Simulation Secure Multi-Input Quadratic Functional Encryption

Ferran Alborch Escobar[1,2,3], Sébastien Canard[2], and Fabien Laguillaumie[3]

[1] Applied Crypto Group, Orange Innovation, 14000 Caen, France
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France
[3] LIRMM, Université de Montpellier, CNRS, 34095 Montpellier, France

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. In this work we propose the first secret-key multi-input quadratic functional encryption scheme satisfying simulation security. On contrary, current constructions supporting quadratic functionalities, proposed by Agrawal *et al.* in CRYPTO '21 and TCC '22, only reach indistinguishibility-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.

Keywords: Functional Encryption · Multi-input · Quadratic Fuctions

1 Introduction

Functional encryption is a generalization of public key encryption first formalized in [14,24], allowing for more control over the access on encrypted data. In general terms, this means that there is an authority capable to generate 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.

On the construction of functional encryption (FE) for some specific functionalities, [1] 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 [8,12,15]. Another set of works, first introduced in [11], refer to quadratic functions (QFE), for which papers in [10,22,17] propose instantiations in the standard model. For higher degrees it is known that succinct degree-3 functional encryption implies indistinguishability obfuscation [23] 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 [14,24]. 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 [14]. 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 trough 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 [14,24,5]. 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 in selective simulation security against one challenge ciphertext, and multiple functional keys, which makes sense in many real-life applications, where several queries are asked to one single database.

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 (calleds 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.* [4] with a secret-key multi-input scheme. It was followed by a transformation from single-input IPFE to the multi-input case [3] 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.* [6,7] 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 public key multi-input functional encryption is easily constructed from single-input functional encryption [4,6] (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 [4, Section 1.1] and [6, Section 1.2, Appendix A.2].

Function-hiding functional encryption. As in [4,6], 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 [25]. 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 [13,16,26,21] (especially because a recent result shows that Learning With Errors based constructions are impossible [27]). 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 [20] which achieves such security in the Generic Group Model.

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 ciphetext 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 base ourselves on the single-input quadratic functional encryption scheme of [17, Section 3], using techniques from the multi-input inner-product functional encryption scheme from [3, 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 [1, Section 3] and inspired by the partially function-hiding inner-product scheme in [17, Section 4]. This is the first simulation secure function-hiding inner-product scheme in the standard model.

1.2 State of the Art

In quadratic functional encryption, the function is generally defined by a matrix $\boldsymbol{F} \in \mathbb{Z}_p^{n \times n}$ with $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{F} \boldsymbol{x}$. 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 $\boldsymbol{y} \in \mathbb{Z}_p^n$ with $\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{y}$. For such a generic construction, we first observe that $\boldsymbol{x}^\top \boldsymbol{F} \boldsymbol{x} = (\boldsymbol{x} \otimes \boldsymbol{x})^\top \mathsf{vect}(\boldsymbol{F})$ where $\otimes$ denotes the Kronecker product and $\mathsf{vect}(\boldsymbol{F})$ is the vectorization of $\boldsymbol{F}$. From that, such naive QFE can be constructed as follows.

Encryption Scheme 1 (Naive QFE)

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

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 $\mathsf{Enc}(\boldsymbol{x}_i \otimes \boldsymbol{x}_j)$ and $\boldsymbol{x}_i, \boldsymbol{x}_j$ to be encrypted independently.

There are only two proposals for multi-input quadratic functional encryption [6,7]. In both, they adapt the single-input quadratic scheme from Lin [22] 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 [6] they make use of two extra functionalities, namely predicated inner-product functional encryption and mixed group inner-product functional encryption, while in [7] 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 [17] 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 [3] 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 [17] 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

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	Simulation security	Size of the ciphertext
Naive (1)	IPFE	QFE	✓	$O(n^2\ell^2)$
[17]	PFH-IPFE	QFE	✓	$O(n\ell)$
[3]	IPFE	MIPFE	✓	$O(n\ell)$
[7]	FH-IPFE	MIQFE	✗	$O(n\ell)$
Our work	FH-IPFE	MIQFE	✓	$O(n\ell^2)$

[3] 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 [7] 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 [20], where they achieve adaptive simulation security against many challenge ciphertexts by constraining themselves to the generic group model (GGM). Since this is an oracle-based model, the impossibility results for adaptive simulation security [14,24,5] no longer hold. There is also [28] 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.

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 [6,7] are simulation sound. However, the way both these schemes are constructed from the single-input quadratic functional encryption scheme by Lin [22] 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 $\boldsymbol{F}_{i,j}$ 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 [14], 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 [17] 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 [17] for details). Such a construction is sketched below, in the secret key setting to simplify the reading.

Encryption Scheme 2 (Simplified Figure 4, [17])

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

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

  and run $(\mathsf{IPFE.pk}, \mathsf{IPFE.msk}) \leftarrow \mathsf{IPFE.SetUp}(1^\kappa, [\boldsymbol{M}]_1)$. *Output* $\mathsf{QFE.msk} = (\boldsymbol{a}, \boldsymbol{B}, \boldsymbol{U}, \boldsymbol{V}, \mathsf{IPFE.pk}, \mathsf{IPFE.msk})$
- **QFE.Enc($\boldsymbol{x}$)** : *Sample* $r \xleftarrow{\$} \mathbb{Z}_p$ *and* $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^2$ *and compute*

$$[\boldsymbol{c}_{\boldsymbol{x}}^1]_1 := [\boldsymbol{x} + \boldsymbol{Uar}]_1,\ [\boldsymbol{c}_{\boldsymbol{x}}^2]_2 := [\boldsymbol{x} + \boldsymbol{VBs}]_2,$$
$$\mathsf{IPFE}.c \leftarrow \mathsf{IPFE.Enc}\left(\mathsf{IPFE.pk}, \begin{pmatrix} r \otimes \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{s} \end{pmatrix} \\ \boldsymbol{x} \otimes \boldsymbol{s} \end{pmatrix}\right).$$

  Output $c_{\boldsymbol{x}} = ([\boldsymbol{c}_{\boldsymbol{x}}^1]_1, [\boldsymbol{c}_{\boldsymbol{x}}^2]_2, \mathsf{IPFE}.c)$.
- **QFE.KeyGen($\boldsymbol{F}$)** : *Compute*

$$\mathsf{IPFE}.sk \leftarrow \mathsf{IPFE.KeyGen}\left(\mathsf{IPFE.msk}, \begin{pmatrix} \mathsf{vect}(\boldsymbol{U}^\top \boldsymbol{F}) \\ \mathsf{vect}(\boldsymbol{FV}) \end{pmatrix}\right).$$

  Output $sk_{\boldsymbol{F}} = (\boldsymbol{F}, \mathsf{IPFE}.sk)$.
- **QFE.Dec($c_{\boldsymbol{x}}, sk_{\boldsymbol{F}}$)** : *Compute*

$$[d]_T \leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE}.c, \mathsf{IPFE}.sk),\ [v]_T := e([\boldsymbol{c}_{\boldsymbol{x}}^1]_1, \boldsymbol{F}[\boldsymbol{c}_{\boldsymbol{x}}^2]_2) - [d]_T$$

  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 $\boldsymbol{c}_{\boldsymbol{x}}^1$, $\boldsymbol{c}_{\boldsymbol{x}}^2$ which are combined as ${\boldsymbol{c}_{\boldsymbol{x}}^1}^\top \boldsymbol{F} \boldsymbol{c}_{\boldsymbol{x}}^2$ to give the desired value $\boldsymbol{x}^\top \boldsymbol{F} \boldsymbol{x}$ 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 $\boldsymbol{U}$ and $\boldsymbol{V}$ (in concrete, $[\boldsymbol{c}_{\boldsymbol{x}}^i]_i$ 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 $e([\boldsymbol{c}_{\boldsymbol{x}_i}^1]_1, \boldsymbol{F}_{i,j}[\boldsymbol{c}_{\boldsymbol{x}_j}^2]_2)$:

$$\begin{aligned} e([\boldsymbol{c}_{\boldsymbol{x}_i}^1]_1, \boldsymbol{F}_{i,j}[\boldsymbol{c}_{\boldsymbol{x}_j}^2]_2) = {} & [\boldsymbol{x}_i^\top \boldsymbol{F}_{i,j} \boldsymbol{x}_j]_T + [\boldsymbol{x}_i^\top \boldsymbol{F_{i,j}} \boldsymbol{VBs}_i]_T \\ & + [\boldsymbol{Uar}_i^\top \boldsymbol{F}_i, j \boldsymbol{x}_j]_T + [\boldsymbol{Uar}_i^\top \boldsymbol{F}_i, j \boldsymbol{VBs}_i]_T. \end{aligned}$$

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

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

However, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ 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 $\mathsf{IPFE}.c_{i,j}$ and $\mathsf{IPFE}.c_{j,i}$. This allows us to "interweave" the ciphertexts, where in input i we compute half of what is needed for $e([\boldsymbol{ct}_{\boldsymbol{x}_i}]_1, \boldsymbol{F}_{i,j}[\boldsymbol{ct}_{\boldsymbol{x}_j}]_2)$ and half of what is needed for $e([\boldsymbol{ct}_{\boldsymbol{x}_j}]_1, \boldsymbol{F}_{j,i}[\boldsymbol{ct}_{\boldsymbol{x}_i}]_2)$, while the other two halves are computed on input j. Therefore, combining the decryptions of $\mathsf{IPFE}.c_{i,j}$ and $\mathsf{IPFE}.c_{j,i}$ we obtain all the extra terms resulting from $e([\boldsymbol{ct}_{\boldsymbol{x}_i}]_1, \boldsymbol{F}_{i,j}[\boldsymbol{ct}_{\boldsymbol{x}_j}]_2)$ and $e([\boldsymbol{ct}_{\boldsymbol{x}_j}]_1, \boldsymbol{F}_{j,i}[\boldsymbol{ct}_{\boldsymbol{x}_i}]_2)$. More specifically, by computing the following where the changes are squared

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

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

$$\begin{gathered} \mathsf{IPFE.Dec}(\mathsf{IPFE}.c_{i,j}, \mathsf{IPFE}.sk_{i,j}) + \mathsf{IPFE.Dec}(\mathsf{IPFE}.c_{j,i}, \mathsf{IPFE}.sk_{j,i}) \\ = \\ \mathsf{IPFE.Dec}(\mathsf{IPFE}.\widetilde{c}_{i,j}, \mathsf{IPFE}.\widetilde{sk}_{i,j}) + \mathsf{IPFE.Dec}(\mathsf{IPFE}.\widetilde{c}_{j,i}, \mathsf{IPFE}.\widetilde{sk}_{j,i}) \end{gathered}$$

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 [17] 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 [3] 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 $\boldsymbol{w} + \boldsymbol{x}$ and then compute an extra $zk_{\boldsymbol{y}} = \boldsymbol{w}^\top \boldsymbol{y}$ during key generation, which one is finally substracted 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 $\boldsymbol{w}$ will still have enough entropy to make $zk_{\boldsymbol{F}}$ indistinguishable from random.

Function-hiding Inner-product Functional Encryption Scheme. Similarly to the construction of partially function-hiding functional encryption from [17, 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: $\mathsf{Enc}^{\mathsf{FH}}(\boldsymbol{x}) = \mathsf{Enc}^{\mathsf{out}}(\mathsf{KeyGen}^{\mathsf{in}}(\boldsymbol{x}))$ and $\mathsf{KeyGen}^{\mathsf{FH}}(\boldsymbol{y}) = \mathsf{KeyGen}^{\mathsf{out}}(\mathsf{Enc}^{\mathsf{in}}(\boldsymbol{y}))$. 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 [17, Section 4] is based on the non function-hiding scheme from [8, 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 [1, Section 3] is enough. As such we use a different approach than [17] 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 [17].

This layering approach is very similar to the function-hiding inner product functional encryption scheme given in [22, Section 6.3]. Indeed, the crucial difference is the order in which the layers are set. In their case they have $\mathsf{Enc}^{\mathsf{FH}}(\boldsymbol{x}) = \mathsf{KeyGen}^{\mathsf{out}}(\mathsf{Enc}^{\mathsf{in}}(\boldsymbol{x}))$ and $\mathsf{KeyGen}^{\mathsf{FH}}(\boldsymbol{y}) = \mathsf{Enc}^{\mathsf{out}}(\mathsf{KeyGen}^{\mathsf{in}}(\boldsymbol{y}))$. However, this set up does not work for proving simulation security. As evidence, let us denote $\mathsf{key}^{\mathsf{out}}, \mathsf{key}^{\mathsf{in}}$ 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) $\mathsf{key}^{\mathsf{out}}$. 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 [1] 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 $\mathsf{key}^{\mathsf{in}}$ 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.

2 Preliminaries

One-dimensional elements will be noted as lower-case letters $(x, y, \ldots)$, while vectors will use bold lower-case letters $(\boldsymbol{x}, \boldsymbol{y}, \ldots)$ and matrices will use bold upper-case letters $(\boldsymbol{M}, \boldsymbol{U}, \boldsymbol{V}, \ldots)$. Let D be a probability distribution, $x \leftarrow D$ means the element x is sampled from the distribution D, while for any set $\mathcal{Y}$, $y \xleftarrow{\$} \mathcal{Y}$ 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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ for any $n, m \geq 1$ we define $[\boldsymbol{A}]_s$ as the representation of $\boldsymbol{A}$ in the group $\mathbb{G}_s$. In other words, $[\boldsymbol{A}]_s = (a_{ij} P_s) \in \mathbb{G}_s^{n \times m}$.For any two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{Z}_p^{n \times m}$, $e([\boldsymbol{A}]_1, [\boldsymbol{B}]_2) := [\boldsymbol{AB}]_T$ and for $s \in \{1, 2, T\}$ we have $[\boldsymbol{A}]_s + [\boldsymbol{B}]_s := [\boldsymbol{A} + \boldsymbol{B}]_s$.

Finally, we define the PPT algorithm PGGen 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 Functional Encryption

Functional encryption is a generalization of encryption first formalized by Boneh *et al.* [14] and O'Neill [24], 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 1 (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 $f : \mathcal{X} \to \mathcal{S}$. We define a* secret-key functional encryption scheme *the following tuple of* PPT *algorithms:*

- $\mathsf{FE.SetUp}(1^\kappa, \mathcal{F})$: *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.*
- $\mathsf{FE.Enc}(\mathsf{FE.msk}, x)$: *given the master secret key* FE.msk *and a plaintext $x \in \mathcal{X}$ as inputs, it outputs a ciphertext c_x.*
- $\mathsf{FE.KeyGen}(\mathsf{FE.msk}, f)$: *given the master secret key* FE.msk *and a function $f \in \mathcal{F}$ as inputs, it outputs a functional key sk_f.*

– $\mathsf{FE.Dec}(c_x, sk_f)$: *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 2 (Correctness of Functional Encryption). *Let* $\kappa \in \mathbb{N}_{>0}$ *be a security parameter and* $\mathsf{FE} = (\mathsf{FE.SetUp}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ *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*

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

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

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. The standard definitions for this are case as follows.

Definition 3 (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* $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_\ell \to \mathcal{S}$*. We define a* secret-key multi-input quadratic functional encryption scheme *as the following tuple of* PPT *algorithms:*

– $\mathsf{MIFE.SetUp}(1^\kappa, \mathcal{F})$*: given the security parameter* 1^κ *and a family of* ℓ*-ary functions* $\mathcal{F}$*, it outputs some public parameters* $\mathsf{MIFE.param}$ *a master secret key* $\mathsf{MIFE.msk}$*. We will assume the public parameters as inputs in all other algorithms.*
– $\mathsf{MIFE.Enc}(\mathsf{MIFE.msk}, i, x_i)$*: given the master secret key* $\mathsf{MIFE.msk}$*, an input number* $i \in [\ell]$ *and* $x_i \in \mathcal{X}_i$*, it outputs a ciphertext* c_{x_i}*.*
– $\mathsf{MIFE.KeyGen}(\mathsf{MIFE.msk}, f)$*: given the master secret key* $\mathsf{MIFE.msk}$ *and a function* $f \in \mathcal{F}$ *as inputs, it outputs a functional decryption key* sk_f*.*
– $\mathsf{MIFE.Dec}(c_{x_1}, \ldots, c_{x_\ell}, sk_f)$*: a deterministic algorithm that given ciphertexts* $c_{x_1}, \ldots, c_{x_\ell}$ *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 4 (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* $\mathsf{MIFE} = (\mathsf{MIFE.SetUp}, \mathsf{MIFE.Enc}, \mathsf{MIFE.KeyGen}, \mathsf{MIFE.Dec})$ *be a secret-key multi-input functional encryption scheme. We say it is* correct *if for any* $x_1 \in \mathcal{X}_1, \ldots, x_\ell \in \mathcal{X}_\ell$ *and* $f \in \mathcal{F}$ *we have*

$$\Pr\left[\mathsf{MIFE.Dec}(c_{x_1}, \ldots, c_{x_\ell}, sk_f) \neq f(x_1, \ldots, x_\ell)\right] = \mathsf{negl}(\kappa)$$

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

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

$\mathsf{Exp}^{real}_{\mathcal{A}}(1^\kappa)$:
1: $\mathsf{MIFE.msk} \leftarrow \mathsf{MIQFE.SetUp}(1^\kappa, \mathcal{F})$
2: $(\{x_i\}_{i\in[\ell]}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $\boldsymbol{x}_i \in \mathcal{X}_i$
3: For all $i \in [\ell]$, $c_{x_i} \leftarrow \mathsf{MIFE.Enc}(\mathsf{MIFE.msk}, i, \boldsymbol{x}_i)$
4: $\gamma \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{MIFE.KeyGen}}(\mathsf{msk},\cdot)}(\{c_{x_i}\}_{i\in\ell}, \mathsf{st}_1)$

$\mathsf{Exp}^{ideal}_{\mathcal{A},\mathcal{S}}(1^\kappa)$:
1: $\widetilde{\mathsf{MIFE.msk}} \leftarrow \mathsf{MIFE.SetUpSim}(1^\kappa, \mathcal{F})$
2: $(\{x_i\}_{i\in[\ell]}, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x_i \in \mathcal{X}_i$
3: For all $i \in [\ell]$, $\widetilde{c}_{x_i} \leftarrow \mathsf{MIFE.EncSim}(\mathsf{MIFE.}\widetilde{\mathsf{msk}}, i)$
4: $\gamma \leftarrow \mathcal{A}_2^{\widetilde{\mathcal{O}}_{\mathsf{MIFE.KeyGen}}(\mathsf{MIFE.}\widetilde{\mathsf{msk}},\{x_i\}_{i\in[\ell]},\cdot)}(\{\widetilde{c}_{x_i}\}_{i\in[\ell]}, \mathsf{st}_1)$

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 5 (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 $\mathsf{MIFE} = (\mathsf{MIFE.SetUp}, \mathsf{MIFE.Enc}, \mathsf{MIFE.KeyGen}, \mathsf{MIFE.Dec})$ be a secret-key multi-input functional encryption scheme. For any PPT simulator $\mathcal{S} := (\mathsf{MIFE.SetUpSim}, \mathsf{MIFE.EncSim}, \mathsf{MIFE.KeyGenSim})$ and any PPT adversary $\mathcal{A}$ we define the experiments in Table 2 where the oracles are described as follows.*

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

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

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

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 6 (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 $\mathsf{FE} = (\mathsf{FE.SetUp}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ be a secret-key functional encryption scheme. For any* PPT *simulator $\mathcal{S} := (\mathsf{FE.SetUpSim}, \mathsf{FE.EncSim}, \mathsf{FE.KeyGenSim})$ and any* PPT *adversary $\mathcal{A}$ we define the experiments in Table 3 where the oracles are described as follows.*

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

$\mathsf{Exp}^{real}_{\mathcal{A}}(1^\kappa)$:
1: $\mathsf{FE.msk} \leftarrow \mathsf{FE.SetUp}(1^\kappa, \mathcal{F})$
2: $(x, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$
3: $c_x \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, x)$
4: $\gamma \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{FE.KeyGen}}(\mathsf{FE.msk},\cdot)}(c_x, \mathsf{st}_1)$

$\mathsf{Exp}^{ideal}_{\mathcal{A},\mathcal{S}}(1^\kappa)$:
1: $\widetilde{\mathsf{FE.msk}} \leftarrow \mathsf{FE.SetUpSim}(1^\kappa, \mathcal{F})$
2: $(x, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ where $x \in \mathcal{X}$
3: $\widetilde{c}_x \leftarrow \mathsf{FE.EncSim}(\widetilde{\mathsf{FE.msk}})$
4: $\gamma \leftarrow \mathcal{A}_2^{\widetilde{\mathcal{O}}_{\mathsf{FE.KeyGen}}(\widetilde{\mathsf{FE.msk}},x,\cdot)}(\widetilde{c}_x, \mathsf{st}_1)$

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

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

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

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.

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 $\mathcal{F}^{n,\ell}_{\mathsf{Q},B} : (\llbracket 0, B \rrbracket^n)^\ell \to \llbracket 0, (n\ell)^2 \cdot B^3 \rrbracket$ be the family of ℓ-ary functions such that a function $\boldsymbol{F} \in \mathcal{F}^{n,\ell}_{\mathsf{Q},B}$ is defined by a matrix in $\llbracket 0, B \rrbracket^{n\ell \times n\ell}$ which we note as

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

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

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

3.1 Description of the Scheme

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

Encryption Scheme 3 (MIQFE Scheme)

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

$$\begin{aligned}&\mathsf{MIQFE.param} = \mathcal{PG} \text{ and}\\ &\mathsf{MIQFE.msk} = (\{\boldsymbol{u}_i, c_i\}_{i\in[\ell]}, \{\boldsymbol{w}_{i,j}, \mathsf{IPFE.msk}_{i,j}\}_{i,j\in[\ell]}).\end{aligned}$$

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

$$\begin{aligned}\boldsymbol{ct}_{\boldsymbol{x}_i} &:= \boldsymbol{x}_i + c_i \boldsymbol{u}_i \in \mathbb{Z}_p^n,\\ \mathsf{IPFE}.c_{i,j} &\leftarrow \mathsf{IPFE.Enc}\left(\mathsf{IPFE.msk}_{i,j}, \boldsymbol{w}_{i,j} + c_j \begin{pmatrix}\boldsymbol{ct}_{\boldsymbol{x}_i}\\ \boldsymbol{x}_i\end{pmatrix}\right) \text{ for } j \in [\ell].\end{aligned}$$

  Output $\mathsf{MIQFE}.c_i = (\boldsymbol{ct}_{\boldsymbol{x}_i}, \{\mathsf{IPFE}.c_{i,j}\}_{j\in[\ell]})$.
- $\mathbf{MIQFE.KeyGen}(\mathsf{MIQFE.msk}, \boldsymbol{F})$: *Compute*

$$\begin{aligned}\mathsf{IPFE}.sk_{i,j} &\leftarrow \mathsf{IPFE.KeyGen}\left(\mathsf{IPFE.msk}_{i,j}, \begin{pmatrix}\boldsymbol{u}_j^\top \boldsymbol{F}_{j,i}\\ \boldsymbol{F}_{i,j}\boldsymbol{u}_j\end{pmatrix}\right) \text{ for } i, j \in [\ell],\\ zk_{\boldsymbol{F}} &\leftarrow \sum_{i,j\in[\ell]} \boldsymbol{w}_{i,j}^\top \begin{pmatrix}\boldsymbol{u}_j^\top \boldsymbol{F}_{j,i}\\ \boldsymbol{F}_{i,j}\boldsymbol{u}_j\end{pmatrix}.\end{aligned}$$

  Output $\mathsf{MIQFE}.sk_{\boldsymbol{F}} = (\boldsymbol{F}, \{\mathsf{IPFE}.sk_{i,j}\}_{i,j\in[\ell]}, zk_{\boldsymbol{F}})$.
- $\mathbf{MIQFE.Dec}(\mathsf{MIQFE}.c_1, \ldots, \mathsf{MIQFE}.c_\ell, \mathsf{MIQFE}.sk_{\boldsymbol{F}})$: *Compute*

$$\begin{aligned}[d_{i,j}]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE}.c_{i,j}, \mathsf{IPFE}.sk_{i,j})\\ [v]_T &:= \left(\sum_{i,j\in[\ell]} [\boldsymbol{ct}_{\boldsymbol{x}_i}^\top \boldsymbol{F}_{i,j} \boldsymbol{ct}_{\boldsymbol{x}_j}]_T - [d_{i,j}]_T\right) + [zk_{\boldsymbol{F}}]_T\end{aligned}$$

  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 $\boldsymbol{y}$ 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 $\boldsymbol{w}_{i,j} + c_j \begin{pmatrix}\boldsymbol{ct}_{\boldsymbol{x}_i}\\ \boldsymbol{x}_i\end{pmatrix}$ 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 1. *The* MIQFE *scheme defined in Section 3.1 is a correct multi-input functional encryption scheme for* $\mathcal{F}_{\mathsf{Q},B}^{n,\ell}$ *as long as* IPFE *is a correct scheme for* $\widetilde{\mathcal{F}}_{\mathsf{IP}}^{2n}$.

For the proof we refer to Appendix B.

Theorem 1. *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*

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

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 $\mathsf{Exp}_{\mathcal{A}}^{real}(1^\kappa) =$ Game $0 \approx_s$ Game $1 \approx_s$ Game $2 \approx_s$ Game $3 \approx_c$ Game $4 = \mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{ideal}(1^\kappa)$. Let also $\mathsf{IPFE.Sim} = (\mathsf{IPFE.SetUpSim}, \mathsf{IPFE.EncSim}, \mathsf{IPFE.KeyGenSim})$ 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 $i = 0,1,2,3$) $\mathsf{Adv}_{i(i+1)}(\mathcal{A}') := \left|\Pr\left[\tilde{b} = 1 | b = 0\right] - \Pr\left[\tilde{b} = 1 | b = 1\right]\right|$.

Game 0. This is the real experiment for 1-SEL-SIM security for MIQFE.

Game 1. In this Game we are changing the ciphertext $\boldsymbol{ct}_{\boldsymbol{x}_i}$ to $c_i \boldsymbol{u}_i$ which is random, and then change $\boldsymbol{u}_i$ to $\widetilde{\boldsymbol{u}}_i := \boldsymbol{u}_i - c_i^{-1}\boldsymbol{x}_i$ 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 $\boldsymbol{u}_i$ and $\widetilde{\boldsymbol{u}}_i$.

First we show that coherence is kept. Indeed,

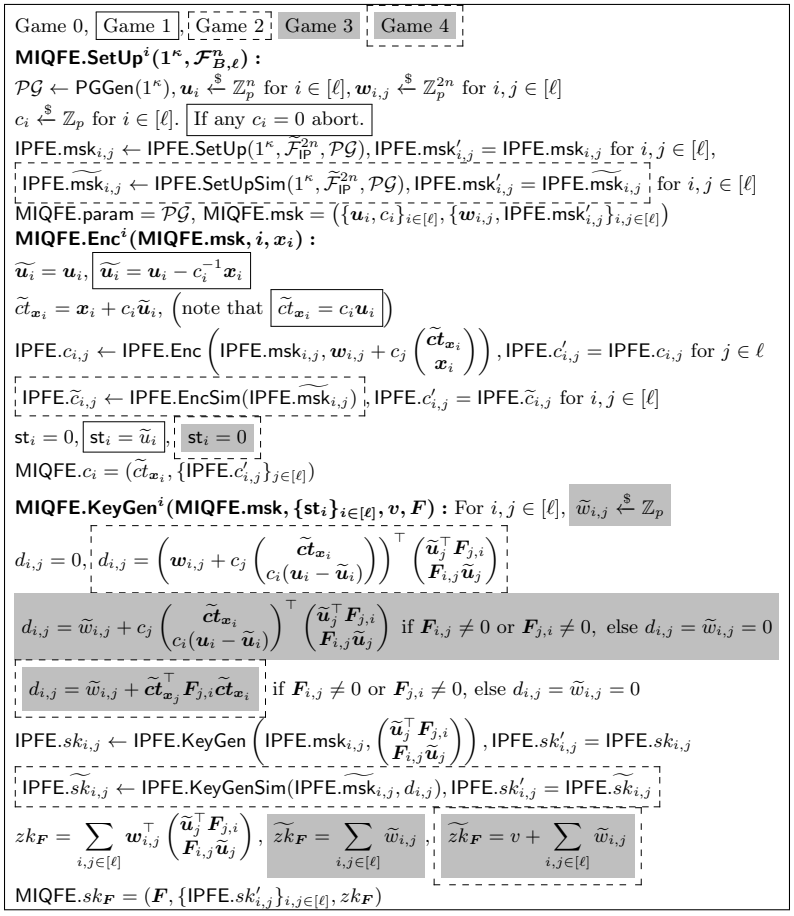

Fig. 1. Games for the security proof of the MIQFE scheme from Encryption Scheme 3.

$$
\begin{aligned}
[\widetilde{v}]_T &= \left(\sum_{i,j\in[\ell]} [\widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i}\boldsymbol{F}_{i,j}\widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_j}]_T - [d_{i,j}]_T\right) + [zk_{\boldsymbol{F}}]_T \\
&= \left(\sum_{i,j\in[\ell]} \left[c_i\boldsymbol{u}_i^\top \boldsymbol{F}_{i,j}c_j\boldsymbol{u}_j\right]_T - \left[\left(\boldsymbol{w}_{i,j} + c_j\left(\widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i}\boldsymbol{x}_i\right)\right)^\top \begin{pmatrix}\widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i}\\ \boldsymbol{F}_{i,j}\widetilde{\boldsymbol{u}}_j\end{pmatrix}\right]_T\right) \\
&\quad + \left[\sum_{i,j\in[\ell]} \boldsymbol{w}_{i,j}^\top \begin{pmatrix}\widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i}\\ \boldsymbol{F}_{i,j}\widetilde{\boldsymbol{u}}_j\end{pmatrix}\right]_T \\
&= \sum_{i,j\in[\ell]} \left[c_i\boldsymbol{u}_i^\top \boldsymbol{F}_{i,j}c_j\boldsymbol{u}_j\right]_T - [c_jc_i\boldsymbol{u}_i^\top(\boldsymbol{u}_j - c_j^{-1}\boldsymbol{x}_j)^\top \boldsymbol{F}_{j,i}]_T \\
&\quad - [c_j\boldsymbol{x}_i^\top \boldsymbol{F}_{i,j}(\boldsymbol{u}_j - c_j^{-1}\boldsymbol{x}_j)]_T
\end{aligned}
$$

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

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 $\boldsymbol{u}_i$ and $\widetilde{\boldsymbol{u}}_i$ are indistinguishable, so will be Game 0 and Game 1. Now, $\widetilde{\boldsymbol{u}}_i$ will exist as long as c_i has an inverse, which it will since $c_i \neq 0$, and given that $\boldsymbol{u}_i$ is sampled uniformly at random in $\mathbb{Z}_p^n$ and used only once (we are proving one selective security), then $\boldsymbol{u}_i$ and $\widetilde{\boldsymbol{u}}_i$ are computationally indistinguishable. Therefore, for any PPT adversary $\mathcal{A}'$, $\mathsf{Adv}_{01}(\mathcal{A}') \leq \ell/p$.

Game 2. In this Game we are substituting the IPFE algorithms ($\mathsf{IPFE.SetUp}$, $\mathsf{IPFE.Enc}$, $\mathsf{IPFE.KeyGen}$) used to compute $d_{i,j}$ which we do not modify, by their corresponding simulators ($\mathsf{IPFE.SetUpSim}$, $\mathsf{IPFE.EncSim}$, $\mathsf{IPFE.KeyGenSim}$), so distinguishing between games would imply an adversary breaking 1-$\mathsf{SEL\text{-}SIM}$ security for IPFE.

More formally, we prove in Lemma 2 that for any PPT adversary $\mathcal{A}'$, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{12}(\mathcal{A}') \leq \ell^2 \cdot \mathsf{Adv}^{\mathsf{FH\text{-}SIM}}_{\mathsf{IPFE}}(\mathcal{B})$. 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 $\boldsymbol{w}_{i,j}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}$ for a random value $\widetilde{w}_{i,j}$, as long as $\boldsymbol{F}_{j,i} \neq \boldsymbol{0}$ or $\boldsymbol{F}_{i,j} \neq \boldsymbol{0}$. Otherwise we keep the value at 0. First we show that coherence is held. Indeed,

$$
\begin{aligned}
[zk_{\boldsymbol{F}}]_T - \sum_{i,j\in[\ell]} [d_{i,j}]_T &= \\
&= \left[\sum_{i,j\in[\ell]} \boldsymbol{w}_{i,j}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix} - \sum_{i,j\in[\ell]} \left(\boldsymbol{w}_{i,j} + c_j \begin{pmatrix} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}\right)^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}\right]_T \\
&= \left[\sum_{i,j\in[\ell]} c_j \begin{pmatrix} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}\right]_T , \\
[\widetilde{zk}_{\boldsymbol{F}}]_T - \sum_{i,j\in[\ell]} [\widetilde{d}_{i,j}]_T &= \left[\sum_{i,j\in[\ell]} \widetilde{w}_{i,j} - \sum_{i,j\in[\ell]} \widetilde{w}_{i,j} + c_j \begin{pmatrix} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}\right]_T \\
&= \left[\sum_{i,j\in[\ell]} c_j \begin{pmatrix} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}\right]_T .
\end{aligned}
$$

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

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 $\widetilde{w}_{i,j} + c_j \begin{pmatrix} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}^\top \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^\top \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}$ to $\widetilde{w}_{i,j} + \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_j}^\top \boldsymbol{F}_{j,i} \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i}$ and modify $zk_{\boldsymbol{F}}$ from $\sum_{i,j\in[\ell]} \widetilde{w}_{i,j}$ to $v + \sum_{i,j\in[\ell]} \widetilde{w}_{i,j}$ 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

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

since $c_i(\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) = \boldsymbol{x}_i$ by definition of $\widetilde{\boldsymbol{u}}_i$. and in Game 4 we get

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

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

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

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

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 tow instances of the scheme [1, Section 3].

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

4.1 Description of the Scheme

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

Encryption Scheme 4 (Function-hiding IPFE Scheme)

- **IPFE.SetUp($1^\kappa, \mathcal{F}^n_{\mathsf{IP}}$)** : *Sample* $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2, e) \leftarrow \mathsf{PGGen}(1^\kappa)$, $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{n+1}$ *and* $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^n$. *Output* $\mathsf{IPFE.param} = \mathcal{PG}$ *and* $\mathsf{IPFE.msk} = (\boldsymbol{u}, \boldsymbol{v})$.
- **IPFE.Enc(IPFE.msk, $\boldsymbol{x}$)** : *Sample* $c \xleftarrow{\$} \mathbb{Z}_p$. *Compute*

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

  Output $\mathsf{IPFE}.c_{\boldsymbol{x}} := (ct_1, ct_2)$.
- **IPFE.KeyGen(IPFE.msk, $\boldsymbol{y}$)** : *Sample* $t \xleftarrow{\$} \mathbb{Z}_p$. *Compute*

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

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

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

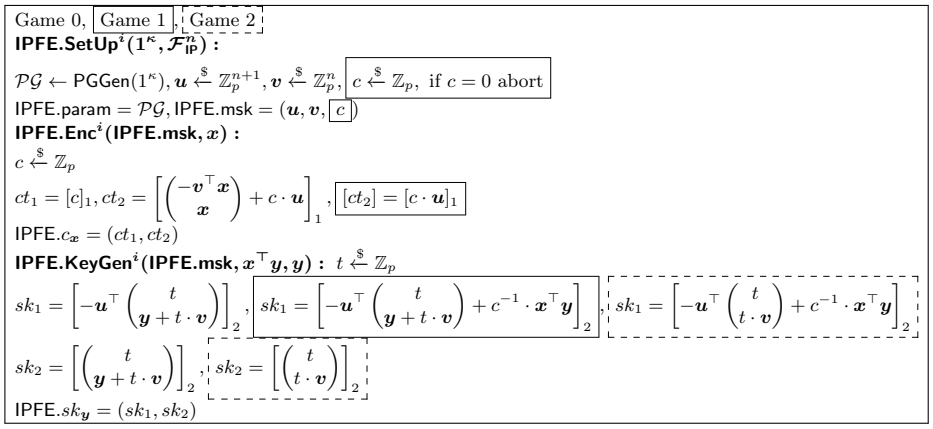

Fig. 2. Games for the security proof of the IPFE from Encryption Scheme 4.

4.2 Correctness and Security

Proposition 2. *The* IPFE *scheme defined in Encryption Scheme 4 is a correct functional encryption scheme for* $\mathcal{F}_{\mathsf{IP}}^n$.

For the proof we refer to Appendix B

Theorem 2. *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*

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

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

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 2 with changes in Game $i+1$ being over Game i. We show that $\mathsf{Exp}_{\mathcal{A}}^{real}(1^\kappa) =$ Game 0 $\approx_s$ Game 1 $\approx_c$ Game 2 $= \mathsf{Exp}_{\mathcal{A},\mathcal{S}}^{ideal}(1^\kappa)$.

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 $\mathsf{Adv}_{i(i+1)}(\mathcal{A}') := \left|\Pr\left[\tilde{b} = 1 | b = 0\right] - \Pr\left[\tilde{b} = 1 | b = 1\right]\right|$.

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 $\left[\begin{pmatrix} -\boldsymbol{v}^\top \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix} + c \cdot \boldsymbol{u}\right]_1$ to $[c \cdot \boldsymbol{u}]$ and to keep coherence we must change sk_1 from we must change

$sk_1 = \left[-\boldsymbol{u}^\top \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{y} + \boldsymbol{V}\boldsymbol{t} \end{pmatrix}\right]_2$ to $\left[-\boldsymbol{u}^\top \begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{y} + \boldsymbol{V}\boldsymbol{t} \end{pmatrix} + c^{-1} \cdot \boldsymbol{x}^\top \boldsymbol{y}\right]_2$. 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 $\boldsymbol{u}$ for $\widetilde{\boldsymbol{u}} := \boldsymbol{u} - c^{-1} \cdot \begin{pmatrix} -\boldsymbol{v}^\top \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix}$, since

$$\left[\begin{pmatrix} -\boldsymbol{v}^\top \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix} + c \cdot \widetilde{\boldsymbol{u}}\right]_1 = \left[\begin{pmatrix} -\boldsymbol{v}^\top \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix} + c \cdot \left(\boldsymbol{u} - c^{-1} \cdot \begin{pmatrix} -\boldsymbol{v}^\top \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix}\right)\right]_1 = [c \cdot \boldsymbol{u}]_1,$$

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

and coherence is indeed held given that

$$\begin{aligned} e(\widetilde{ct}_1, sk_1) &= \left[-c \cdot \boldsymbol{u}^\top \begin{pmatrix} t \\ \boldsymbol{y} + t \cdot \boldsymbol{v} \end{pmatrix} + \boldsymbol{x}^\top \boldsymbol{y}\right]_T, \\ e(\widetilde{ct}_2, sk_2) &= \left[c \cdot \boldsymbol{u}^\top \begin{pmatrix} t \\ \boldsymbol{y} + t \cdot \boldsymbol{v} \end{pmatrix}\right]_T \end{aligned}$$

which gives $[v]_T = [\boldsymbol{x}^\top \boldsymbol{y}]_T$ 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 $\boldsymbol{u}$ and $\widetilde{\boldsymbol{u}}$ are indistinguishable, so will be Game 0 and Game 1. Now, $\widetilde{\boldsymbol{u}}$ will exist as long as c has an inverse, which it will since $c \neq 0$, and given that $\boldsymbol{u}$ is sampled uniformly at random in $\mathbb{Z}_p^{n+1}$ and used only once (we are proving one selective security), then $\boldsymbol{u}$ and $\widetilde{\boldsymbol{u}}$ are indistinguishable. Therefore, for any PPT adversary $\mathcal{A}'$, $\mathsf{Adv}_{01}(\mathcal{A}') \leq 1/p$.

Game 2. In this Game we finish the simulation by changing in both sk_1 and sk_2 the vector $\begin{pmatrix} t \\ \boldsymbol{y} + t \cdot \boldsymbol{v} \end{pmatrix}$ to $\begin{pmatrix} t \\ t \cdot \boldsymbol{v} \end{pmatrix}$. This means that the encryption simulation can be performed without knowledge of $\boldsymbol{x}$ and the key generation simulation can be performed only knowing the output $\boldsymbol{x}^\top \boldsymbol{y}$.

More formally, we prove in Lemma 3 that for any PPT adversary $\mathcal{A}'$ there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{12}(\mathcal{A}') \leq 2Q_{sk} \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2} + 2Q_{sk}/p - 1$. The intuition is that using a Hybrid argument through the queries asked we use the n-fold DDH assumption (see Appendix A) to swap $t \cdot \boldsymbol{v}$ for a value sampled uniformly at random so we can remove $\boldsymbol{y}$.

Table 4. Efficiency estimates for our MIQFE and IPFE constructions.

	Secret key	Ciphertext	Functional key
Generic MIQFE	$\ell^2 \cdot \mathsf{IPFE}^{2n}_{\mathsf{msk}} + \ell(1+n)\lvert p\rvert + \ell^2 2n\lvert p\rvert$	$\ell \cdot \mathsf{IPFE}^{2n}_{c_{\boldsymbol{x}}} + n\lvert p\rvert$	$\ell^2 \cdot \mathsf{IPFE}^{2n}_{sk_{\boldsymbol{y}}} + \lvert p\rvert$
IPFE	$(2n+1)\lvert p\rvert$	$(n+2)\lvert\mathbb{G}_1\rvert$	$(n+2)\lvert\mathbb{G}_2\rvert$
Concrete MIQFE	$\ell^2(4n+1)\lvert p\rvert + \ell(1+n)\lvert p\rvert + \ell^2 2n\lvert p\rvert$	$\ell(2n+2)\lvert\mathbb{G}_1\rvert + n\lvert p\rvert$	$\ell^2 \cdot (2n+2)\lvert\mathbb{G}_2\rvert + \lvert p\rvert$

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

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

5 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 $\{\boldsymbol{u}_i\}_{i\in[\ell]} \in \mathbb{Z}_p^n$, $\{c_i\}_{i\in[\ell]} \in \mathbb{Z}_p$ and $\{\boldsymbol{w}_{i,j\in[\ell]}\} \in \mathbb{Z}_p^{2n}$; the ciphertexts need for $\boldsymbol{ct}_{\boldsymbol{x}_i} \in \mathbb{Z}_p^n$; and the functional keys need for $zk_{\boldsymbol{F}} \in \mathbb{Z}_p$.

Our function-hiding IPFE scheme supporting n coefficients, needs in the master secret key for $\boldsymbol{u} \in \mathbb{Z}_p^{n+1}$ and $\boldsymbol{v} \in \mathbb{Z}_p^n$; 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 4, where IPFE^k_s denotes the size of the element referred by s of the base IPFE scheme for k coefficients.

6 Conclusion

These results may inspire further work in the subject. 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.

Finally, our results can be of use for applications where a simulation secure MIQFE is needed. As an example, recently a simulation secure MIPFE was used to instantiate an efficient randomized functional encryption scheme able to respond differentially private linear queries [9]. As such, a randomized functional encryption scheme able to respond differentially private quadratic queries is a potential future application.

Acknowledgements We thank Hieu Phan for the useful and helpful discussions concerning this work. This work was funded by the French ANR Project ANR-23-CE39-0009-06 TRUST, the France 2030 ANR Project ANR-22-PECY-003 SecureCompute and the French ANR Project ANR-21-CE39-0006 SANGRIA.

References

1. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) Public-Key Cryptography – PKC 2015. pp. 733–751. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_33
2. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: 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
3. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 597–627. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_20
4. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 601–626. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_21
5. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: New perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 500–518. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_28
6. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 208–238. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-84259-8_8
7. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography. pp. 711–740. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-22318-1_25
8. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 333–362. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
9. Alborch Escobar, F., Canard, S., Laguillaumie, F., Phan, D.H.: 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
10. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 152–181. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_6

11. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 67–98. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
12. Benhamouda, F., Bourse, F., Lipmaa, H.: Cca-secure inner-product functional encryption from projective hash functions. In: Fehr, S. (ed.) Public-Key Cryptography – PKC 2017. pp. 36–66. Springer, Berlin, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_2
13. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 470–491. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_20
14. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) Theory of Cryptography. pp. 253–273. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
15. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 733–764. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_25
16. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) Public-Key Cryptography–PKC 2016. pp. 164–195. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_7
17. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography – PKC 2020. pp. 95–120. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_4
18. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. pp. 578–602. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_32
19. Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie–hellman assumptions. Journal of Cryptology **30**(1), 242–288 (2017). https://doi.org/10.1007/s00145-015-9220-6
20. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) Security and Cryptography for Networks. pp. 544–562. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_29
21. Kim, S., Kim, J., Seo, J.H.: 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
22. Lin, H.: Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 599–629. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_20
23. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 630–660. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_21

24. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Paper 2010/556 (2010), https://eprint.iacr.org/2010/556
25. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) Theory of Cryptography. pp. 457–473. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_27
26. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) Information Security. pp. 408–425. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-45871-7_24
27. Ünal, A.: Impossibility results for lattice-based functional encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 169–199. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_7
28. Zhao, Q., Zeng, Q., Liu, X., Xu, H.: Simulation-based security of function-hiding inner product encryption. Science China. Information Sciences **61**(4), 048102 (2018)

A The DDH Assumption

Let p be a prime number, we define the following distribution, using the framework from [19]. The DDH distribution over $\mathbb{Z}_p^2$ samples $t \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\boldsymbol{t} := (1, t)^\top$.

Definition 7. *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*

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

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

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

Definition 8. *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*

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

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

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

Lemma 1 (Lemma 1,[19]). *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*

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

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

B Proofs of Correctness

B.1 MIQFE scheme

Proposition 3. *The* MIQFE *scheme defined in Section 3.1 is a correct multi-input functional encryption scheme for $\mathcal{F}^{n,\ell}_{\mathsf{Q},B}$ as long as* IPFE *is a correct scheme for $\widetilde{\mathcal{F}}^{2n}_{\mathsf{IP}}$.*

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

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

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

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

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

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

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

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

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

B.2 IPFE scheme

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

Proof. The first pairing operation gives us

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

and the second pairing operation gives us

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

This in turn gives us $[v]_T = \left[\boldsymbol{x}^\top \boldsymbol{y}\right]_T$.

C Proofs of Auxiliary Lemma

C.1 Proof of Lemma in Section 3

Lemma 2. *For any* PPT *adversary* $\mathcal{A}'$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that*

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

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 3. 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 $\approx_c \mathsf{H}_1^1 \approx_c \ldots \approx_c \mathsf{H}_1^\ell \approx_c \mathsf{H}_2^1 \approx_c \ldots \approx_c \mathsf{H}_\ell^\ell =$ 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 $\mathsf{H}_\eta^{\iota-1}$, 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 $\mathsf{Adv}_{(\iota-1)\iota}^{\mathsf{H}}(\mathcal{A}') := |\Pr[\tilde{b} = 1 | b = 0] - \Pr[\tilde{b} = 1 | b = 1]|$ for $\iota \in [\ell]$. We define analogously $\mathsf{Adv}_{(i-1)i}^{\rho}(\mathcal{A}')$ for distinguishing between Hybrid H_{i-1}^2 and Hybrid H_i^0.

Formally speaking, going from hybrid $\mathsf{H}_\eta^{\iota-1}$ to hybrid H_η^ι 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 $c_\eta(\boldsymbol{u}_\eta - \widetilde{\boldsymbol{u}}_\eta) = \boldsymbol{x}_\eta$ by definition of $\widetilde{\boldsymbol{u}}_\eta$. Furthermore, the plaintext

Hybrid $\mathsf{H}^{\iota}_{\eta}$

MIQFE.SetUp$^{i}(1^{\kappa}, \mathcal{F}^{n}_{B,\ell})$:

$\mathcal{PG} \leftarrow \mathsf{PGGen}(1^{\kappa}), \boldsymbol{u}_i \xleftarrow{\$} \mathbb{Z}_p^n$ for $i \in [\ell], \boldsymbol{w}_{i,j} \xleftarrow{\$} \mathbb{Z}_p^{2n}$ for $i, j \in [\ell]$ $c_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [\ell]$. If any $c_i = 0$ abort.

-if $(i,j) <_L (\eta, \iota)$: $\mathsf{IPFE.msk}'_{i,j} \leftarrow \mathsf{IPFE.SetUpSim}(1^{\kappa}, \widetilde{\mathcal{F}}^{2n}_{\mathsf{IP}}, \mathcal{PG})$,

-if $(i,j) = (\eta, \iota)$: $\boxed{\mathsf{IPFE.msk}'_{i,j} \leftarrow \mathsf{IPFE.SetUpSim}(1^{\kappa}, \widetilde{\mathcal{F}}^{2n}_{\mathsf{IP}}, \mathcal{PG})}$,

-if $(i,j) >_L (\eta, \iota)$: $\mathsf{IPFE.msk}'_{i,j} \leftarrow \mathsf{IPFE.SetUp}(1^{\kappa}, \widetilde{\mathcal{F}}^{2n}_{\mathsf{IP}}, \mathcal{PG})$,

$\mathsf{MIQFE.msk} = \left(\{\boldsymbol{u}_i, c_i\}_{i \in [\ell]}, \{\boldsymbol{w}_{i,j}, \mathsf{IPFE.msk}'_{i,j}\}_{i,j \in [\ell]}\right)$

MIQFE.Enc^{i}(MIQFE.msk, $\boldsymbol{i}, \boldsymbol{x_i}$) :

$\widetilde{\boldsymbol{u_i}} = \boldsymbol{u}_i - c_i^{-1}\boldsymbol{x}_i, \widetilde{ct}_{\boldsymbol{x}_i} = c_i \boldsymbol{u}_i, \mathsf{st}_i = \widetilde{u}_i$

-if $(i,j) <_L (\eta, \iota)$: $\mathsf{IPFE}.c'_{i,j} \leftarrow \mathsf{IPFE.EncSim}(\widetilde{\mathsf{IPFE.msk}}_{i,j})$

-if $(i,j) = (\eta, \iota)$: $\boxed{\mathsf{IPFE}.c'_{i,j} \leftarrow \mathsf{IPFE.EncSim}(\widetilde{\mathsf{IPFE.msk}}_{i,j})}$

-if $(i,j) >_L (\eta, \iota)$: $\mathsf{IPFE}.c'_{i,j} \leftarrow \mathsf{IPFE.Enc}\left(\mathsf{IPFE.msk}_{i,j}, \boldsymbol{w}_{i,j} + \begin{pmatrix} c_j \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_j \boldsymbol{x}_i \end{pmatrix}\right)$

$\mathsf{MIQFE}.c_i = (\widetilde{ct}_{\boldsymbol{x}_i}, \{\mathsf{IPFE}.c'_{i,j}\}_{j \in [\ell]})$

MIQFE.KeyGen^{i}(MIQFE.msk, $\{\mathsf{st}_i\}_{i \in [\ell]}, \boldsymbol{v}, \boldsymbol{F}$) : For $i, j \in [\ell]$

$$d_{i,j} = \left(\boldsymbol{w}_{i,j} + \begin{pmatrix} c_j \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_j c_i (\boldsymbol{u}_i - \widetilde{\boldsymbol{u}}_i) \end{pmatrix}\right)^{\top} \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^{\top} \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}, zk_{\boldsymbol{F}} = \sum_{i,j \in [\ell]} \boldsymbol{w}_{i,j}^{\top} \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^{\top} \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}$$

-if $(i,j) <_L (\eta, \iota)$: $\mathsf{IPFE}.sk'_{i,j} \leftarrow \mathsf{IPFE.KeyGenSim}(\widetilde{\mathsf{IPFE.msk}}_{i,j}, d_{i,j})$

-if $(i,j) = (\eta, \iota)$: $\boxed{\mathsf{IPFE}.sk'_{i,j} \leftarrow \mathsf{IPFE.KeyGenSim}(\widetilde{\mathsf{IPFE.msk}}_{i,j}, d_{i,j})}$

-if $(i,j) >_L (\eta, \iota)$: $\mathsf{IPFE}.sk'_{i,j} \leftarrow \mathsf{IPFE.KeyGen}\left(\mathsf{IPFE.msk}_{i,j}, \begin{pmatrix} \widetilde{\boldsymbol{u}}_j^{\top} \boldsymbol{F}_{j,i} \\ \boldsymbol{F}_{i,j} \widetilde{\boldsymbol{u}}_j \end{pmatrix}\right)$

$\mathsf{MIQFE}.sk_{\boldsymbol{F}} = (\boldsymbol{F}, \{\mathsf{IPFE}.sk'_{i,j}\}_{i,j \in [\ell]}, zk_{\boldsymbol{F}})$

Fig. 3. Games for the hybrid argument in Lemma 2. Changes are squared.

$\boldsymbol{w}_{i,j} + \begin{pmatrix} c_j \widetilde{\boldsymbol{ct}}_{\boldsymbol{x}_i} \\ c_j \boldsymbol{x}_i \end{pmatrix}$ is indistinguishable from random since $\boldsymbol{w}_{i,j}$ 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 $\mathsf{H}^{\iota-1}_{\eta}$ and $\mathsf{H}^{\iota}_{\eta}$ 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 $\mathsf{Adv}^{\mathsf{H}}_{(\iota-1)\iota}(\mathcal{A}') \leq \mathsf{Adv}^{\mathsf{FH\text{-}SIM}}_{\mathsf{IPFE}}(\mathcal{B})$.

To finalize the proof we note that going from hybrid $\mathsf{H}^{\ell}_{\eta-1}$ to hybrid H^{1}_{η} and going from Game 1 to hybrid H^{1}_{1} are analogous to going from hybrid $\mathsf{H}^{\iota-1}_{\eta}$ to $\mathsf{H}^{\iota}_{\eta}$ 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

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

C.2 Proof of Lemma in Section 4

Lemma 3. *For any* PPT *adversary* $\mathcal{A}'$ *there exists a* PPT *adversary* $\mathcal{B}$ *such that*

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

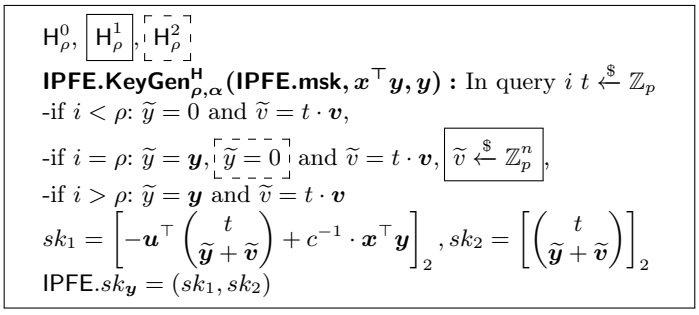

Fig. 4. Games for the hybrid argument in Lemma 3. Changes only occur on the KeyGen algorithm so the others are omitted and assumed the same as in Game 1 in Figure 2.

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

Proof. We will prove this through a series of Hybrid Games H^α_ρ for $\rho \in [Q_{sk}]$ and $\alpha \in \{0, 1, 2\}$, defined in Figure 4. We show that Game 1 $= \mathsf{H}^0_1 \approx_c \mathsf{H}^1_1 \approx_s \mathsf{H}^2_1 \approx_c \mathsf{H}^0_2 \approx_c \ldots \approx_c \mathsf{H}^2_{Q_{sk}} \approx_c = \mathsf{H}^0_{Q_{sk}+1}$ 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^{i-1}_ρ, otherwise it interacts as in Hybrid H^i_ρ. At the end of the interaction, $\mathcal{A}'$ will make its guess $\tilde{b} \in \{0, 1\}$. We define $\mathsf{Adv}^\alpha_{(i-1)i}(\mathcal{A}') := |\Pr[\tilde{b} = 1|b = 0] - \Pr[\tilde{b} = 1|b = 1]|$ for $i = 1, 2$. We define analogously $\mathsf{Adv}^\rho_{(i-1)i}(\mathcal{A}')$ for distinguishing between Hybrid H^2_{i-1} and Hybrid H^0_i.

Hybrid H^1_ρ . In this change we have swapped $\widetilde{\boldsymbol{v}}$ from $t \cdot \boldsymbol{v}$ 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 $[\boldsymbol{t}]_2 = [(t, 1)^\top]_2$, $[\boldsymbol{W}]_2 := [(\boldsymbol{w}_1, \boldsymbol{w}_2)^\top]_2$ for some $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{Z}_p^n$, it runs $\mathsf{IPFE.SetUp}^{\mathsf{H}}_{\rho,1}$ and $\mathsf{IPFE.Enc}^{\mathsf{H}}_{\rho,1}$ and in $\widetilde{\mathsf{IPFE.msk}}$ it substitutes $\boldsymbol{v}$ for $[\boldsymbol{w}_2]_2$. Note that only having the element in $\mathbb{G}_2$ is not an issue since after Game 1 $\boldsymbol{v}$ 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 [1, Section 3] instead of the order in [22, Section 6.3] comes into play.

Then for query $i \in [Q_{sk}]$, if $i \neq \rho$ it runs $\mathsf{KeyGen}^{\mathsf{H}}_{\rho,1}$ normally, but if $i = \rho$ it sets $[t]_2$ as the first part of the n-fold DDH challenge and $\widetilde{\boldsymbol{v}} = [\boldsymbol{w}_1]_2$. 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 $[\boldsymbol{W}]_2$ is of the form $[\boldsymbol{t}\boldsymbol{r}^\top]_2 = [(t\boldsymbol{r}, \boldsymbol{r})^\top]_2$ for some $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^n$ then it is the same distribution as in Hybrid H^0_ρ while if $[\boldsymbol{W}]_2$ is uniformly at random it is the same distribution as in Hybrid H^1_ρ. As such we

get

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

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

Hybrid H^2_ρ . In this change we have swapped $\boldsymbol{y}$ from $\boldsymbol{y}$ to 0. Distinguishing between these Hybrids is distinguishing between $\boldsymbol{y} + \widetilde{\boldsymbol{v}}$ and $\widetilde{\boldsymbol{v}}$, and since $\widetilde{\boldsymbol{v}}$ is sampled uniformly at random and only used once they are indistinguishable. Therefore, for any PPT adversary $\mathcal{A}'$, $\mathsf{Adv}^{\alpha}_{12}(\mathcal{A}') = 0$.

Hybrid $\mathsf{H}^0_{\rho+1}$. In this change we have swapped back $\widetilde{v}$ from sampled uniformly at random to $t \cdot \boldsymbol{v}$. As such this transition is analogous to the change from H^0_ρ to H^1_ρ, so for any PPT adversary $\mathcal{A}'$ there exists a PPT adversary $\mathcal{B}$ such that

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

Finally, adding all the transitions together and noting that H^0_1 is identically distributed to Game 1, while $\mathsf{H}^0_{Q_{sk}+1}$ is identically distributed to Game 2, we get

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

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