \cdots & \mathbf{F}_{\ell,\ell} \end{pmatrix}$$

with $\mathbf{F}_{i,j} \in [0, B]^{n \times n}$, and applied to $(\mathbf{x}_1, \dots, \mathbf{x}_\ell) \in ([0, B]^n)^\ell$ gives $\mathbf{F}(\mathbf{x}_1, \dots, \mathbf{x}_\ell) := \sum \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j$ for $i, j \in [\ell]$.

Our MIQFE scheme is based on a function-hiding inner-product functional encryption, whose family of functions is $\tilde{\mathcal{F}}_{\text{IP}}^{2n} : \mathbb{Z}_p^{2n} \rightarrow \mathbb{G}_T$ (for some $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \text{PGGen}(1^\kappa)$). A function $\mathbf{y} \in \tilde{\mathcal{F}}_{\text{IP}}^{2n}$ is defined by a vector in \mathbb{Z}_p^{2n} and applied to \mathbf{z} gives $\mathbf{y}(\mathbf{z}) := [\mathbf{z}^\top \mathbf{y}]_T$.

3.1 Description of the Scheme

Let $\text{IPFE} = (\text{IPFE.Setup}, \text{IPFE.Enc}, \text{IPFE.KeyGen}, \text{IPFE.Dec})$ be a function-hiding inner-product functional encryption scheme for the family of functions $\tilde{\mathcal{F}}_{\text{IP}}^{2n}$. Below is a description of our MIQFE scheme for the family of functions $\mathcal{F}_{Q,B}^{n,\ell}$.

Encryption Scheme 3 (MIQFE Scheme)

- **MIQFE.Setup** $(1^\kappa, \mathcal{F}_{Q,B}^{n,\ell})$: Sample $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \text{PGGen}(1^\kappa)$, $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^n$ and $c_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [\ell]$ and sample $\mathbf{w}_{i,j} \xleftarrow{\$} \mathbb{Z}_p^{2n}$ for $i, j \in [\ell]$. Run $\text{IPFE.msk}_{i,j} \leftarrow \text{IPFE.Setup}(1^\kappa, \tilde{\mathcal{F}}_{\text{IP}}^{2n}, \mathcal{PG})$ for $i, j \in [\ell]$. Output

MIQFE.param = \mathcal{PG} and

$$\text{MIQFE.msk} = (\{\mathbf{u}_i, c_i\}_{i \in [\ell]}, \{\mathbf{w}_{i,j}, \text{IPFE.msk}_{i,j}\}_{i,j \in [\ell]}).$$

- **MIQFE.Enc**(MIQFE.msk, i, \mathbf{x}_i) : Compute

$$\mathbf{ct}_{\mathbf{x}_i} := \mathbf{x}_i + c_i \mathbf{u}_i \in \mathbb{Z}_p^n,$$

$$\text{IPFE.c}_{i,j} \leftarrow \text{IPFE.Enc} \left(\text{IPFE.msk}_{i,j}, \mathbf{w}_{i,j} + c_j \begin{pmatrix} \mathbf{ct}_{\mathbf{x}_i} \\ \mathbf{x}_i \end{pmatrix} \right) \text{ for } j \in [\ell].$$

Output $\text{MIQFE.c}_i = (\mathbf{ct}_{\mathbf{x}_i}, \{\text{IPFE.c}_{i,j}\}_{j \in [\ell]}).$

- **MIQFE.KeyGen**(MIQFE.msk, \mathbf{F}) : Compute

$$\text{IPFE.sk}_{i,j} \leftarrow \text{IPFE.KeyGen} \left(\text{IPFE.msk}_{i,j}, \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix} \right) \text{ for } i, j \in [\ell],$$

$$zk_{\mathbf{F}} \leftarrow \sum_{i,j \in [\ell]} \mathbf{w}_{i,j}^\top \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix}.$$

Output $\text{MIQFE.sk}_{\mathbf{F}} = (\mathbf{F}, \{\text{IPFE.sk}_{i,j}\}_{i,j \in [\ell]}, zk_{\mathbf{F}}).$

- **MIQFE.Dec**(MIQFE.c₁, ..., MIQFE.c_ℓ, MIQFE.sk_ℱ) : Compute

$$[d_{i,j}]_T \leftarrow \text{IPFE.Dec}(\text{IPFE.c}_{i,j}, \text{IPFE.sk}_{i,j})$$

$$[v]_T := \left(\sum_{i,j \in [\ell]} [\mathbf{ct}_{\mathbf{x}_i}^\top \mathbf{F}_{i,j} \mathbf{ct}_{\mathbf{x}_j}]_T - [d_{i,j}]_T \right) + [zk_{\mathbf{F}}]_T$$

Output $\log([v]_T)$ if $v \in [0, (n\ell)^2 \cdot B^3]$ and \perp otherwise.

Remark 1 We define the output of the functions \mathbf{y} to be in \mathbb{G}_T to be compatible with existing function-hiding IPFE and only for that. Designing such a scheme without pairings is a hard open problem. This means that all schemes require bounded inputs to have a bounded output from which the discrete logarithm can be computed. Therefore, this requires us to perform the operations for the decryption algorithm in the exponent, since the input $\mathbf{w}_{i,j} + c_j \begin{pmatrix} \mathbf{ct}_{\mathbf{x}_i} \\ \mathbf{x}_i \end{pmatrix}$ is not bounded by definition thus making it unfeasible to compute $d_{i,j}$ in plain.

This means then that we add no extra pairing operations on top of those needed for the function-hiding scheme. As such, were there to be a scheme without pairings or allowing non-bounded inputs this property would immediately translate to our construction.

3.2 Correctness and Security

Proposition 5 *The MIQFE scheme defined in Section 3.1 is a correct multi-input functional encryption scheme for $\mathcal{F}_{Q,B}^{n,\ell}$ as long as IPFE is a correct scheme for $\tilde{\mathcal{F}}_{\text{IP}}^{2n}$.*

Proof For ease of notation and reading we will leave out the $[\cdot]_T$. First we have

$$\mathbf{ct}_{\mathbf{x}_i}^\top \mathbf{F}_{i,j} \mathbf{ct}_{\mathbf{x}_j} = \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j + c_i \mathbf{u}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j + \mathbf{x}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j + c_i \mathbf{u}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j$$

from which we want to cancel out everything but $\mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j$.

Let us take a look at $d_{i,j}$. Given that IPFE is a correct scheme for $\mathcal{F}_{\text{IP}}^n$ we have that

$$\begin{aligned} d_{i,j} &= \left(\mathbf{w}_{i,j} + \begin{pmatrix} c_j \mathbf{ct}_{\mathbf{x}_i} \\ c_j \mathbf{x}_i \end{pmatrix} \right)^\top \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix} \\ &= \mathbf{w}_{i,j}^\top \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix} + c_j \mathbf{x}_i^\top (\mathbf{u}_j^\top \mathbf{F}_{j,i}) + c_j c_i \mathbf{u}_i^\top (\mathbf{u}_j^\top \mathbf{F}_{j,i}) + c_j \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{u}_j \\ &= \mathbf{w}_{i,j}^\top \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix} + c_j \mathbf{u}_j^\top \mathbf{F}_{j,i} \mathbf{x}_i + c_j \mathbf{u}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i + \mathbf{x}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j. \end{aligned}$$

Note that only the third term of $\mathbf{ct}_{\mathbf{x}_i} \mathbf{F}_{i,j} \mathbf{ct}_{\mathbf{x}_j}$ will cancel out, but at the same time the second and fourth terms of $\mathbf{ct}_{\mathbf{x}_j} \mathbf{F}_{j,i} \mathbf{ct}_{\mathbf{x}_i}$ appear. This means that by adding over all $i, j \in [\ell]$ all cancels out and we get

$$\sum_{i,j \in [\ell]} \mathbf{ct}_{\mathbf{x}_i}^\top \mathbf{F}_{i,j} \mathbf{ct}_{\mathbf{x}_j} - d_{i,j} = \sum_{i,j \in [\ell]} \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j + \sum_{i,j \in \ell} \mathbf{w}_{i,j}^\top \begin{pmatrix} \mathbf{u}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \mathbf{u}_j \end{pmatrix}$$

which by construction of $zk_{\mathbf{F}}$ means that

$$v = \sum_{i,j \in [\ell]} \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j.$$

□

Theorem 6 *The MIQFE scheme described in 3 is one selective multi-input simulation secure, if the underlying inner-product functional encryption scheme is one selective function-hiding simulation secure. In other words, for any PPT adversary \mathcal{A} there exist PPT adversaries \mathcal{B} such that*

$$\text{Adv}_{\text{MIQFE}}^{\text{MI-SIM}}(\mathcal{A}) \leq \ell^2 \cdot \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}) + \frac{\ell}{p}.$$

Proof Let \mathcal{A} be a PPT adversary playing the 1-SEL-SIM security game for MIQFE, and let $\kappa \in \mathbb{N}$ be a security parameter. We will prove the result through a series of games Game i for $i \in \{0, 1, 2, 3, 4\}$, defined in Figure 1 with changes in Game $i + 1$ being over Game i . We show that $\text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa) = \text{Game 0} \approx_s \text{Game 1} \approx_s \text{Game 2} \approx_s \text{Game 3} \approx_c \text{Game 4} = \text{Exp}_{\mathcal{A}, \mathcal{S}}^{\text{ideal}}(1^\kappa)$. Let also $\text{IPFE.Sim} = (\text{IPFE.SetupSim}, \text{IPFE.EncSim}, \text{IPFE.KeyGenSim})$ be the simulator for function-hiding 1-SEL-SIM for the IPFE scheme.

Let \mathcal{C}' be a challenger that chooses $b \in \{0, 1\}$ uniformly at random. If $b = 0$ it interacts with a PPT adversary \mathcal{A}' as in Game i , otherwise it interacts as in Game $i + 1$. At the end of the interaction, \mathcal{A}' will make its guess $\tilde{b} \in \{0, 1\}$. We define (for $i = 0, 1, 2, 3$) $\text{Adv}_{i(i+1)}(\mathcal{A}') := \left| \Pr[\tilde{b} = 1 | b = 0] - \Pr[\tilde{b} = 1 | b = 1] \right|$.

Game 0.

This is the real experiment for 1-SEL-SIM security for MIQFE.

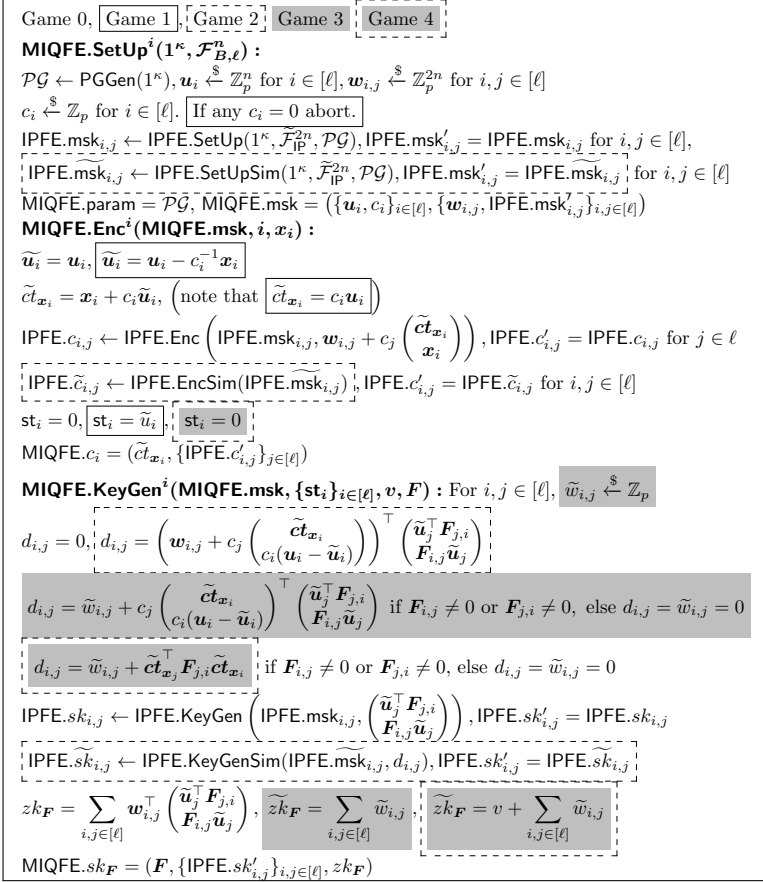


Fig. 1 Games for the security proof of the MIQFE scheme from Encryption Scheme 3.

Game 1.

In this Game we are changing the ciphertext $\tilde{c} \mathbf{x}_i$ to $c_i \mathbf{u}_i$ which is random, and then change \mathbf{u}_i to $\tilde{\mathbf{u}}_i := \mathbf{u}_i - c_i^{-1} \mathbf{x}_i$ in the rest of algorithms to maintain coherence. Also, since we need $c_i \neq 0$, we will abort **Setup** whenever this is not satisfied. Then, distinguishing Games is distinguishing between \mathbf{u}_i and $\tilde{\mathbf{u}}_i$.

First we show that coherence is kept. Indeed,

$$\begin{aligned}
[\tilde{v}]_T &= \left(\sum_{i,j \in [\ell]} [\tilde{c} \mathbf{x}_i \mathbf{F}_{i,j} \tilde{c} \mathbf{x}_j]_T - [d_{i,j}]_T \right) + [zk_{\mathbf{F}}]_T \\
&= \left(\sum_{i,j \in [\ell]} [c_i \mathbf{u}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j]_T - \left[(\mathbf{w}_{i,j} + c_j \begin{pmatrix} \tilde{c} \mathbf{x}_i \\ \mathbf{x}_i \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix}) \right]_T \right) \\
&\quad + \left[\sum_{i,j \in [\ell]} \mathbf{w}_{i,j}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} \right]_T
\end{aligned}$$

$$\begin{aligned}
&= \sum_{i,j \in [\ell]} \left[c_i \mathbf{u}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j \right]_T - [c_j c_i \mathbf{u}_i^\top (\mathbf{u}_j - c_j^{-1} \mathbf{x}_j)^\top \mathbf{F}_{j,i}]_T \\
&\quad - [c_j \mathbf{x}_j^\top \mathbf{F}_{i,j} (\mathbf{u}_j - c_j^{-1} \mathbf{x}_j)]_T \\
&= \sum_{i,j \in [\ell]} \left[c_i \mathbf{u}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j \right]_T - [c_j \mathbf{u}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i]_T + [\mathbf{x}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i]_T \\
&\quad - [\mathbf{x}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j]_T + [\mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j]_T \\
&= \left[\sum_{i,j \in [\ell]} \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j \right]_T .
\end{aligned}$$

Then, for the change in c_i , the probability to abort during **SetUp** is $1/p$ for all $i \in [\ell]$ independently. Then, as long as \mathbf{u}_i and $\tilde{\mathbf{u}}_i$ are indistinguishable, so will be Game 0 and Game 1. Now, $\tilde{\mathbf{u}}_i$ will exist as long as c_i has an inverse, which it will since $c_i \neq 0$, and given that \mathbf{u}_i is sampled uniformly at random in \mathbb{Z}_p^n and used only once (we are proving one selective security), then \mathbf{u}_i and $\tilde{\mathbf{u}}_i$ are computationally indistinguishable. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{01}(\mathcal{A}') \leq \ell/p$.

Game 2.

In this Game we are substituting the IPFE algorithms (IPFE.SetUp, IPFE.Enc, IPFE.KeyGen) used to compute $d_{i,j}$ which we do not modify, by their corresponding simulators (IPFE.SetUpSim, IPFE.EncSim, IPFE.KeyGenSim), so distinguishing between games would imply an adversary breaking 1-SEL-SIM security for IPFE.

More formally, we prove in Lemma 7 that for any PPT adversary \mathcal{A}' , there exists a PPT adversary \mathcal{B} such that $\text{Adv}_{12}(\mathcal{A}') \leq \ell^2 \cdot \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B})$. The intuition is that through the use of a Hybrid argument, we swap in the simulators for every $d_{i,j}$ with $i, j \in [\ell]$.

Game 3.

In this Game we change the construction of $d_{i,j}$, more specifically we swap $\mathbf{w}_{i,j}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix}$ for a random value $\tilde{w}_{i,j}$, as long as $\mathbf{F}_{j,i} \neq \mathbf{0}$ or $\mathbf{F}_{i,j} \neq \mathbf{0}$. Otherwise we keep the value at 0. First we show that coherence is held. Indeed,

$$\begin{aligned}
[z\mathbf{k}_{\mathbf{F}}]_T - \sum_{i,j \in [\ell]} [d_{i,j}]_T &= \\
&= \left[\sum_{i,j \in [\ell]} \mathbf{w}_{i,j}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} - \sum_{i,j \in [\ell]} \left(\mathbf{w}_{i,j} + c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{t}\mathbf{x}_i \\ c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) \end{pmatrix} \right)^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} \right]_T \\
&= \left[\sum_{i,j \in [\ell]} c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{t}\mathbf{x}_i \\ c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} \right]_T , \\
[\tilde{z}\mathbf{k}_{\mathbf{F}}]_T - \sum_{i,j \in [\ell]} [\tilde{d}_{i,j}]_T &= \left[\sum_{i,j \in [\ell]} \tilde{w}_{i,j} - \sum_{i,j \in [\ell]} \tilde{w}_{i,j} + c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{t}\mathbf{x}_i \\ c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} \right]_T
\end{aligned}$$

$$= \left[\sum_{i,j \in [\ell]} c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{x}_i \\ c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} \right]_T.$$

Then, since each $\mathbf{w}_{i,j}$ is sampled uniformly at random in \mathbb{Z}_p^{2n} , as long as the adversary has access to less than $2n$ different samples of $\mathbf{w}_{i,j}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix}$, it is still indistinguishable from random due to the remaining entropy of $\mathbf{w}_{i,j}$. This is an argument used in [46] to show that their multi-input inner-product scheme is simulation sound. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{23}(\mathcal{A}') = 0$.

Game 4.

In this Game we finish the simulation by changing one last time the construction of $d_{i,j}$. More specifically we change $d_{i,j}$ from $\tilde{w}_{i,j} + c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{x}_i \\ c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix}$ to $\tilde{w}_{i,j} + \tilde{\mathbf{c}}\mathbf{x}_j^\top \mathbf{F}_{j,i} \tilde{\mathbf{c}}\mathbf{x}_i$ and modify $zk_{\mathbf{F}}$ from $\sum_{i,j \in [\ell]} \tilde{w}_{i,j}$ to $v + \sum_{i,j \in [\ell]} \tilde{w}_{i,j}$ to maintain coherence. Notably, it is in this step where the function-hiding property of the underlying IPFE scheme is relevant since we can run the key generation simulator only knowing the desired output, and no other information about the linear function. It is also in this change where the “interweaving” of the IPFE ciphertexts commented in the technical overview can be seen.

Firstly, we show that coherence is held. Indeed, in Game 3 we have

$$\tilde{zk}_{\mathbf{F}}^{(3)} = \sum_{i,j \in [\ell]} \tilde{d}_{i,j}^{(3)} - c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{x}_i \\ \mathbf{x}_i \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix}$$

since $c_i(\mathbf{u}_i - \tilde{\mathbf{u}}_i) = \mathbf{x}_i$ by definition of $\tilde{\mathbf{u}}_i$, and in Game 4 we get

$$\begin{aligned} & \sum_{i,j \in [\ell]} \tilde{d}_{i,j}^{(4)} - c_j \begin{pmatrix} \tilde{\mathbf{c}}\mathbf{x}_i \\ \mathbf{x}_i \end{pmatrix}^\top \begin{pmatrix} \tilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \tilde{\mathbf{u}}_j \end{pmatrix} = \\ &= \sum_{i,j \in [\ell]} \tilde{d}_{i,j}^{(4)} - \left((c_j c_i \mathbf{u}_i)^\top (\mathbf{u}_j - c_j^{-1} \mathbf{x}_j)^\top \mathbf{F}_{i,j} + c_j \mathbf{x}_i \mathbf{F}_{i,j} (\mathbf{u}_j - c_j^{-1} \mathbf{x}_j) \right) \\ &= \sum_{i,j \in [\ell]} \tilde{d}_{i,j}^{(4)} - \left(c_j \mathbf{u}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i - \mathbf{x}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i + \mathbf{x}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j - \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j \right) \\ &= \sum_{i,j \in [\ell]} \tilde{w}_{i,j} + \tilde{\mathbf{c}}\mathbf{x}_j^\top \mathbf{F}_{j,i} \tilde{\mathbf{c}}\mathbf{x}_i - \left(\tilde{\mathbf{c}}\mathbf{x}_j^\top \mathbf{F}_{j,i} \tilde{\mathbf{c}}\mathbf{x}_i - \mathbf{x}_i^\top \mathbf{F}_{i,j} \mathbf{x}_j \right) \\ &= v + \sum_{i,j \in [\ell]} \tilde{w}_{i,j} \\ &= \tilde{zk}_{\mathbf{F}}^{(4)}. \end{aligned}$$

It is in this equality that we see the “interweaving” at work, since $\mathbf{x}_j^\top \mathbf{F}_{j,i} c_i \mathbf{u}_i$ and $\mathbf{x}_i^\top \mathbf{F}_{i,j} c_j \mathbf{u}_j$ get canceled only because we are adding for all $i, j \in [\ell]$. Then, as long as $\tilde{d}_{i,j}^{(3)}$ and $\tilde{d}_{i,j}^{(4)}$ are indistinguishable, then $\tilde{zk}_{\mathbf{F}}^{(3)}$ and $\tilde{zk}_{\mathbf{F}}^{(4)}$ are also indistinguishable. To complete the argument, we note that since $\tilde{w}_{i,j}$ is sampled uniformly at random, then $\tilde{d}_{i,j}^{(3)}$ is indistinguishable from $\tilde{w}_{i,j}$ which in turn is indistinguishable from $\tilde{d}_{i,j}^{(4)}$. All in all, for any PPT adversary \mathcal{A}' , $\text{Adv}_{34}(\mathcal{A}') = 0$.

Hybrid H_η^ι MIQFE.Setupi ($1^\kappa, \mathcal{F}_{B,\ell}^n$) : $\mathcal{P}\mathcal{G} \leftarrow \text{PGGen}(1^\kappa), \mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^n$ for $i \in [\ell], \mathbf{w}_{i,j} \xleftarrow{\$} \mathbb{Z}_p^{2n}$ for $i, j \in [\ell], c_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [\ell]$. If any $c_i = 0$ abort. -if $(i, j) <_L (\eta, \iota)$: $\text{IPFE.msk}'_{i,j} \leftarrow \text{IPFE.SetupSim}(1^\kappa, \widetilde{\mathcal{F}}_{\text{IP}}^{2n}, \mathcal{P}\mathcal{G})$, -if $(i, j) = (\eta, \iota)$: $\boxed{\text{IPFE.msk}'_{i,j} \leftarrow \text{IPFE.SetupSim}(1^\kappa, \widetilde{\mathcal{F}}_{\text{IP}}^{2n}, \mathcal{P}\mathcal{G})}$, -if $(i, j) >_L (\eta, \iota)$: $\text{IPFE.msk}'_{i,j} \leftarrow \text{IPFE.Setup}(1^\kappa, \widetilde{\mathcal{F}}_{\text{IP}}^{2n}, \mathcal{P}\mathcal{G})$, $\text{MIQFE.msk} = (\{\mathbf{u}_i, c_i\}_{i \in [\ell]}, \{\mathbf{w}_{i,j}, \text{IPFE.msk}'_{i,j}\}_{i,j \in [\ell]})$ MIQFE.Enci (MIQFE.msk , i, \mathbf{x}_i) : $\widetilde{\mathbf{u}}_i = \mathbf{u}_i - c_i^{-1} \mathbf{x}_i, \widetilde{c}_{\mathbf{x}_i} = c_i \mathbf{u}_i, \text{st}_i = \widetilde{\mathbf{u}}_i$ -if $(i, j) <_L (\eta, \iota)$: $\text{IPFE.c}'_{i,j} \leftarrow \text{IPFE.EncSim}(\text{IPFE.msk}_{i,j})$ -if $(i, j) = (\eta, \iota)$: $\boxed{\text{IPFE.c}'_{i,j} \leftarrow \text{IPFE.EncSim}(\text{IPFE.msk}_{i,j})}$ -if $(i, j) >_L (\eta, \iota)$: $\text{IPFE.c}'_{i,j} \leftarrow \text{IPFE.Enc}\left(\text{IPFE.msk}_{i,j}, \mathbf{w}_{i,j} + \begin{pmatrix} c_j \widetilde{c}_{\mathbf{x}_i} \\ c_j \mathbf{x}_i \end{pmatrix}\right)$ $\text{MIQFE.c}_i = (\widetilde{c}_{\mathbf{x}_i}, \{\text{IPFE.c}'_{i,j}\}_{j \in [\ell]})$ MIQFE.KeyGeni (MIQFE.msk , $\{\text{st}_i\}_{i \in [\ell]}, \mathbf{v}, \mathbf{F}$) : For $i, j \in [\ell]$ $d_{i,j} = \left(\mathbf{w}_{i,j} + \begin{pmatrix} c_j \widetilde{c}_{\mathbf{x}_i} \\ c_j c_i (\mathbf{u}_i - \widetilde{\mathbf{u}}_i) \end{pmatrix} \right)^\top \begin{pmatrix} \widetilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \widetilde{\mathbf{u}}_j \end{pmatrix}, z_{k_{\mathbf{F}}} = \sum_{i,j \in [\ell]} \mathbf{w}_{i,j}^\top \begin{pmatrix} \widetilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \widetilde{\mathbf{u}}_j \end{pmatrix}$ -if $(i, j) <_L (\eta, \iota)$: $\text{IPFE.sk}'_{i,j} \leftarrow \text{IPFE.KeyGenSim}(\text{IPFE.msk}_{i,j}, d_{i,j})$ -if $(i, j) = (\eta, \iota)$: $\boxed{\text{IPFE.sk}'_{i,j} \leftarrow \text{IPFE.KeyGenSim}(\text{IPFE.msk}_{i,j}, d_{i,j})}$ -if $(i, j) >_L (\eta, \iota)$: $\text{IPFE.sk}'_{i,j} \leftarrow \text{IPFE.KeyGen}\left(\text{IPFE.msk}_{i,j}, \begin{pmatrix} \widetilde{\mathbf{u}}_j^\top \mathbf{F}_{j,i} \\ \mathbf{F}_{i,j} \widetilde{\mathbf{u}}_j \end{pmatrix}\right)$ $\text{MIQFE.sk}_{\mathbf{F}} = (\mathbf{F}, \{\text{IPFE.sk}'_{i,j}\}_{i,j \in [\ell]}, z_{k_{\mathbf{F}}})$

Fig. 2 Games for the hybrid argument in Lemma 7. Changes are squared.

Finally, adding it all up, and considering that Game 0 is the real experiment and Game 4 is the ideal experiment, we get

$$\begin{aligned} \text{Adv}_{\text{MIQFE}}^{\text{MI-SIM}}(\mathcal{A}) &= \text{Adv}_{01}(\mathcal{A}) + \text{Adv}_{12}(\mathcal{A}) + \text{Adv}_{23}(\mathcal{A}) + \text{Adv}_{34}(\mathcal{A}) \\ &\leq \ell^2 \cdot \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}) + \frac{\ell}{p}. \end{aligned}$$

□

3.3 Proof of Auxiliary Lemma

Lemma 7 For any PPT adversary \mathcal{A}' , there exists a PPT adversary \mathcal{B} such that

$$\text{Adv}_{12}(\mathcal{A}') \leq \ell^2 \cdot \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}).$$

Proof We will prove the result through a series of Hybrid Games H_η^ι for $\iota \in [\ell]$ and $\eta \in [\ell]$, defined in Figure 2. For ease of notation we define the *lexicographical order* ($\mathbb{N}^2, <_L$) where $(a, b) <_L (c, d)$ if and only if $a < c$ or $a = c$ and $b < d$, analogously, $(a, b) >_L (c, d)$ if and only if $a > c$ or $a = c$ and $b > d$. We show that Game 1 $\approx_c H_1^1 \approx_c \dots \approx_c H_1^\ell \approx_c H_2^1 \approx_c \dots \approx_c H_\ell^\ell = \text{Game 2}$.

Let \mathcal{C}' be a challenger that chooses $b \in \{0, 1\}$ uniformly at random. If $b = 0$ it interacts with a PPT adversary \mathcal{A}' as in Hybrid $H_\eta^{\iota-1}$, otherwise it interacts as in Hybrid H_η^ι . At the end of the interaction, \mathcal{A}' will make its guess $\tilde{b} \in \{0, 1\}$. We define $\text{Adv}_{(\iota-1)\iota}^H(\mathcal{A}') := |\Pr[\tilde{b} = 1 | b = 0] - \Pr[\tilde{b} = 1 | b = 1]|$ for $\iota \in [\ell]$. We define analogously $\text{Adv}_{(\iota-1)\iota}^\rho(\mathcal{A}')$ for distinguishing between Hybrid $H_{\iota-1}^2$ and Hybrid H_ι^0 .

Formally speaking, going from hybrid $H_\eta^{\ell-1}$ to hybrid H_η^ℓ we are swapping the IPFE algorithms by their respective simulators. First, we note that coherence is held since the output of the decryption algorithm $d_{i,j}$ is the same in both hybrids given that $c_\eta(\mathbf{u}_\eta - \tilde{\mathbf{u}}_\eta) = \mathbf{x}_\eta$ by definition of $\tilde{\mathbf{u}}_\eta$. Furthermore, the plaintext $\mathbf{w}_{i,j} + \begin{pmatrix} c_j \tilde{\mathbf{c}} \mathbf{x}_i \\ c_j \mathbf{x}_i \end{pmatrix}$ is indistinguishable from random since $\mathbf{w}_{i,j}$ is sampled uniformly at random and used only once (we are proving one selective security).

Then, distinguishing the simulators in the context of the MIQFE or by their own has the same advantage, so for any PPT adversary \mathcal{A}' trying to distinguish $H_\eta^{\ell-1}$ and H_η^ℓ we can construct a PPT adversary \mathcal{B} which distinguishes the real experiment from the ideal experiment in the function-hiding simulation security game for IPFE. Therefore, we get that for any PPT adversary \mathcal{A}' there exists a PPT adversary \mathcal{B} such that $\text{Adv}_{(\ell-1)\ell}^H(\mathcal{A}') \leq \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B})$.

To finalize the proof we note that going from hybrid $H_{\eta-1}^\ell$ to hybrid H_η^1 and going from Game 1 to hybrid H_1^1 are analogous to going from hybrid $H_\eta^{\ell-1}$ to H_η^ℓ as well as the fact that Game 2 is exactly the same as hybrid H_ℓ^ℓ . Since there are ℓ^2 relevant changes we get

$$\begin{aligned} \text{Adv}_{12}(\mathcal{A}) &= \ell^2 \cdot \text{Adv}_{(\ell-1)\ell}^H(\mathcal{A}) \\ &\leq \ell^2 \cdot \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}). \end{aligned}$$

□

4 Function-Hiding Inner-Product Functional Encryption Instantiation

In the previous section we give an instantiation of a one selective multi-input simulation secure MIQFE based on a one selective function-hiding simulation secure IPFE scheme. As said in Section 1.2, such a scheme does not exist in the standard model, so in this section we give a function-hiding inner-product functional encryption scheme for bounded-norm inner-product functionalities and prove it to be one selective function-hiding simulation secure. As explained in Section 1.3, we will construct the scheme by layering two instances of the scheme [8, Section 3].

We will first describe the family of functions we want to cover. Let $\mathcal{F}_{\text{IP}}^n : \llbracket 0, B \rrbracket^n \rightarrow \llbracket 0, n \cdot B^2 \rrbracket$ be the family of functions such that a function $\mathbf{y} \in \mathcal{F}_{\text{IP}}^n$ is defined by a vector in $\llbracket 0, B \rrbracket$ and applied to \mathbf{z} gives $\mathbf{y}(\mathbf{z}) := \mathbf{z}^\top \mathbf{y}$.

4.1 Description of the Scheme

We define our scheme for the family of functions $\mathcal{F}_{\text{IP}}^n$ as defined above as follows.

Encryption Scheme 4 (Function-hiding IPFE Scheme)

- **IPFE.Setup**($1^\kappa, \mathcal{F}_{\text{IP}}^n$) : Sample $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \text{PGGen}(1^\kappa)$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{n+1}$ and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^n$. Output $\text{IPFE.param} = \mathcal{PG}$ and $\text{IPFE.msk} = (\mathbf{u}, \mathbf{v})$.

- **IPFE.Enc(IPFE.msk, \mathbf{x})** : Sample $c \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$ct_1 := [c]_1 \in \mathbb{G}_1; ct_2 := \left[\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \mathbf{u} \right]_1 \in \mathbb{G}_1^{n+1}$$

Output $\text{IPFE}.c_{\mathbf{x}} := (ct_1, ct_2)$.

- **IPFE.KeyGen(IPFE.msk, \mathbf{y})** : Sample $t \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$sk_1 := \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2 \in \mathbb{G}_2; sk_2 := \left[\begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2 \in \mathbb{G}_2^{n+1}$$

Output $\text{IPFE}.sk_{\mathbf{y}} := (sk_1, sk_2)$.

- **IPFE.Dec(IPFE. $c_{\mathbf{x}}$, IPFE. $sk_{\mathbf{y}}$)** : Compute $[v]_T := e(ct_1, sk_1) + e(ct_2, sk_2)$.
Output $\log([v]_T)$ if $v \in \llbracket 0, n \cdot B^2 \rrbracket$ and \perp otherwise.

Remark 2 Note that this scheme is easily transformable into a scheme for the family of functions $\widetilde{\mathcal{F}}_{\mathbb{P}}^n$ by eliminating the discrete logarithm at the end of decryption and not bounding \mathbf{x} and \mathbf{y} .

4.2 Correctness and Security

Proposition 8 *The IPFE scheme defined in Encryption Scheme 4 is a correct functional encryption scheme for $\mathcal{F}_{\mathbb{P}}^n$.*

Proof The first pairing operation gives us

$$e(ct_1, sk_1) = \left[c \cdot (-\mathbf{u})^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_T,$$

and the second pairing operation gives us

$$\begin{aligned} e(ct_2, sk_2) &= \left[\left(\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \mathbf{u} \right)^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_T \\ &= \left[-t \cdot \mathbf{v}^\top \mathbf{x} + \mathbf{x}^\top \mathbf{y} + t \cdot \mathbf{x}^\top \mathbf{v} + c \cdot \mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_T \\ &= \left[\mathbf{x}^\top \mathbf{y} + c \cdot \mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_T. \end{aligned}$$

This in turn gives us $[v]_T = [\mathbf{x}^\top \mathbf{y}]_T$. □

Theorem 9 *The IPFE scheme described in 4 is one selective function-hiding simulation secure, if the DDH assumption holds in group \mathbb{G}_2 . In other words, for any PPT adversary \mathcal{A} there exists a PPT adversary \mathcal{B} such that*

$$\text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{A}) \leq 2Q_{sk} \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(\mathcal{B}) + \frac{1}{p} + \frac{2Q_{sk}}{p-1}.$$

where Q_{sk} denotes the number of queries performed to KeyGen.

Game 0, Game 1 , Game 2 IPFE.Setup ^{i} ($1^\kappa, \mathcal{F}_{\text{IP}}^n$) : $\mathcal{P}\mathcal{G} \leftarrow \text{PGGen}(1^\kappa), \mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{n+1}, \mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^n, c \xleftarrow{\$} \mathbb{Z}_p, \text{ if } c = 0 \text{ abort}$ $\text{IPFE.param} = \mathcal{P}\mathcal{G}, \text{IPFE.msk} = (\mathbf{u}, \mathbf{v}, [c])$ IPFE.Enc ^{i} (IPFE.msk , \mathbf{x}) : $c \xleftarrow{\$} \mathbb{Z}_p$ $ct_1 = [c]_1, ct_2 = \left[\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \mathbf{u} \right]_1, [ct_2] = [c \cdot \mathbf{u}]_1$ $\text{IPFE.c}_\mathbf{x} = (ct_1, ct_2)$ IPFE.KeyGen ^{i} (IPFE.msk , $\mathbf{x}^\top \mathbf{y}, \mathbf{y}$) : $t \xleftarrow{\$} \mathbb{Z}_p$ $sk_1 = \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2, [sk_1] = \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} + c^{-1} \cdot \mathbf{x}^\top \mathbf{y} \right]_2, [sk_1] = \left[-\mathbf{u}^\top \begin{pmatrix} t \\ t \cdot \mathbf{v} \end{pmatrix} + c^{-1} \cdot \mathbf{x}^\top \mathbf{y} \right]_2$ $sk_2 = \left[\begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2, [sk_2] = \left[\begin{pmatrix} t \\ t \cdot \mathbf{v} \end{pmatrix} \right]_2$ $\text{IPFE.sk}_\mathbf{y} = (sk_1, sk_2)$
--

Fig. 3 Games for the security proof of the IPFE from Encryption Scheme 4.

Proof Let \mathcal{A} be a PPT adversary playing the function-hiding 1-SEL-SIM security game for IPFE, and let $\kappa \in \mathbb{N}$ be a security parameter. We will prove the result through a series of games Game i for $i \in \{0, 1, 2\}$, defined in Figure 3 with changes in Game $i + 1$ being over Game i . We show that $\text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa) = \text{Game 0} \approx_s \text{Game 1} \approx_c \text{Game 2} = \text{Exp}_{\mathcal{A}, \mathcal{S}}^{\text{ideal}}(1^\kappa)$.

Let \mathcal{C}' be a challenger that chooses $b \in \{0, 1\}$ uniformly at random. If $b = 0$ it interacts with a PPT adversary \mathcal{A}' as in Game i , otherwise it interacts as in Game $i + 1$. At the end of the interaction, \mathcal{A}' will make its guess $\tilde{b} \in \{0, 1\}$. For $i = 0, 1$, we define $\text{Adv}_{i(i+1)}(\mathcal{A}') := \left| \Pr[\tilde{b} = 1 | b = 0] - \Pr[\tilde{b} = 1 | b = 1] \right|$.

Game 0.

This is the real experiment for FH 1-SEL-SIM security for IPFE.

Game 1.

In this Game we change the construction of ct_2 from $\left[\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \mathbf{u} \right]_1$ to $[c \cdot \mathbf{u}]$ and to keep coherence we must change sk_1 from we must change $sk_1 = \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + \mathbf{V}t \end{pmatrix} \right]_2$ to $\left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + \mathbf{V}t \end{pmatrix} + c^{-1} \cdot \mathbf{x}^\top \mathbf{y} \right]_2$. To be able to make this change we move the sampling of c to the Setup and abort if $c = 0$.

This change can also be seen as swapping \mathbf{u} for $\tilde{\mathbf{u}} := \mathbf{u} - c^{-1} \cdot \begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix}$, since

$$\begin{aligned} \left[\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \tilde{\mathbf{u}} \right]_1 &= \left[\begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} + c \cdot \left(\mathbf{u} - c^{-1} \cdot \begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} \right) \right]_1 = [c \cdot \mathbf{u}]_1, \\ \left[-\tilde{\mathbf{u}}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2 &= \left[- \left(\mathbf{u} - c^{-1} \cdot \begin{pmatrix} -\mathbf{v}^\top \mathbf{x} \\ \mathbf{x} \end{pmatrix} \right)^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_2 \\ &= \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} - t \cdot c^{-1} \cdot (\mathbf{v}^\top \mathbf{x}) + c^{-1} \cdot \mathbf{x}^\top \mathbf{y} \right]_2 \end{aligned}$$

$$\begin{aligned}
& \left. + t \cdot c^{-1} \cdot \mathbf{x}^\top \mathbf{v} \right]_2 \\
& = \left[-\mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} + c^{-1} \cdot \mathbf{x}^\top \mathbf{y} \right]_2,
\end{aligned}$$

and coherence is indeed held given that

$$\begin{aligned}
e(\tilde{c}t_1, sk_1) &= \left[-c \cdot \mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} + \mathbf{x}^\top \mathbf{y} \right]_T, \\
e(\tilde{c}t_2, sk_2) &= \left[c \cdot \mathbf{u}^\top \begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix} \right]_T
\end{aligned}$$

which gives $[v]_T = [\mathbf{x}^\top \mathbf{y}]_T$ during decryption.

First, for the change in c , since we are proving one selective simulation its sampling can be moved to **SetUp**, and the protocol will abort with probability $1/p$. Then, as long as \mathbf{u} and $\tilde{\mathbf{u}}$ are indistinguishable, so will be Game 0 and Game 1. Now, $\tilde{\mathbf{u}}$ will exist as long as c has an inverse, which it will since $c \neq 0$, and given that \mathbf{u} is sampled uniformly at random in \mathbb{Z}_p^{n+1} and used only once (we are proving one selective security), then \mathbf{u} and $\tilde{\mathbf{u}}$ are indistinguishable. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{01}(\mathcal{A}') \leq 1/p$.

Game 2.

In this Game we finish the simulation by changing in both sk_1 and sk_2 the vector $\begin{pmatrix} t \\ \mathbf{y} + t \cdot \mathbf{v} \end{pmatrix}$ to $\begin{pmatrix} t \\ t \cdot \mathbf{v} \end{pmatrix}$. This means that the encryption simulation can be performed without knowledge of \mathbf{x} and the key generation simulation can be performed only knowing the output $\mathbf{x}^\top \mathbf{y}$.

More formally, we prove in Lemma 10 that for any PPT adversary \mathcal{A}' there exists a PPT adversary \mathcal{B} such that $\text{Adv}_{12}(\mathcal{A}') \leq 2Q_{sk} \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}} + 2Q_{sk}/p - 1$. The intuition is that using a Hybrid argument through the queries asked we use the n -fold DDH assumption (see Appendix 2.2) to swap $t \cdot \mathbf{v}$ for a value sampled uniformly at random so we can remove \mathbf{y} .

Finally, adding it all up, and considering that Game 0 is the real experiment and Game 2 is the ideal experiment, we get

$$\text{Adv}_{\text{PFSE}}^{\text{FH-SIM}}(\mathcal{A}) = \text{Adv}_{01}(\mathcal{A}) + \text{Adv}_{12}(\mathcal{A}) \leq 2Q_{sk} \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(\mathcal{B}) + \frac{1}{p} + \frac{2Q_{sk}}{p-1}.$$

□

4.3 Proof of Auxiliary Lemma

Lemma 10 *For any PPT adversary \mathcal{A}' there exists a PPT adversary \mathcal{B} such that*

$$\text{Adv}_{12}(\mathcal{A}') \leq 2Q_{sk} \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}} + \frac{2Q_{sk}}{p-1}.$$

where Q_{sk} denotes the number of queries performed to **KeyGen**.

Proof We will prove this through a series of Hybrid Games H_ρ^α for $\rho \in [Q_{sk}]$ and $\alpha \in \{0, 1, 2\}$, defined in Figure 4. We show that Game 1 = $H_1^0 \approx_c H_1^1 \approx_s H_1^2 \approx_c H_2^0 \approx_c \dots \approx_c H_{Q_{sk}}^2 \approx_{c=H_{Q_{sk+1}}^0}$ Game 2.

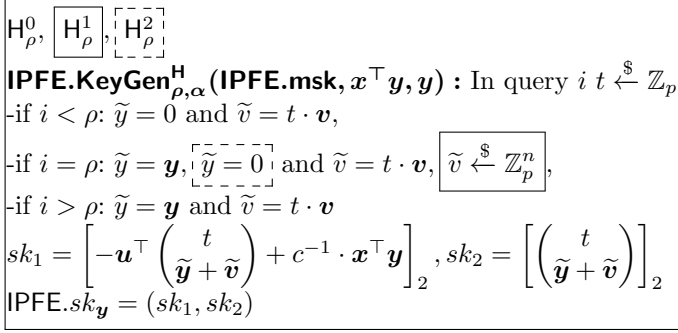


Fig. 4 Games for the hybrid argument in Lemma 10. Changes only occur on the KeyGen algorithm so the others are omitted and assumed the same as in Game 1 in Figure 3.

Let \mathcal{C}' be a challenger that chooses $b \in \{0, 1\}$ uniformly at random. If $b = 0$ it interacts with a PPT adversary \mathcal{A}' as in Hybrid H_ρ^{i-1} , otherwise it interacts as in Hybrid H_ρ^i . At the end of the interaction, \mathcal{A}' will make its guess $\tilde{b} \in \{0, 1\}$. We define $\text{Adv}_{(i-1)i}^\alpha(\mathcal{A}') := |\Pr[\tilde{b} = 1|b = 0] - \Pr[\tilde{b} = 1|b = 1]|$ for $i = 1, 2$. We define analogously $\text{Adv}_{(i-1)i}^\rho(\mathcal{A}')$ for distinguishing between Hybrid H_{i-1}^2 and Hybrid H_i^0 .

Hybrid H_ρ^1 .

In this change we have swapped $\tilde{\mathbf{v}}$ from $t \cdot \mathbf{v}$ to uniformly at random. We show that for any PPT adversary \mathcal{A}' trying to distinguish these two Hybrids we can construct a PPT adversary \mathcal{B} against the n -fold DDH assumption in \mathbb{G}_2 .

When, \mathcal{B} receives the n -fold DDH challenge $[t]_2 = [(t, 1)^\top]_2$, $[\mathbf{W}]_2 := [(\mathbf{w}_1, \mathbf{w}_2)^\top]_2$ for some $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{Z}_p^n$, it runs $\text{IPFE.SetUp}_{\rho,1}^H$ and $\text{IPFE.Enc}_{\rho,1}^H$ and in IPFE.msk it substitutes \mathbf{v} for $[\mathbf{w}_2]_2$. Note that only having the element in \mathbb{G}_2 is not an issue since after Game 1 \mathbf{v} is only used during KeyGen and as such only the values in \mathbb{G}_2 are needed. It is in this step where using our order in layering the schemes from [8, Section 3] instead of the order in [14, Section 6.3] comes into play.

Then for query $i \in [Q_{sk}]$, if $i \neq \rho$ it runs $\text{KeyGen}_{\rho,1}^H$ normally, but if $i = \rho$ it sets $[t]_2$ as the first part of the n -fold DDH challenge and $\tilde{\mathbf{v}} = [\mathbf{w}_1]_2$. Once again, only having the elements in \mathbb{G}_2 is not an issue since the operations can be performed in the group. Finally, it outputs the same bit as adversary \mathcal{A}' . It is clear to see that if the DDH challenge $[\mathbf{W}]_2$ is of the form $[\mathbf{tr}^\top]_2 = [(tr, \mathbf{r})^\top]_2$ for some $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^n$ then it is the same distribution as in Hybrid H_ρ^0 while if $[\mathbf{W}]_2$ is uniformly at random it is the same distribution as in Hybrid H_ρ^1 . As such we get

$$\begin{aligned} \text{Adv}_{01}^\alpha(\mathcal{A}') &\leq \text{Adv}_{\mathbb{G}_2}^{\text{Q-DDH}}(\mathcal{B}) \\ &\leq \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(\mathcal{B}) + \frac{1}{p-1}, \end{aligned}$$

where we have applied the random self-reducibility of DDH, Lemma 1.

Hybrid H_ρ^2 .

In this change we have swapped \mathbf{y} from \mathbf{y} to $\mathbf{0}$. Distinguishing between these Hybrids is distinguishing between $\mathbf{y} + \tilde{\mathbf{v}}$ and $\tilde{\mathbf{v}}$, and since $\tilde{\mathbf{v}}$ is sampled uniformly at random

and only used once they are indistinguishable. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{12}^\alpha(\mathcal{A}') = 0$.

Hybrid $H_{\rho+1}^0$.

In this change we have swapped back \tilde{v} from sampled uniformly at random to $t \cdot \mathbf{v}$. As such this transition is analogous to the change from H_ρ^0 to H_ρ^1 , so for any PPT adversary \mathcal{A}' there exists a PPT adversary \mathcal{B} such that

$$\text{Adv}_{i(i+1)}^\rho(\mathcal{A}') \leq \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(\mathcal{B}) + \frac{1}{p-1}.$$

Finally, adding all the transitions together and noting that H_1^0 is identically distributed to Game 1, while $H_{Q_{sk}+1}^0$ is identically distributed to Game 2, we get

$$\text{Adv}_{12}(\mathcal{A}') \leq 2Q_{sk} \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}} + \frac{2Q_{sk}}{p-1}.$$

where Q_{sk} denotes the number of queries performed to KeyGen. \square

5 Randomized Quadratic Functional Encryption Scheme

In this section we describe our randomized quadratic functional encryption scheme for bounded-norm quadratic functionalities. We first describe the family of functions we want to cover. Let $\mathcal{F}_{Q,B}^n : \llbracket 0, B \rrbracket^n \rightarrow \llbracket 0, n^2 \cdot B^3 \rrbracket$ be the family of quadratic functions such that a function $\mathbf{F} \in \mathcal{F}_{Q,B}^n$ is defined by a matrix in $\llbracket 0, B \rrbracket^{n \times n}$ which applied to $\mathbf{x} \in \llbracket 0, B \rrbracket^n$ gives $\mathbf{x}^\top \mathbf{F} \mathbf{x}$. We then define the family of randomized functions $\hat{\mathcal{F}}_{Q,B,\epsilon}^n : \llbracket 0, B \rrbracket^n \times \mathcal{R} \rightarrow \mathbb{Z}$ such that a function $\hat{\mathbf{F}} \in \hat{\mathcal{F}}_{Q,B,\epsilon}^n$ is defined by a matrix in $\llbracket 0, B \rrbracket^{n \times n}$ and a distribution D_ϵ over \mathbb{Z} which applied to $\mathbf{x} \in \llbracket 0, B \rrbracket^n$ gives $\mathbf{x}^\top \mathbf{F} \mathbf{x} + e(r_{\mathbf{F}})$, where $r_{\mathbf{F}} \leftarrow \mathcal{R}$ is used as a seed to sample $e(r_{\mathbf{F}})$ following the distribution D_ϵ .

To construct our RQFE scheme we use a function-hiding inner-product functional encryption whose family of functions is $\tilde{\mathcal{F}}_{\mathbb{P}}^{2n} : \mathbb{Z}_p^{2n} \rightarrow \mathbb{G}_T$ (for some $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \text{PGGen}(1^\kappa)$). A function $\mathbf{y} \in \tilde{\mathcal{F}}_{\mathbb{P}}^{2n}$ is defined by a vector in \mathbb{Z}_p^{n+m} and applied to \mathbf{z} gives $\mathbf{y}(\mathbf{z}) := [\mathbf{z}^\top \mathbf{y}]_T$.

5.1 Description of the Scheme

Let $\text{IPFE} = (\text{IPFE.Setup}, \text{IPFE.Enc}, \text{IPFE.KeyGen}, \text{IPFE.Dec})$ be a function-hiding inner-product functional encryption scheme for the family of functions $\tilde{\mathcal{F}}_{\mathbb{P}}^{2n}$. Below is a description of our RQFE scheme for the family of functions $\hat{\mathcal{F}}_{Q,B,\epsilon}^n$.

Encryption Scheme 5 (RQFE Scheme)

- **RQFE.Setup** $(1^\kappa, \hat{\mathcal{F}}_{Q,B,\epsilon}^n)$: Sample $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \text{PGGen}(1^\kappa)$ and choose D_ϵ a distribution over \mathbb{Z}_p . Then sample $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^{2n}$

and $c \xleftarrow{\$} \mathbb{Z}_p$. Run $\text{IPFE.msk} \leftarrow \text{IPFE.Setup}(1^\kappa, \tilde{\mathcal{F}}_{\text{IP}}^{2n}, \mathcal{P}\mathcal{G})$. Output

$$\begin{aligned} \text{RQFE.param} &= (\mathcal{P}\mathcal{G}, D_\epsilon) \text{ and} \\ \text{RQFE.msk} &= (\mathbf{u}, \mathbf{w}, c, \text{IPFE.msk}). \end{aligned}$$

- **RQFE.Enc**(RQFE.msk, \mathbf{x}) : Compute

$$\begin{aligned} \mathbf{ct}_x &:= \mathbf{x} + c \cdot \mathbf{u}, \\ \text{IPFE.c} &\leftarrow \text{IPFE.Enc} \left(\text{IPFE.msk}, \mathbf{w} + c \begin{pmatrix} \mathbf{ct}_x \\ \mathbf{x} \end{pmatrix} \right). \end{aligned}$$

Output $\text{RQFE.c}_x = (\mathbf{ct}_x, \text{IPFE.c})$.

- **RQFE.KeyGen**(RQFE.msk, \mathbf{F}) : Sample $r_{\mathbf{F}} \leftarrow \mathcal{R}$ for $e(r_{\mathbf{F}}) \sim D_\epsilon$ and $u'_{\mathbf{F}} \xleftarrow{\$} \mathbb{Z}_p$. Compute

$$\begin{aligned} t'_{\mathbf{F}} &= e(r_{\mathbf{F}}) + u'_{\mathbf{F}}, \\ \text{IPFE.sk} &\leftarrow \text{IPFE.KeyGen} \left(\text{IPFE.msk}, \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F} \mathbf{u} \end{pmatrix} \right), \\ zk_{\mathbf{F}} &\leftarrow \mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F} \mathbf{u} \end{pmatrix} - u'_{\mathbf{F}}. \end{aligned}$$

Output $\text{RQFE.sk}_{\mathbf{F}} = (\mathbf{F}, \text{IPFE.sk}, t'_{\mathbf{F}}, zk_{\mathbf{F}})$.

- **RQFE.Dec**(RQFE.c $_x$, RQFE.sk $_{\mathbf{F}}$) : Compute

$$\begin{aligned} [d]_T &\leftarrow \text{IPFE.Dec}(\text{IPFE.c}, \text{IPFE.sk}) \\ [v]_T &:= [\mathbf{ct}_x^\top \mathbf{F} \mathbf{ct}_x]_T - [d]_T + [t'_{\mathbf{F}} + zk_{\mathbf{F}}]_T \end{aligned}$$

Output $\log([v]_T)$ if $v \in [0, n^2 \cdot B^3 + \alpha]$ and \perp otherwise, where α is such that $\Pr[D_\epsilon > \alpha] = \text{neg}(\kappa)$.

Remark 3 Note that in our RQFE construction, setting the distribution D_ϵ as the zero distribution, the construction reduces exactly to the multi-input construction from Encryption Scheme 3 for the parameter $\ell = 1$, mirroring the result in [6] where their randomized inner-product scheme reduces to the multi-input inner-product scheme of [20, Section 3.1 Figure 4].

5.2 Correctness and Security

Proposition 11 *The RQFE scheme defined in Encryption Scheme 5 is a correct randomized functional encryption scheme for $\tilde{\mathcal{F}}_{\mathbf{Q}, B, \epsilon}^{n, m}$ as long as IPFE is a correct scheme for $\tilde{\mathcal{F}}_{\text{IP}}^{n+m}$.*

Proof For ease of notation and reading we will leave out the $[\cdot]_T$. Let $\mathbf{x} \in \llbracket 0, B \rrbracket^n$ be any plaintext and $\hat{\mathbf{F}}^1, \dots, \hat{\mathbf{F}}^Q \in \hat{\mathcal{F}}_\epsilon^{\ell, X, Y}$ any set of randomized functions. Then for any $i \in [Q]$ we get that $v^i \leftarrow \text{RQFE.Dec}(\text{RQFE.c}_{\mathbf{x}}, \text{RQFE.sk}_{\mathbf{F}^i})$ satisfies the following:

$$\mathbf{ct}_{\mathbf{x}}^\top \mathbf{F}^i \mathbf{ct}_{\mathbf{x}} = \mathbf{x}^\top \mathbf{F}^i \mathbf{x} + c \cdot \mathbf{u}^\top \mathbf{F}^i \mathbf{x} + \mathbf{x}^\top \mathbf{F}^i c \cdot \mathbf{u} + c \cdot \mathbf{u}^\top \mathbf{F}^i c \cdot \mathbf{u}$$

from which we want to cancel out everything but $\mathbf{x}^\top \mathbf{F}^i \mathbf{x}$.

Let us take a look at γ^i (which are all the extra terms apart from $\mathbf{x}^\top \mathbf{F}^i \mathbf{x}$). Given that IPFE is a correct scheme for $\mathcal{F}_{\text{IP}}^{n+m}$ we have that

$$\begin{aligned} d^i &= \left(\mathbf{w} + c \begin{pmatrix} \mathbf{ct}_{\mathbf{x}} \\ \mathbf{x} \end{pmatrix} \right)^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F}^i \\ \mathbf{F}^i \mathbf{u} \end{pmatrix} \\ &= \mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F}^i \\ \mathbf{F}^i \mathbf{y} \end{pmatrix} + c \cdot \mathbf{x}^\top (\mathbf{u}^\top \mathbf{F}^i) + c^2 \cdot \mathbf{u}^\top (\mathbf{u}^\top \mathbf{F}^i) + c \cdot \mathbf{x}^\top \mathbf{F}^i \mathbf{u} \\ &= \mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F}^i \\ \mathbf{F}^i \mathbf{u} \end{pmatrix} + c \cdot \mathbf{u}^\top \mathbf{F}^i \mathbf{x} + c \cdot \mathbf{u}^\top \mathbf{F}^i c \cdot \mathbf{u} + \mathbf{x}^\top \mathbf{F}^i c \cdot \mathbf{u}. \end{aligned}$$

Finally, adding $t'_{\mathbf{F}^i}$ and $z_{\mathbf{F}^i}$ we get

$$\begin{aligned} v^i &= \mathbf{ct}_{\mathbf{x}}^\top \mathbf{F}^i \mathbf{ct}_{\mathbf{x}} - d^i + t'_{\mathbf{F}^i} + z_{\mathbf{F}^i} \\ &= \mathbf{x}^\top \mathbf{F}^i \mathbf{x} + c \cdot \mathbf{u}^\top \mathbf{F}^i \mathbf{x} + \mathbf{x}^\top \mathbf{F}^i c \cdot \mathbf{u} + c \cdot \mathbf{u}^\top \mathbf{F}^i c \cdot \mathbf{u} \\ &\quad - \left(\mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F}^i \\ \mathbf{F}^i \mathbf{u} \end{pmatrix} + c \cdot \mathbf{u}^\top \mathbf{F}^i \mathbf{x} + c \cdot \mathbf{u}^\top \mathbf{F}^i c \cdot \mathbf{u} + \mathbf{x}^\top \mathbf{F}^i c \cdot \mathbf{u} \right) \\ &\quad + e_{\mathbf{F}^i} + u'_{\mathbf{F}^i} + \mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F}^i \\ \mathbf{F}^i \mathbf{u} \end{pmatrix} - u'_{\mathbf{F}^i} \\ &= \mathbf{x}^\top \mathbf{F}^i \mathbf{x} + e(r_{\mathbf{F}^i}). \end{aligned}$$

This is clearly the same as $\hat{\mathbf{F}}^i(\mathbf{x}; r)$ for $r \leftarrow \mathcal{R}$ when $e(r_{\mathbf{F}}) \leq \alpha$ since $e(r_{\mathbf{F}})$ is sampled from D_ϵ independently for every query \mathbf{F} . And since $\Pr[D_\epsilon > \alpha] = \text{neg}(\kappa)$ by definition, then v^i and $\hat{\mathbf{F}}^i(\mathbf{x}; r)$ are computationally indistinguishable. \square

Theorem 12 *The RQFE scheme described in 5 is one selective simulation secure, if the underlying inner-product functional encryption scheme is one selective function-hiding simulation secure. In other words, for any PPT adversary \mathcal{A} there exist PPT adversaries \mathcal{B} such that*

$$\text{Adv}_{\text{RQFE}}^{\text{SIM}}(\mathcal{A}) \leq \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}) + \frac{1}{p}.$$

Proof Let \mathcal{A} be a PPT adversary playing the 1-SEL-SIM security game for RQFE, and let $\kappa \in \mathbb{N}$ be a security parameter. We will prove the result through a series of games Game i for $i \in \{0, 1, 2, 3, 4, 5\}$, defined in Figure 5 with changes in Game $i + 1$ being over Game i . We show that $\text{Exp}_{\mathcal{A}}^{\text{real}}(1^\kappa) = \text{Game } 0 \approx_s \text{Game } 1 \approx_c \text{Game } 2 \approx_s \text{Game } 3 \approx_s \text{Game } 4 \approx_s \text{Game } 5 = \text{Exp}_{\mathcal{A}, \mathcal{S}}^{\text{ideal}}(1^\kappa)$. Let also $\text{IPFE.Sim} = (\text{IPFE.SetupSim}, \text{IPFE.EncSim}, \text{IPFE.KeyGenSim})$ be the simulator for function-hiding 1-SEL-SIM for the IPFE scheme.

Let \mathcal{C}' be a challenger that chooses $b \in \{0, 1\}$ uniformly at random. If $b = 0$ it interacts with a PPT adversary \mathcal{A}' as in Game i , otherwise it interacts as in Game

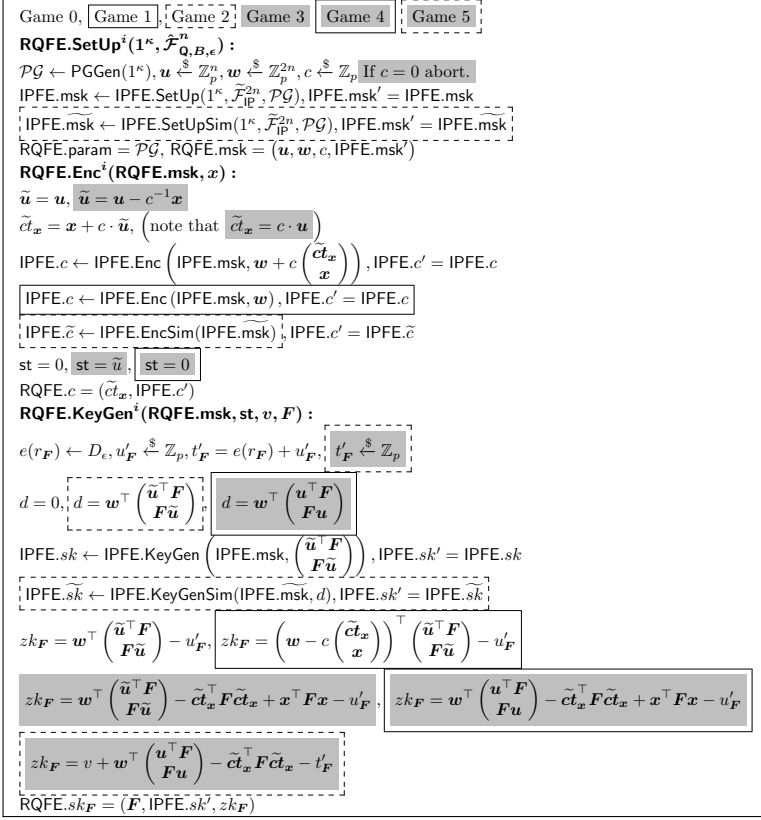


Fig. 5 Games for the security proof of the RQFE scheme from Encryption Scheme 5.

$i + 1$. At the end of the interaction, \mathcal{A}' will make its guess $\tilde{b} \in \{0, 1\}$. We define (for $i = 0, 1, 2, 3, 4$) $\text{Adv}_{i(i+1)}(\mathcal{A}') := \left| \Pr[\tilde{b} = 1 | b = 0] - \Pr[\tilde{b} = 1 | b = 1] \right|$.

Game 0.

This is the real experiment for 1-SEL-SIM security for RQFE.

Game 1.

In this game we simulate the IPFE plaintext, by sampling it by uniformly at random. In other words, we change $\mathbf{w} + c \begin{pmatrix} \tilde{c}\mathbf{x} \\ \mathbf{x} \end{pmatrix}$ for \mathbf{w} during the encryption. Then, to keep coherence, we change $zk_{\mathbf{F}}$ from $\mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F}\mathbf{u} \end{pmatrix} - u'_{\mathbf{F}}$ to $\begin{pmatrix} \mathbf{w} - c \begin{pmatrix} \tilde{c}\mathbf{x} \\ \mathbf{x} \end{pmatrix} \end{pmatrix}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F}\mathbf{u} \end{pmatrix} - u'_{\mathbf{F}}$. Indeed, we get

$$[d]_T = \left[\mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F}\mathbf{u} \end{pmatrix} \right]_T,$$

$$[v]_T = \left[\tilde{c}\mathbf{x}^\top \mathbf{F}\tilde{c}\mathbf{x} - d + t'_{\mathbf{F}} + zk_{\mathbf{F}} \right]_T$$

$$\begin{aligned}
&= \left[\mathbf{ct}_x^\top \mathbf{F} \mathbf{ct}_x - \mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F} \mathbf{u} \end{pmatrix} + t'_F + \left(\mathbf{w} - c \begin{pmatrix} \mathbf{ct}_x \\ \mathbf{x} \end{pmatrix} \right)^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F} \mathbf{u} \end{pmatrix} - u'_F \right]_T \\
&= \left[\mathbf{x}^\top \mathbf{F} \mathbf{x} + e(r_F) \right]_T.
\end{aligned}$$

Then, distinguishing between Game 0 and Game 1 is the same as distinguishing between $\mathbf{w} + c \begin{pmatrix} \mathbf{ct}_x \\ \mathbf{x} \end{pmatrix}$ and \mathbf{w} . Since \mathbf{w} is sampled uniformly at random in \mathbb{Z}_p^{2n} and only used once (we are proving one selective security), these two vectors are statistically indistinguishable. As such, for any PPT adversary \mathcal{A}' , $\text{Adv}_{01}(\mathcal{A}') = 0$.

Game 2.

In this game we are substituting the IPFE algorithms (IPFE.Setup, IPFE.Enc, IPFE.KeyGen) used to compute d by their corresponding simulators (IPFE.SetupSim, IPFE.EncSim, IPFE.KeyGenSim). Then, distinguishing between games would imply an adversary breaking 1-SEL-SIM security for IPFE. More formally, for any PPT adversary \mathcal{A}' , there exists a PPT adversary \mathcal{B} such that $\text{Adv}_{12}(\mathcal{A}') \leq \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B})$.

Game 3.

In this game we simulate the ciphertext by swapping \mathbf{ct}_x for $c \cdot \mathbf{u}$ which is random. To do so while keeping coherence, Game 3 is a rewriting of Game 2 replacing \mathbf{u} for $\tilde{\mathbf{u}} := \mathbf{u} - c^{-1} \mathbf{x}$ where we abort the Setup if $c \neq 0$. Indeed,

$$\begin{aligned}
\tilde{\mathbf{ct}}_x &:= \mathbf{x} + c \cdot \tilde{\mathbf{u}} = \mathbf{x} + c(\mathbf{u} - c^{-1} \mathbf{x}) = c \cdot \mathbf{u}, \\
zk_F &= \left(\mathbf{w} - c \begin{pmatrix} \tilde{\mathbf{ct}}_x \\ \mathbf{x} \end{pmatrix} \right)^\top \begin{pmatrix} \tilde{\mathbf{u}}^\top \mathbf{F} \\ \mathbf{F} \tilde{\mathbf{u}} \end{pmatrix} - u'_F \\
&= \mathbf{w}^\top \begin{pmatrix} \tilde{\mathbf{u}}^\top \mathbf{F} \\ \mathbf{F} \tilde{\mathbf{u}} \end{pmatrix} - \left(c(\mathbf{u} - c^{-1} \mathbf{x})^\top \mathbf{F} c \cdot \mathbf{u} + \mathbf{x}^\top \mathbf{F} c(\mathbf{u} - c^{-1} \mathbf{x}) \right) - u'_F \\
&= \mathbf{w}^\top \begin{pmatrix} \tilde{\mathbf{u}}^\top \mathbf{F} \\ \mathbf{F} \tilde{\mathbf{u}} \end{pmatrix} - \left(c \cdot \mathbf{u}^\top \mathbf{F} c \cdot \mathbf{u} - \mathbf{x}^\top \mathbf{F} c \cdot \mathbf{u} + \mathbf{x}^\top \mathbf{F} c \cdot \mathbf{u} - \mathbf{x}^\top \mathbf{F} \mathbf{x} \right) \\
&= \mathbf{w}^\top \begin{pmatrix} \tilde{\mathbf{u}}^\top \mathbf{F} \\ \mathbf{F} \tilde{\mathbf{u}} \end{pmatrix} - \tilde{\mathbf{ct}}_x^\top \mathbf{F} \tilde{\mathbf{ct}}_x + \mathbf{x}^\top \mathbf{F} \mathbf{x} - u'_F
\end{aligned}$$

As such, for the change in c , the probability to abort during Setup is $1/p$. Then, as long as \mathbf{u} and $\tilde{\mathbf{u}}$ are indistinguishable, so will be Game 0 and Game 1. Now, $\tilde{\mathbf{u}}$ will exist as long as c has an inverse, which it will since $c \neq 0$, and given that \mathbf{u} is sampled uniformly at random in \mathbb{Z}_p^n and used only once (we are proving one selective security), then \mathbf{u} and $\tilde{\mathbf{u}}$ are statistically indistinguishable. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{23}(\mathcal{A}') \leq 1/p$.

Game 4.

In this game we change back d from $\mathbf{w}^\top \begin{pmatrix} \tilde{\mathbf{u}}^\top \mathbf{F} \\ \mathbf{F} \tilde{\mathbf{u}} \end{pmatrix}$ to $\mathbf{w}^\top \begin{pmatrix} \mathbf{u}^\top \mathbf{F} \\ \mathbf{F} \mathbf{u} \end{pmatrix}$, and to keep coherence we do the same change in zk_F . It is clear that given an adversary that distinguishes between the values that have been changed, then we can construct an adversary that distinguishes \mathbf{u} from $\tilde{\mathbf{u}}$. As we discussed in Game 3, \mathbf{u} is sampled uniformly at random in \mathbb{Z}_p^n and used only once, so \mathbf{u} and $\tilde{\mathbf{u}}$ are statistically indistinguishable. As such, for any PPT adversary \mathcal{A}' , $\text{Adv}_{34}(\mathcal{A}') = 0$.

Table 5 Efficiency estimates for our MIQFE and IPFE constructions.

	Secret key	Ciphertext	Functional key
Generic MIQFE	$\ell^2 \cdot \text{IPFE}_{\text{msk}}^{2n} + \ell(1+n) p + \ell^2 2n p $	$\ell \cdot \text{IPFE}_{c_x}^{2n} + n p $	$\ell^2 \cdot \text{IPFE}_{sk_y}^{2n} + p $
Generic RQFE	$\text{IPFE}_{\text{msk}}^{2n} + (3n+1) p $	$\text{IPFE}_{c_x}^{2n} + n p $	$\text{IPFE}_{sk_y}^{2n} + 2 p $
IPFE	$(2n+1) p $	$(n+2) \mathbb{G}_1 $	$(n+2) \mathbb{G}_2 $
Concrete MIQFE	$\ell^2(4n+1) p + \ell(1+n) p + \ell^2 2n p $	$\ell(2n+2) \mathbb{G}_1 + n p $	$\ell^2 \cdot (2n+2) \mathbb{G}_2 + p $
Concrete RQFE	$(7n+1) p $	$(2n+2) \mathbb{G}_1 + n p $	$(2n+2) \mathbb{G}_2 + 2 p $

Game 5.

In this game we finish the simulation by sampling t'_F uniformly at random and incorporating the noise from D_ϵ from the value $v = \mathbf{x}^\top \mathbf{F} \mathbf{x} + e(r_F)$, the inherent leakage of the scheme. Coherence is indeed held, since $zk_F + t'_F$ remains the same. This means that distinguishing between Game 4 and Game 5 is distinguishing $e(r_F) + u'_F$ from uniformly at random. Since u'_F is sampled uniformly in \mathbb{Z}_p both games are indistinguishable. Therefore, for any PPT adversary \mathcal{A}' , $\text{Adv}_{45}(\mathcal{A}') = 0$.

Finally, adding it all up, and considering that Game 0 is the real experiment and Game 5 is the ideal experiment, we get

$$\begin{aligned} \text{Adv}_{\text{RQFE}}^{\text{SIM}}(\mathcal{A}) &= \text{Adv}_{01}(\mathcal{A}) + \text{Adv}_{12}(\mathcal{A}) + \text{Adv}_{23}(\mathcal{A}) + \text{Adv}_{34}(\mathcal{A}) + \text{Adv}_{45}(\mathcal{A}) \\ &\leq \text{Adv}_{\text{IPFE}}^{\text{FH-SIM}}(\mathcal{B}) + \frac{1}{p}. \end{aligned}$$

□

6 Generic Efficiency Considerations

For any generic function-hiding simulation secure IPFE, our MIQFE scheme supporting ℓ inputs of n coefficient each, needs for ℓ^2 instances of IPFE, ℓ for each input to handle the noise generated with combining with each of the other inputs. A part from that, the master secret key needs for $\{\mathbf{u}_i\}_{i \in [\ell]} \in \mathbb{Z}_p^n$, $\{c_i\}_{i \in [\ell]} \in \mathbb{Z}_p$ and $\{\mathbf{w}_{i,j \in [\ell]}\} \in \mathbb{Z}_p^{2n}$; the ciphertexts need for $ct_{x_i} \in \mathbb{Z}_p^n$; and the functional keys need for $zk_F \in \mathbb{Z}_p$.

For any generic function-hiding simulation secure IPFE, our RQFE scheme supporting inputs of n coefficient each, needs only one instance of IPFE in comparison to the ℓ^2 needed for MIQFE, thus eliminating the quadratic overcost. Apart from that, the master secret key needs for $\mathbf{u} \in \mathbb{Z}_p^n$, $c \in \mathbb{Z}_p$ and $\mathbf{w} \in \mathbb{Z}_p^{2n}$; the ciphertexts need for $ct_x \in \mathbb{Z}_p^n$; and the functional keys need for $t'_F \in \mathbb{Z}_p$ and $zk_F \in \mathbb{Z}_p$.

Our function-hiding IPFE scheme supporting n coefficients, needs in the master secret key for $\mathbf{u} \in \mathbb{Z}_p^{n+1}$ and $\mathbf{v} \in \mathbb{Z}_p^n$; the ciphertext for $ct_1 \in \mathbb{G}_1$ and $ct_2 \in \mathbb{G}_1^{n+1}$; and the functional keys need for $sk_1 \in \mathbb{G}_2$ and $sk_2 \in \mathbb{G}_2^{n+1}$, which also gives us concrete efficiency estimates of a concrete instantiation of our MIQFE scheme. All this is shown in Table 5, where IPFE_s^k denotes the size of the element referred by s of the base IPFE scheme for k coefficients.

7 Privacy and Implementation of RQFE

In this section we discuss the applications of our RQFE in differential privacy. More specifically, we give a short summary of the result and setup from [6] where they prove the relationship between RFE schemes and differential privacy and we specify how our RQFE scheme works in this setting.

7.1 Differential Privacy Considerations

The first objective is to prove whether our RQFE is differentially private for some choice of distribution D_ϵ , and to do so we must construct the privacy mechanism \mathcal{M} we need to analyze. Given that in functional encryption the adversary has access to the ciphertext and the functional decryption key from which they derive the output, the mechanism should include these three objects: ciphertext, functional key and output. More formally, for the family of queries $\mathcal{F}_{Q,B}^n$, we define the mechanism \mathcal{M} to be

$$\mathcal{M}(\mathbf{x}, \mathbf{F}; (r_{\mathbf{x}}, r_{\mathbf{F}})) = \begin{cases} \text{RQFE}.c_{\mathbf{x}} \leftarrow \text{RQFE}.Enc(\text{RQFE}.msk, \mathbf{x}; r_{\mathbf{x}}) \\ \text{RQFE}.sk_{\mathbf{F}} \leftarrow \text{RQFE}.KeyGen(\text{RQFE}.msk, \mathbf{F}; r_{\mathbf{F}}) \\ s \leftarrow \text{RQFE}.Dec(c_{\mathbf{x}}, sk_{\mathbf{F}}) \end{cases}$$

for $(r_{\mathbf{x}}, r_{\mathbf{F}}) \leftarrow \mathcal{R}$, where Enc and $KeyGen$ denote the algorithms taking $r_{\mathbf{x}}$ and $r_{\mathbf{F}}$ as seeds for their randomness respectively and $\text{RQFE}.msk \leftarrow \text{RQFE}.SetUp(1^\kappa, \hat{\mathcal{F}}_{Q,B}^n)$. We divide the randomness space in two to more accurately represent the randomness needed for the RQFE algorithms.

However, directly verifying that differential privacy holds for such a mechanism is not evident, since one would need to compare the distributions of the ciphertext and the functional key for different neighbouring databases. Ideally, this would be done by extending the differential privacy of the decryption s , where it suffices to verify that the mechanism $\mathcal{M}'(\mathbf{x}, \mathbf{F}; r) = \mathbf{x}^\top \mathbf{F} \mathbf{x} + e(r)$ where $e(r) \leftarrow D_\epsilon$ is differentially private. To do so we use the following Theorem.

Theorem 13 (Differential Privacy for Randomized Functional Encryption, [6]) *Let RFE be a 1-SEL-SIM secure randomized functional encryption scheme against Q functional key queries and \mathcal{M}' be an ϵ computationally differentially private mechanism for Q queries. Then the mechanism \mathcal{M} is an ϵ computationally differentially private mechanism for Q queries.*

In this case, computational differential privacy [47] is used, a relaxation of differential privacy where the adversary is considered computationally bounded instead of statistical. This is due to the fact that randomized functional encryption relies in computational assumptions, and as such the confidentiality of \mathbf{x} in the ciphertext is not guaranteed against a statistical adversary. Since computational differential privacy is a relaxation of differential privacy, it suffices

to choose \mathcal{M}' such that it satisfies standard differential privacy (Definition 13) for the whole scheme to be computationally differentially private.

As said in Section 2.4, we will use the Geometric mechanism with the parameters in Lemma 3 in function of the sensitivity of the family of functions $\Delta_{\mathcal{F}}$. For the quadratic family of functions $\mathcal{F}_{Q,B}^n$, we get $\Delta_{\mathcal{F}} = 2n \cdot B^2$. Indeed we have that for any $\mathbf{F} \in \mathcal{F}_{Q,B}^n$ and any neighbouring databases \mathbf{x}, \mathbf{x}' differing only in coordinate k ,

$$\begin{aligned}
\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}')\|_1 &= \left| \sum_{i,j \in n} x_i \cdot x_j \cdot f_{i,j} - \sum_{i,j \in n} x'_i \cdot x'_j \cdot f_{i,j} \right| \\
&= \left| \sum_{i,j \in n} f_{i,j} (x_i \cdot x_j - x'_i \cdot x'_j) \right| \\
&= \left| \sum_{i \in n} f_{i,k} (x_i \cdot x_k - x'_i \cdot x'_k) + \sum_{j \in n} f_{k,j} (x_k \cdot x_j - x'_k \cdot x'_j) \right| \\
&= \left| \sum_{i \in n} f_{i,k} \cdot x_i (x_k - x'_k) + \sum_{j \in n} f_{k,j} \cdot x_j (x_k - x'_k) \right| \\
&= \left| \sum_{i \in n} f_{i,k} \cdot x_i + \sum_{j \in n} f_{k,j} \cdot x_j \right| \\
&\leq \left| \sum_{i \in n} f_{i,k} \cdot x_i \right| + \left| \sum_{j \in n} f_{k,j} \cdot x_j \right|
\end{aligned}$$

which in turn, given that $\mathbf{x} \in \llbracket 0, B \rrbracket^n$ and $\mathbf{F} \in \llbracket 0, B \rrbracket^{n \times n}$ results in

$$\begin{aligned}
\Delta_{\mathcal{F}} &:= \max_{\substack{\|\mathbf{x} - \mathbf{x}'\| \leq 1 \\ \mathbf{F} \in \mathcal{F}_{Q,B}^n}} \|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}')\|_1 \\
&\leq 2n \cdot B^2.
\end{aligned}$$

Finally, adding this together with Lemma 3 and the property of sequential composition (Proposition 2) we get that by defining $D_\epsilon \sim \text{Geo}(\exp(-\epsilon/2Q \cdot n \cdot B^2))$ the mechanism \mathcal{M}' is $(\epsilon, 0)$ -differentially private against Q adaptive queries. Also, by applying Lemma 4 we get that \mathcal{M}' is $(O(1/\epsilon) \cdot 2Q \cdot n \cdot B^2 \cdot \log(\beta/2), \beta)$ -useful.

7.2 Implementation Specifics

For a concrete implementation, we take the RQFE scheme from Encryption Scheme 5 using as a base function-hiding IPFE scheme the one presented in Encryption Scheme 4, due to the fact that it is the only known implementation holding simulation security in the standard model. Note that the decryption algorithm requires to compute the discrete logarithm of the output, which

means this output should be bounded by some bound $K = \text{poly}(\kappa)$. Then, we can use the baby-step giant-step algorithm [48] to compute this discrete logarithm in $\tilde{O}(K^{1/2})$. For a practical implementation we will consider $K \approx 2^{40}$ so that the computation of the discrete logarithm takes around 1.5s.

To better adapt to real-life use-cases we have done two adjustments to the construction proposed in Section 5. First of all we consider positive *and negative* inputs for both the database \mathbf{x} and the function \mathbf{F} . This does not change the output bound K , and only requires us to run the discrete logarithm algorithm twice: once with the generator of \mathbb{G}_T (for the positive values), and once with its inverse (for the negative values). Secondly, we have separated the bound for the inputs into two, a bound X for the database and a bound F for the function. The only change this makes is the bound K is now defined as $n^2 \cdot X^2 \cdot F + \alpha$ with α such that $\Pr[D_\epsilon > \alpha] = \text{neg}(\kappa)$. These modifications do not change the security proof, and allow us to consider databases and functions in a much more fine-grained manner.

Given these changes and our notation, the RQFE output will be bounded by $K = n^2 \cdot X^2 \cdot F + \alpha$ with probability β , where α, β are taken from the utility of the mechanism \mathcal{M}' , and we want $K \approx 2^{40}$. We assume the number of queries supported $Q = 16$, the bound for those queries $F = 2^4$, the privacy budget $\epsilon = 0.1$ and the probability $\beta = 2^{-100}$ so as to be negligible. This gives us a direct correlation between the number of database entries and the maximum bound for these database entries. Note that these values need to be low since the adversary can choose any challenge \mathbf{x}^n such that $\|\mathbf{x}\|_\infty < X$ and query any function $\mathbf{F}^{n \times n}$ such that $\|\mathbf{F}\|_\infty < F$. In real life scenarios it may be more beneficial to make assumptions on the database or the queries (for example sparse matrices) such that the output bound K can be guaranteed to be $\approx 2^{40}$ with bigger parameters. For example, in χ^2 testing the encrypted observations would add to a known value, and the quadratic function to be applied is very sparse (it is only non-zero at the diagonal and one column).

The implementation was coded in C using the libraries GMP [49] for multiple precision integers and MCL [50] for fast pairing-friendly groups based on elliptic curves. More specifically, we use their implementation of the curve BLS12-381 [51], a fast pairing-friendly curve achieving 128-bit security. We give in Table 6 the resulting values when run for different values of n (database entries) and $|X|$ (size of database entries). These experimental results show what was expected, the sizes of the master secret key, ciphertext and functional decryption key all grow linearly with the amount of database entries n . The encryption times also grow linearly as expected, while key generation times and decryption times grow quadratically with n , due to the fact that matrix operations are needed ($\mathbf{u}^\top \mathbf{F}$ and $\mathbf{F}\mathbf{u}$ during key generation and $\mathbf{c}\mathbf{t}_x^\top \mathbf{F}\mathbf{c}\mathbf{t}_x$ during decryption). Finally, the set up times show a less conclusive behaviour since the times are dominated by computing the public parameters (setting up the pairing friendly curve).

Table 6 Efficiency values for RQFE instantiation.

	n	$ X $	msk	c_x	sk_F
Sizes	10	14	1 KB	2 KB	3 KB
	100	11	15 KB	17 KB	28 KB
	1 000	7	156 KB	172 KB	281 KB
	10 000	4	1 MB	1 MB	2 MB

	n	SetUp	Encrypt	KeyGen	Decrypt
Comp. time	10	0.0007 s	0.0008s	0.0019s	1.2317 s
	100	0.0008 s	0.0122 s	0.0174 s	1.5613 s
	1 000	0.0041 s	0.0940 s	0.2712 s	2.2094 s
	10 000	0.0070 s	0.9199 s	14.7027 s	13.8325 s

8 Conclusion

This work contributes to the relations between multi-input functional encryption and efficient randomized functional encryption for differential privacy, and may inspire further work in the subject. Regarding the MIQFE scheme, an interesting direction to follow would be to improve the ciphertext efficiency of the scheme, to see if $O(n\ell)$ is possible while satisfying simulation security. Another noteworthy direction to explore is to look for a transformation directly from single input QFE to MIQFE, so as to try to avoid pairing based schemes, which seem inevitable when using function-hiding IPFE.

Regarding the RQFE scheme and differential privacy, an interesting challenge would be to tackle adaptive databases, where an analyst holding a functional key can obtain a statistic from a changing database while maintaining differential privacy. In this case, some sort of homomorphism property seems necessary as well as the capability to be secure against several plaintexts, even if constraining these (one must be able to be generated from another through some well-defined operations) may help in proving these properties.

References

- [1] D. Boneh, A. Sahai, B. Waters, *Functional encryption: Definitions and challenges*, in *Theory of Cryptography*, ed. by Y. Ishai (Springer, Berlin, Heidelberg, 2011), pp. 253–273. https://doi.org/10.1007/978-3-642-19571-6_16
- [2] A. O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Paper 2010/556 (2010). <https://eprint.iacr.org/2010/556>
- [3] A. Shamir, *Identity-based cryptosystems and signature schemes*, in *Advances in Cryptology*, ed. by G.R. Blakley, D. Chaum (Springer, Berlin, Heidelberg, 1985), pp. 47–53. https://doi.org/10.1007/3-540-39568-7_5
- [4] A. Sahai, B. Waters, *Fuzzy identity-based encryption*, in *Advances in Cryptology – EUROCRYPT 2005*, ed. by R. Cramer (Springer, Berlin,

- Heidelberg, 2005), pp. 457–473. https://doi.org/10.1007/11426639_27
- [5] V. Goyal, A. Jain, V. Koppula, A. Sahai, *Functional encryption for randomized functionalities*, in *Theory of Cryptography*, ed. by Y. Dodis, J.B. Nielsen (Springer, Berlin, Heidelberg, 2015), pp. 325–351. https://doi.org/10.1007/978-3-662-46497-7_13
- [6] F. Alborch Escobar, S. Canard, F. Laguillaumie, D.H. Phan, Computational differential privacy for encrypted databases supporting linear queries. *Proceedings on Privacy Enhancing Technologies* **2024**(4), 583–604 (2024). <https://doi.org/10.56553/popets-2024-0131>
- [7] J. Zalonis, F. Armknecht, L. Scheu-Hachtel, Differentially private functional encryption. *Proceedings on Privacy Enhancing Technologies* **2024**(2), 509–530 (2024). <https://doi.org/10.56553/popets-2024-0061>
- [8] M. Abdalla, F. Bourse, A.D. Caro, D. Pointcheval, *Simple functional encryption schemes for inner products*, in *Public-Key Cryptography – PKC 2015*, ed. by J. Katz (Springer, Berlin, Heidelberg, 2015), pp. 733–751. https://doi.org/10.1007/978-3-662-46447-2_33
- [9] S. Agrawal, B. Libert, D. Stehlé, *Fully secure functional encryption for inner products, from standard assumptions*, in *Advances in Cryptology – CRYPTO 2016*, ed. by M. Robshaw, J. Katz (Springer, Berlin, Heidelberg, 2016), pp. 333–362. https://doi.org/10.1007/978-3-662-53015-3_12
- [10] F. Benhamouda, F. Bourse, H. Lipmaa, *CCA-secure inner-product functional encryption from projective hash functions*, in *Public-Key Cryptography – PKC 2017*, ed. by S. Fehr (Springer, Berlin, Heidelberg, 2017), pp. 36–66. https://doi.org/10.1007/978-3-662-54388-7_2
- [11] G. Castagnos, F. Laguillaumie, I. Tucker, *Practical fully secure unrestricted inner product functional encryption modulo p* , in *Advances in Cryptology – ASIACRYPT 2018*, ed. by T. Peyrin, S. Galbraith (Springer, Cham, 2018), pp. 733–764. https://doi.org/10.1007/978-3-030-03329-3_25
- [12] C.E.Z. Baltico, D. Catalano, D. Fiore, R. Gay, *Practical Functional Encryption for Quadratic Functions with Applications to Predicate Encryption*, in *Advances in Cryptology – CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 67–98. https://doi.org/10.1007/978-3-319-63688-7_3
- [13] P. Ananth, A. Sahai, *Projective Arithmetic Functional Encryption and Indistinguishability Obfuscation from Degree-5 Multilinear Maps*, in *Advances in Cryptology – EUROCRYPT 2017*, ed. by J.S. Coron, J.B. Nielsen (Springer International Publishing, Cham, 2017), pp. 152–181. https://doi.org/10.1007/978-3-319-56620-7_6
- [14] H. Lin, *Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs*, in *Advances in Cryptology – CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 599–629. https://doi.org/10.1007/978-3-319-63688-7_20
- [15] R. Gay, *A New Paradigm for Public-Key Functional Encryption for Degree-2 Polynomials*, in *Public-Key Cryptography – PKC 2020*, ed.

- by A. Kiayias, M. Kohlweiss, P. Wallden, V. Zikas (Springer International Publishing, Cham, 2020), pp. 95–120. https://doi.org/10.1007/978-3-030-45374-9_4
- [16] H. Lin, S. Tessaro, *Indistinguishability Obfuscation from Trilinear Maps and Block-Wise Local PRGs*, in *Advances in Cryptology – CRYPTO 2017*, ed. by J. Katz, H. Shacham (Springer International Publishing, Cham, 2017), pp. 630–660. https://doi.org/10.1007/978-3-319-63688-7_21
- [17] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, H. Wee, *Functional Encryption: New Perspectives and Lower Bounds*, in *Advances in Cryptology – CRYPTO 2013*, ed. by R. Canetti, J.A. Garay (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013), pp. 500–518. https://doi.org/10.1007/978-3-642-40084-1_28
- [18] S. Goldwasser, S.D. Gordon, V. Goyal, A. Jain, J. Katz, F.H. Liu, A. Sahai, E. Shi, H.S. Zhou, *Multi-input Functional Encryption*, in *Advances in Cryptology – EUROCRYPT 2014*, ed. by P.Q. Nguyen, E. Oswald (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014), pp. 578–602. https://doi.org/10.1007/978-3-642-55220-5_32
- [19] M. Abdalla, R. Gay, M. Raykova, H. Wee, *Multi-input inner-product functional encryption from pairings*, in *Advances in Cryptology – EUROCRYPT 2017*, ed. by J.S. Coron, J.B. Nielsen (Springer, Cham, 2017), pp. 601–626. https://doi.org/10.1007/978-3-319-56620-7_21
- [20] M. Abdalla, D. Catalano, D. Fiore, R. Gay, B. Ursu, *Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings*, in *Advances in Cryptology – CRYPTO 2018*, ed. by H. Shacham, A. Boldyreva (Springer International Publishing, Cham, 2018), pp. 597–627. https://doi.org/10.1007/978-3-319-96884-1_20
- [21] S. Agrawal, R. Goyal, J. Tomida, *Multi-input Quadratic Functional Encryption from Pairings*, in *Advances in Cryptology – CRYPTO 2021*, ed. by T. Malkin, C. Peikert (Springer International Publishing, Cham, 2021), pp. 208–238. https://doi.org/10.1007/978-3-030-84259-8_8
- [22] S. Agrawal, R. Goyal, J. Tomida, *Multi-Input Quadratic Functional Encryption: Stronger Security, Broader Functionality*, in *Theory of Cryptography*, ed. by E. Kiltz, V. Vaikuntanathan (Springer Nature Switzerland, Cham, 2022), pp. 711–740. https://doi.org/10.1007/978-3-031-22318-1_25
- [23] E. Shen, E. Shi, B. Waters, *Predicate Privacy in Encryption Systems*, in *Theory of Cryptography*, ed. by O. Reingold (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 457–473. https://doi.org/10.1007/978-3-642-00457-5_27
- [24] A. Bishop, A. Jain, L. Kowalczyk, *Function-hiding inner product encryption*, in *Advances in Cryptology – ASIACRYPT 2015*, ed. by T. Iwata, J.H. Cheon (Springer, Berlin, Heidelberg, 2015), pp. 470–491. https://doi.org/10.1007/978-3-662-48797-6_20
- [25] P. Datta, R. Dutta, S. Mukhopadhyay, *Functional encryption for inner*

- product with full function privacy*, in *Public-Key Cryptography–PKC 2016*, ed. by C.M. Cheng, K.M. Chung, G. Persiano, B.Y. Yang (Springer, Berlin, Heidelberg, 2016), pp. 164–195. https://doi.org/10.1007/978-3-662-49384-7_7
- [26] J. Tomida, M. Abe, T. Okamoto, *Efficient Functional Encryption for Inner-Product Values with Full-Hiding Security*, in *Information Security*, ed. by M. Bishop, A.C.A. Nascimento (Springer International Publishing, Cham, 2016), pp. 408–425. https://doi.org/10.1007/978-3-319-45871-7_24
- [27] S. Kim, J. Kim, J.H. Seo, A new approach to practical function-private inner product encryption. *Theoretical Computer Science* **783**, 22–40 (2019). <https://doi.org/https://doi.org/10.1016/j.tcs.2019.03.016>
- [28] A. Ünal, *Impossibility Results for Lattice-Based Functional Encryption Schemes*, in *Advances in Cryptology – EUROCRYPT 2020*, ed. by A. Canteaut, Y. Ishai (Springer International Publishing, Cham, 2020), pp. 169–199. https://doi.org/10.1007/978-3-030-45721-1_7
- [29] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, D.J. Wu, *Function-Hiding Inner Product Encryption Is Practical*, in *Security and Cryptography for Networks*, ed. by D. Catalano, R. De Prisco (Springer International Publishing, Cham, 2018), pp. 544–562. https://doi.org/10.1007/978-3-319-98113-0_29
- [30] I. Komargodski, G. Segev, E. Yogev, Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. *Journal of Cryptology* **31**(1), 60–100 (2018). <https://doi.org/10.1007/s00145-016-9250-8>
- [31] S. Agrawal, D.J. Wu, *Functional encryption: deterministic to randomized functions from simple assumptions*, in *Advances in Cryptology – EUROCRYPT 2017*, ed. by J.S. Coron, J.B. Nielsen (Springer, Cham, 2017), pp. 30–61. https://doi.org/10.1007/978-3-319-56614-6_2
- [32] C. Dwork, F. McSherry, K. Nissim, A. Smith, *Calibrating noise to sensitivity in private data analysis*, in *Theory of Cryptography*, ed. by S. Halevi, T. Rabin (Springer, Berlin, Heidelberg, 2006), pp. 265–284. https://doi.org/10.1007/11681878_14
- [33] D. Desfontaines, B. Pejó, Sok: differential privacies. *Proceedings on privacy enhancing technologies* **2020**(2), 288–313 (2020). <https://doi.org/10.2478/popets-2020-0028>
- [34] P.R. Bureau, the U.S. Census Bureau’s 2020 Census Data Products, D. Team. Why the census bureau chose differential privacy. *Census Briefs* (2020)
- [35] A. Machanavaajhala, D. Kifer, J. Abowd, J. Gehrke, L. Vilhuber, *Privacy: Theory meets Practice on the Map*, in *2008 IEEE 24th International Conference on Data Engineering (IEEE, Cancun, Mexico, 2008)*, pp. 277–286. <https://doi.org/10.1109/ICDE.2008.4497436>
- [36] Y. Zhang, D. Ramage, Z. Xu, Y. Zhang, S. Zhai, P. Kairouz. Private federated learning in gboard (2023). <https://doi.org/10.48550/arXiv.2306.14793>

- [37] B. Ding, J. Kulkarni, S. Yekhanin, Collecting telemetry data privately. *Advances in Neural Information Processing Systems* **30** (2017). URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/253614bbac999b38b5b60cae531c4969-Abstract.h
- [38] K. Pearson, X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **50**(302), 157–175 (1900). <https://doi.org/10.1080/14786440009463897>
- [39] M. Gaboardi, H. Lim, R. Rogers, S. Vadhan, *Differentially Private Chi-Squared Hypothesis Testing: Goodness of Fit and Independence Testing*, in *Proceedings of The 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 48, ed. by M.F. Balcan, K.Q. Weinberger (PMLR, New York, New York, USA, 2016), pp. 2111–2120. URL <https://proceedings.mlr.press/v48/rogers16.html>
- [40] Z. Kazan, K. Shi, A. Groce, A.P. Bray, *The Test of Tests: A Framework for Differentially Private Hypothesis Testing*, in *Proceedings of the 40th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 202, ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (PMLR, 2023), pp. 16131–16151. URL <https://proceedings.mlr.press/v202/kazan23a.html>
- [41] F. Alborch Escobar, S. Canard, F. Laguillaumie, *Simulation Secure Multi-Input Quadratic Functional Encryption, in Selected Areas in Cryptography – SAC 2024* (2024). URL <https://sacworkshop.org/SAC24/preproceedings/EscobarCanardLaguillaumie.pdf>
- [42] Q. Zhao, Q. Zeng, X. Liu, H. Xu, Simulation-based security of function-hiding inner product encryption. *Science China. Information Sciences* **61**(4), 048102 (2018)
- [43] G. Herold, E. Kiltz, C. Ràfols, J. Villar, An algebraic framework for diffie–hellman assumptions. *Journal of Cryptology* **30**(1), 242–288 (2017). <https://doi.org/10.1007/s00145-015-9220-6>
- [44] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, M. Naor, *Our Data, Ourselves: Privacy Via Distributed Noise Generation*, in *Advances in Cryptology - EUROCRYPT 2006*, ed. by S. Vaudenay (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 486–503. https://doi.org/10.1007/11761679_29
- [45] A. Ghosh, T. Roughgarden, M. Sundararajan, *Universally Utility-Maximizing Privacy Mechanisms*, in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, USA, 2009), p. 351–360. <https://doi.org/10.1145/1536414.1536464>
- [46] M. Abdalla, D. Catalano, D. Fiore, R. Gay, B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. *Cryptology ePrint Archive*, Paper

- 2017/972 (2017). https://doi.org/10.1007/978-3-319-96884-1_20. <https://eprint.iacr.org/2017/972>
- [47] I. Mironov, O. Pandey, O. Reingold, S. Vadhan, *Computational differential privacy*, in *Advances in Cryptology - CRYPTO 2009*, ed. by S. Halevi (Springer, Berlin, Heidelberg, 2009), pp. 126–142. https://doi.org/10.1007/978-3-642-03356-8_8
- [48] J.M. Pollard, Kangaroos, monopoly and discrete logarithms. *J. Cryptol.* **13**(4), 437–447 (2000). <https://doi.org/10.1007/s001450010010>
- [49] T. Granlund, the GMP development team, *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6th edn. (2023). <http://gmplib.org/>
- [50] S. Mitsunari, *MCL: A portable and fast pairing-based cryptography library*, 1st edn. (2024). <https://github.com/herumi/mcl/tree/master>
- [51] S. Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. <https://electriccoin.co/blog/new-snark-curve/> (March 2017)