2-Step Multi-Client Quadratic Functional Encryption from Decentralized Function-Hiding Inner-Product

Michel Abdalla[1,2][0000−0002−2447−4329], David Pointcheval[1,2][0000−0002−6668−683X], and Azam Soleimanian[1,2][0000−0001−9881−6435]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
{michel.abdalla,david.pointcheval,azam.soleimanian}@ens.fr
[2] INRIA, Paris, France

Abstract. In this paper, we present a multi-client quadratic functional encryption (MCQFE) scheme from function-hiding inner-product (FHIP). The main challenge in such construction is that all the clients require the access to the master secret key of the underlying FHIP scheme, which clearly breaches the security.
To overcome this challenge, we present an efficient decentralized version of FHIP scheme of Lin (Crypto 16). This leads to a 2-step MCQFE (2-MCQFE) scheme. In a 2-step MCQFE scheme, the encryption phase is a (non-interactive) protocol among clients and a set of honest-but-curious authorities. More precisely, clients are the owner of messages and the master secret-key of the underlying FHIP is shared among authorities. In the first step, the client publishes a pre-ciphertext pct associated with its message. Then in the second step, each authority generates its share ct_i extracted from the pre-ciphertext. The public aggregation of these shares ct_i will generate the target ciphertext ct which then would be applied on the functional key sk_F to compute the quadratic functionality. The security model is strong enough to consider no trust among clients and authorities, and also the revelation of some secret keys (of clients or authorities) through corruptions. We instantiate our 2-MCQFE scheme and prove its security in the random-oracle model based on the SXDH assumption. Moreover, we show that its security holds as long as at least one of the authorities is not corrupted.

1 Introduction

Functional Encryption. Functional encryption is a strong and general tool enabling computation over encrypted data with non-interactive decryption[3]. Given a functional-key sk_F and ciphertext ct_m, everyone can compute $f_F(m)$ in the clear where the system is parameterized by the functionality f.

The idea of functional encryption originated as an extension of Identity-Based Encryption (IBE) [11, 29], Searchable Encryption [1, 10], Attribute-based encryption [22] and Predicate Encryption [21, 23]., where a special form of functionality f in FE can specify it as the mentioned encryption systems. On the other hand, all known constructions supporting general functionality, mainly suffer from: inefficiency, relying on strong assumptions or some limitations on the number of collusions [16, 19, 20]. Thus, as a trade-off one should see FE as a general concept but try to focus on the other special but still wield classes of functionalities. As the special cases of functionality in FE, *inner-product* [3, 6] and *quadratic functionality*

[3] Homomorphic Encryption is another tool for computation over encrypted data, where for the decryption it needs to interact with the owner of the secret-key. While in FE everyone holding the ciphertext and functional-key can decrypt the message.

[7,17,26] have attracted more attention due to their use in real-world applications and other theoretical primitives [8,26].

Inner-Product and Quadratic Functionalities. When a FE system is parameterized by the inner-product functionality (IPFE) [3,6], the ciphertext ct_x and the functional-key sk_y are associated with vectors with the same dimension, namely $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ (res.). Then, the decryption returns $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i$. While the traditional security of IPFE mainly focuses on the privacy of message $\mathbf{x}$, the security requirement for function-hiding IP (FHIP) is stronger and it concerns the privacy of both vectors $\mathbf{x}$ and $\mathbf{y}$. For quadratic functionality [7,26], the ciphertext and the functional-key are respectively associated with message $(\mathbf{x}, \mathbf{y}) \in \mathcal{M}^n \times \mathcal{M}^m$ and the matrix $\boldsymbol{F} \in \mathcal{F}^{n \times m}$ where $\mathcal{M}, \mathcal{F}$ are the message and function space. The decryption algorithm returns $f_F(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \boldsymbol{F} \mathbf{y}$. Most of the existing works have focused on IPFE ([3,6,24,28]), while FE for the quadratic functionality (QFE) has got less attention. To the best of our knowledge, so far there are four works in this field: Lin [26] presented a (single-input) QFE scheme from FHIP based on the SXDH assumption and in the standard model. In [7], authors present a QFE scheme based on the MDDH and 3-party DDH assumptions in the standard model. The more efficient construction of [7] and also the construction in [17] are proved to be secure in the generic group model. In [17], the QFE scheme is also based on the SXDH assumption.

Multi-Client FE. In a multi-client setting of FE (MCFE), message comes from different sources namely, there are (polynomially) many clients who do not trust each other [4,5,12,13,18]. Each client is assigned a secret-key enabling the autonomous encryption secure against other clients. In MCFE, each ciphertext is also associated with a label giving a good flexibility to control the leaked information. More precisely, while for the same label one can have a kind of mix-and-match among messages from different clients, for different labels this would not be possible.

1.1 Contributions

While MCFE for IP has been well studied [2,4,12,13,25], there is no construction for multi-client quadratic FE (MCQFE). In this paper:

An Extended Syntax and its Applications. We introduce an extended syntax of MCQFE such that the encryption phase is a protocol among clients and some authorities. This syntax can overcome some difficulties (in multi-client setting) by allowing some communications during the encryption phase. We also present some applications to show how the mentioned syntax can be used/realized by the real world problems.

Efficient Non-Interactive Instantiation. We instantiate our suggested syntax without any interactions among clients and authorities. More precisely, clients are considered to be the owner of data while the authorities, holding their own secret-keys, are in charge of some computations over encrypted data from clients. Such instantiation also avoid the trust among clients and authorities. The main security requirement is that at least one of the authorities should not collude with others (should not be corrupted) to recover a master secret-key shared among them. Our presented applications explain how this requirement can be fulfilled naturally or by some strategies. Our instantiation is inspired from single-input QFE scheme of Lin [26], while its extension to multi-client setting can be challenging, mainly due to the

corruption of secret-keys (collusion of parties against each other). We explain these challenges and how to go around them (see Section 1.3). Moreover, the size of the ciphertext in our scheme is linear w.r.t the number of clients/slots, making it more interesting to be used in the real-world applications.

Decentralized FHIP. Our instantiation needs a decentralized FHIP (d-FHIP) as a building block. Thus, we present a decentralized FHIP scheme in the standard model and based on the SXDH assumption. We believe this d-FHIP scheme also can be of independent interest, since it is the first decentralized FHFE construction efficiently realizing the inner-product functionality.

1.2 Scenario and Applications

In a standard syntax of MCQFE scheme, each client encrypts its message by its own secret-key and without any help from other parties. We extend this syntax where the encryption phase is a protocol among clients and a set of authorities. The client is the owner of the message and each authority owns an individual secret-key. Clients can generate the target ciphertexts communicating with authorities. While this new syntax may help to go around some challenges, it make sense only if it can be realized by constructions with a reasonable security level, efficiency and number of interactions. Our suggested 2-MCQFE construction realizes such syntax with efficient computational overhead and number of communications (see Fig. 1). The suggested security model avoids any trust among clients and authorities. Moreover, the adversary can corrupt a set of clients and authorities to access some secret-keys as well.

A curious reader may ask why we need to go for such extension and whether it was not possible to have a MCQFE in its standard syntax. Though we do not present any impossibility result in this paper, we discuss the main challenges giving the intuition that the instantiation of MCQFE in its classical syntax and without 3-linear maps can be hard.

First of all, note that a trivial QFE scheme is to encrypt all the multiplications $x_i y_j$ for $i \in [n], j \in [m]$ by a IPFE scheme, though due to its quadratic ciphertext-size (w.r.t the number of slots) it is not interesting. This trivial scheme (even though it is of quadratic ciphertext-size) does not work for the multi-client setting as x_i and y_j come from different users. Moreover, in the following we see how one can probably get MCFE from 3-linear maps in a not so trivial way.

Generally speaking, in a QFE scheme, to make different combinations possible (in an efficient way[4]), each part is encrypted separately as ct_{x_i} and ct_{y_j} (i.e., $\mathsf{ct}_{x,y} = (\mathsf{ct}_0, \mathsf{ct}_{x_i}, \mathsf{ct}_{y_j})_{i,j}$ where ct_0 is either empty or the encoding of some randomness). We believe that two main cases are possible based on the existing works:
(1) either the ciphertexts ct_{x_i} and ct_{y_j} use different randomness (e.g., r for all x_i and r' for all y_j), but a combination of these randomness should be embedded in the ciphertext as ct_0 (or instead, they may use related randomnesses like a matrix and its inverse). Having this combination in ct_0 allows to combine ciphertexts associated with x_i or y_j. This strategy is used in [7, 17].
(2) or the ciphertexts are generated based on the same secret-key allowing to combine the ciphertexts [26].

For a multi-client setting in the first case, we need separate encodings of randomnesses (e.g., $[r]_1$ and $[r']_2$ generated by random oracles) such that they can be combined during

[4] See the mentioned trivial QFE scheme to understand why we need to encrypt each part separately.

the decryption (by pairing as $[rr']_T$). The problem here is that in the decryption we need to combine the randomness with a functional-key sk_F (see [7, 17]) and for the security reason, specially in the multi-client setting that corruptions are possible, sk_F is also involved with some encodings. This means that for the schemes falling in this category [7, 17], one probably can avoid communications through other tools like 3-linear maps, which is out of the scope of this paper.

In the second case, as the clients are not trusted, instead of giving them the same secret-key, one may share the secret-key among some authorities, and add some communications to generate the shares of ciphertexts associated with data from each client. In this paper we are following this idea. We intuitively believe, this strategy needs reasonable number of communications, first because the number of authorities can be independent of the number of clients, and second we can generate and share the key once in the setup phase. Particularly, the linear and homomorphic property of building blocks in our instantiation, allows the authorities to generate and aggregate their shares without communicating with each other.

Fig. 1 explains the scenario of our construction called 2-step MCQFE (2-MCQFE). In a 2-MCQFE scheme, the encryption protocol is a 2-step process: in the first step, each client (independently) publishes a pre-ciphertext pct_m of its message m. Then, in the second step, the authority i receiving pct_m generates its share as $\mathsf{ct}_{m,i}$. The decryption is done without any help from authorities: first, the shares $\mathsf{ct}_{m,i}$ are aggregated to generate a target ciphertext ct_m. Then, the given functional-key sk_F is applied over ciphertexts ct_m to compute the value of the quadratic functionality. The aggregation phase does not need any secret information and can be run by the decryptor or by clients where each client aggregates the shares concerning its own message. Clearly, the latter one needs one round of interaction among clients and authorities, and thus we prefer to put the aggregation on the decryptor side. We instantiate this scenario based on SXDH assumption in the random oracle model such that our presented 2-MCQFE is secure as long as at least one of the authorities is not colluding with others (is not corrupted).

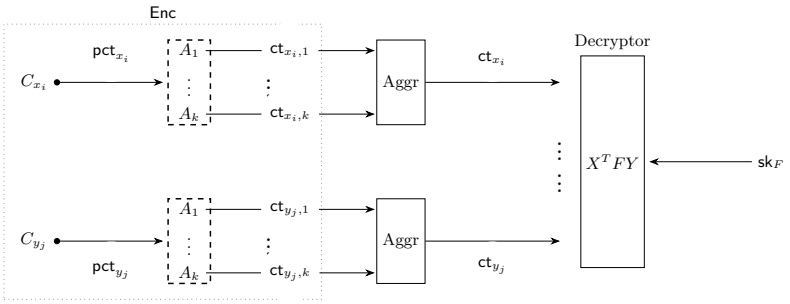

Fig. 1: Scenario of our 2-MCQFE construction. Client C_{x_i} and C_{y_j} independently generate their pre-ciphertexts pct_{x_i} and pct_{y_j}. Then, $\mathsf{ct}_{m,t}$ is the share of authority A_t extracted from pct_m, and ct_m is the aggregation of $\{\mathsf{ct}_{m,t}\}_t$.

Applications. As an application of 2-MCQFE scenario, one can think of data classification where the messages are strings of bits and $\sum_{i,j} f_{i,j} x_i y_j$ computes (weighted) similarity between two collections $\mathbf{x}$ and $\mathbf{y}$ of data. As a special example, imagine that for the sake of a global research, World Health Organization (WHO) needs to collect some medical records from different countries to analyze the behavior of a pandemic or disorder based on similarity of data from different regions. They make a committee, where each country has its own

representative. Each hospital (pre-) encrypts its data and sends it to the committee, were the committee can compute the target ciphertext and send it to WHO. Then WHO uses these ciphertexts to extract its required information regarding the similarities. In this example, though a representative may be interested in getting access to the data from its own country but at the same time it does not want others to access this information (criticizing their crisis-management, e.g.). Our 2-MCQFE scheme leads to a "no-one or every-one" situation where if all the representatives (playing the role of authorities) collude to reveal their secret-key in order to get access to the information from its country or others, it gives this privilege to others as well.

Our 2-MCQFE scenario can be generally used in the decentralized data storage or classified search. In a decentralized storage system, there is a set of servers in charge of data storage. Finding the similarities among data from different clients allows for the data classification during the storage phase which then can reduce the storage-space, retrieval or search time.In this example, servers play the role of authorities. If the servers have no interest in the privacy of stored data, one can use a self-enforcement methodology to avoid them from revealing their 2-MCQFE secret-key. For example, each server has to encrypt a valuable information, like its secret-key associated with a deposit, and gives a zero-knowledge proof that they have encrypted this secret-key. Then the ciphertext would be stored on the storage as well. This forces the server to care about the privacy of data and do not reveal its 2-MCQFE secret-key share.

1.3 Overview and Challenges

Here we give an overview of our 2-MCQFE scheme. The FHIP scheme of Lin (Lin-FHIP) [26] is the main building block of our 2-MCQFE construction. In Lin-FHIP scheme, ct_x associated with message $\mathbf{x}$, belongs to the group $\mathbb{G}_1$ while ct_y associated with function $\mathbf{y}$, belongs to $\mathbb{G}_2$. The decryption algorithm needs a discrete-logarithm (DLog) computation to return $\langle \mathbf{x}, \mathbf{y} \rangle$.

For the sake of generality, we divide the clients into two main categories called: x-side and y-side clients. Where the i-th client on the x-side encrypts x_i, and similarly the j-th client on the y-side encrypts the message y_j. To encrypt the message x_i, the client i (on the x-side) uses its secret-key U_i to build $x'_i = (x_i || [\alpha]_1 U_i)$ where $[\alpha]_1 = \mathcal{H}(\ell) \in \mathbb{G}_1$ is generated by the random oracle providing the access to the same randomness for all the clients. Similarly, the j-th client on the y-side builds $y'_j = (y_j || T_j)$ holding its secret-key T_j. Then seeing x'_i and y'_j as the message and function of Lin-FHIP scheme, they encrypt their messages and output the results as ct_{x_i} and ct_{y_j} (res.). The decryption of Lin-FHIP over ct_{x_i} and ct_{y_j} returns $A_{ij} = [x_i y_j + \alpha U_i T_j]_T$. Now generating the functional-key sk_F as $[\sum f_{ij} U_i T_j]_2$, associated with the matrix $\boldsymbol{F} = [f_{ij}]_{i,j}$, allows to recover $\sum f_{ij} x_i y_j$ holding A_{ij} for all i and j.

This is so similar to the idea of the single-input QFE scheme of Lin [26]. We progressively change this construction confronting the security requirements and challenges in the multi-client setting, which can be summarized as follows:

FHIP in the Public-Key Setting. Note that the clients can not have access to the same master secret-key of FHIP scheme, since the adversary can clearly breaches the security by corrupting only one client. On the other hand, it is required that the decryption of Lin-FHIP should be possible over ct_{x_i} (as the ciphertext for FHIP) and ct_{y_j} (as the functional-key for FHIP) for all i and j. We call this requirement as the *mix-and-match* property which can be achieved if all the values ct_{x_i} and ct_{y_j} are generated via the same master secret-key of FHIP scheme. To satisfy these conflicting requirements, we propose a decentralized FHIP

scheme where the master secret-key is shared among some authorities and the clients have access only to the master public-key. Each authority, holding its secret-key, produces its own share of the inner-product value, these shares then can be aggregated to obtain the final inner-product value. The linear property of inner-product and homomorphic property of Abdalla et al. [3] scheme (as the building block for Lin-FHIP), makes the generation and aggregation of ciphertext-shares possible, without any interaction among authorities.

For the general functionality, a similar idea was introduced in [15] to present a decentralized FHFE scheme using spooky encryption and based on the LWE assumption in the common reference string (CRS) model. Here we study an efficient construction for the special case of inner-product functionality based on SXDH assumption in the standard model.

Corruption Queries. In the security proof, due to the mix-and-matches among ct_{x_i} and ct_{y_j}, one can expect that changing x_i^0 to x_i^1 should be involved with a hybrid over index j. For this hybrid, we use Matrix-DDH (MDDH) assumption to give the same structure to vectors T_j for $j \neq j^*$ as $T_j = [\mathbf{b}^T r_j]_2$ while T_{j^*} is uniformly sampled from $\mathbb{G}_2$. This would allows us to treat all the indices $j \neq j^*$ in the same way. While for index j^* we can change $x_i^0 y_{j^*}^0$ to $x_i^1 y_{j^*}^0$. On the other hand, we can simulate the corruption queries on the secret-keys T_j if and only if T_j belongs to $\mathbb{G}_2$ (and not $T_j \in \mathbb{Z}_q$ where in the above $[T_j]_2$ is simulated). This is because the samples for MDDH belong to the algebraic group $\mathbb{G}_2$ which are used to simulate T_j.

Note that the similar reasoning works for changing y_j^0 to y_j^1. While the previous strategy (i.e., considering $U_i \in \mathbb{G}_1$ instead of $U_i \in \mathbb{Z}_q$) does not work here, since the output of the random oracle is already in the group $\mathbb{G}_1$ and so the computation of ct_{x_i} for $U_i \in \mathbb{G}_1$ is not possible. Instead, we extend the message x_i' and y_j' as: $x_i' = (x_i || [\alpha]_1 U_i || U_i')$ and $y_j' = (y_j || T_j || T_j' [\beta]_2)$ where $[\beta]_2$ is generated by another random oracle, and T_j and U_i' belong to $\mathbb{G}_2$ and $\mathbb{G}_1$, respectively. We also need to modify the functional-key as $\mathsf{sk}_F = ([\sum f_{ij} U_i T_j]_2, [\sum f_{ij} U_i' T_j']_1)$.
Putting together, our construction (see Fig. 1) can be abstractly presented as:

$$\mathsf{ct}_{x,y} = \begin{cases} \mathsf{ct}_{x_i} = \{\mathsf{ct}_{x_i,k}\}_k \leftarrow \mathsf{dFH.Enc}(x_i || \boldsymbol{\alpha} U_i || U_i') & \forall i \\ \mathsf{ct}_{y_j} = \{\mathsf{ct}_{y_j,k}\}_k \leftarrow \mathsf{dFH.KeyGen}(y_j || T_j || T_j' \boldsymbol{\beta}) & \forall j \end{cases}$$

$$\mathsf{sk}_F = ([\sum f_{ij} U_i T_j]_2, [\sum f_{ij} U_i' T_j']_1)$$

where $\mathcal{H}_\alpha(\ell) = [\boldsymbol{\alpha}]_1$, $\mathcal{H}_\beta(\ell) = [\boldsymbol{\beta}]_2$ are random oracles and k stands for the index of the authority in the decentralized FHIP scheme dFH. And as we discussed U_i' and T_j are vectors respectively in groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Exponentially many labels with optimal communications. In the (single-input) QFE scheme of Lin, the master secret-key of the underlying FHIP scheme is generated freshly for each ciphertext. This means in our extension to the multi-client setting, the master secret-key of the underlying decentralized FHIP should be sampled freshly and shared for each label. Therefore; either we should limit the number of labels to polynomially many labels such that the master secret-key of FHIP is shared in the setup phase, for all the labels. Or accept more communications during the encryption phase to share the master secret-key for each label. We avoid this issue, by increasing the length of the message and the dimension of secret-keys, which then allows us to use a long-term master secret-key of the decentralized FHIP scheme, for all the labels.

2 Preliminaries

Notations. In this paper, κ stands for the security parameter. The inner-product of two vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$. All the algorithms are considered probabilistic-polynomial-time (PPT). For an PPT algorithm A, the notation $y \leftarrow A(x)$ means, on input x the algorithm A outputs y. For a given set X, $x \xleftarrow{R} X$, stands for uniform sampling of x from X. Two strings or values are concatenated by $(\cdot||\cdot)$. We define $[n] = \{1, \ldots n\}$. For a vector $\mathbf{x} = (x_1, \ldots, x_n)$ we often use $\mathbf{x} = (x_i)_i$ or $\mathbf{x} = \{x_i\}_i$, and $|\mathbf{x}| = n$ stands for the dimension of the vector. The notation $\cong$ is used to show the indistinguishability of two distributions.

Definition 1 (Matrix Distribution [14]). *Let $\ell, k \in \mathbb{N}$ with $\ell > k$. We call $\mathsf{D}_{\ell,k}$ a matrix distribution if it outputs (in polynomial time and with overwhelming probability) matrices in $\mathbb{Z}_p^{\ell \times k}$ of full rank k. We define $\mathsf{D}_k = \mathsf{D}_{k+1,k}$.*

Definition 2 ($\mathsf{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption [14]). *Let $\mathsf{D}_{\ell,k}$ be a matrix distribution. We define the advantage of an adversary $\mathcal{A}$ for the $\mathsf{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption in the following way:*

$$\mathsf{Adv}^{MDDH}_{\mathsf{D}_{\ell,k},\mathcal{A}}(\kappa) := |\Pr[\mathcal{A}(1^\kappa, \mathcal{G}, [\mathbf{A}], [\mathbf{A}\boldsymbol{w}]) = 1] - \Pr[\mathcal{A}(1^\kappa, \mathcal{G}, [\mathbf{A}], [\boldsymbol{u}]) = 1]|,$$

where $\mathcal{G} = (\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\kappa), \mathbf{A} \leftarrow \mathsf{D}_{\ell,k}, \boldsymbol{w} \leftarrow \mathbb{Z}_p^k, \boldsymbol{u} \leftarrow \mathbb{Z}_p^\ell$. We say that the $\mathsf{D}_{\ell,k}$-Matrix Diffie-Hellman Assumption ($\mathsf{D}_{\ell,k}$-MDDH) holds in group $\mathbb{G}$, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl *such that: $\mathsf{Adv}^{MDDH}_{\mathsf{D}_{\ell,k},\mathcal{A}}(\kappa) \leq \mathrm{negl}(\kappa)$.*

2.1 Functional Encryption

A functional encryption scheme is formally defined as follows.

Definition 3 (Functional Encryption Scheme). *A FE scheme for a functionality $f : \mathcal{M} \times \mathcal{F} \to \mathcal{Z}$ parameterized by $\rho := (\mathcal{M}, \mathcal{F}, \mathcal{Z})$, is defined by four following algorithms.*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$*: where* Setup *receives the security parameter κ, and returns a pair of master public/secret key. The public-key implicitly defines the functionality-parameter ρ.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, M)$*: where* Enc *receives the master public-key* mpk *and a message $M \in \mathcal{M}$, and it returns a ciphertext* ct.
- $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{msk}, F)$*: where* KeyGen *receives the master secret-key* msk *and function F, then it returns a functional-key sk_F.*
- $Y := \mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_F)$*: it receives a ciphertext* ct *and a functional-key sk_F, and returns $\perp$ or a value in the range of f.*

If in this definition Enc receives msk, instead of mpk, we say the resulting FE scheme is a private-key FE scheme.

Correctness. For a correct execution of the above encryption system, $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_F)$ would return $f_F(M)$ where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, M)$, $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{msk}, F)$ and $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\kappa)$.

Here we extend the syntax of FE to a decentralized version which seems in public-key setting from the client's point of view. Comparing with the standard definition of FE, we have added new parties called *authorities* where the encryption and decryption phase are protocols among clients and authorities. Clients are the owner of data and authorities are in charge of some computation holding their secret-keys.

Definition 4 (Decentralized Functional Encryption (dFE)). *A FE scheme for a functionality $f : \mathcal{M} \times \mathcal{F} \to \mathcal{Z}$ parameterized by $\rho := (\mathcal{M}, \mathcal{F}, \mathcal{Z})$, is defined by four following algorithms.*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^k)$*: where* Setup *receives the security parameter κ, and returns the public parameters* mpk *and a set of secret keys* ek_i *as* $\mathsf{msk} = \{\mathsf{ek}_i\}_{i \in k}$*. The public-key implicitly defines the functionality-parameter ρ.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{msk}, M)$*: is a protocol among a clint and the set of authorities. They all receive the master public-key* mpk*. The authority i receives the secret key* ek_i *and the client receives the message $M \in \mathcal{M}$. They communicate and return the ciphertext* ct*.*
- $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, F)$*: is a protocol among the owner of function F and the set of authorities. They communicate and return a functional-key* sk_F*.*
- $Y := \mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_F)$*: it receives a ciphertext* ct *and a functional-key* sk_F*, and returns $\perp$ or a value in the range of f.*

IPFE and QFE. In this paper, we mainly distinct two special FE schemes. For inner-product FE (IPFE), the ciphertext and functional-key are respectively associated with vectors $M = \mathbf{x}$ and $F = \mathbf{y}$ of the same dimension. And a correct decryption returns $f_F(M) = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i \in [n]} x_i y_i$. For a quadratic-FE (QFE) scheme, the ciphertext and functional-key are respectively associated with vector-pairs $M = (\mathbf{x}, \mathbf{y}) \in \mathcal{M}^n \times \mathcal{M}^m$ and matrix $\boldsymbol{F} = [f_{ij}] \in \mathcal{F}^{n \times m}$, while a correct decryption outputs $f_F(M) = \mathbf{x}^T \boldsymbol{F} \mathbf{y} = \sum_{i,j} f_{ij} x_i y_j$.

Function-Hiding FE (FHFE). Informally, the security of FE says that no information about M should be leaked beyond $f_F(M)$. While in FH-security, the confidentiality of function F should be preserved as well. Note that in the public-key setting, getting the function-hiding property is not possible. Having access to the public-key, the adversary can encrypt its chosen messages and execute correct decryptions on these ciphertexts and a given functional-key. Which then obtains a system of equations with enough number of equations to find the unknown (here the unknown is the function F). Here we formally present the function-hiding (FH) property.

Definition 5 (FH-Security of FE). *For a functional encryption scheme* FE*, a PPT adversary $\mathcal{A}$ and a bit $b \leftarrow \{0, 1\}$, we define the game $\mathsf{IND}^b_{\mathsf{FE},\mathcal{A}}(\kappa)$ as shown in Fig. 2. Where the oracle* LREnc *on input (M^0, M^1) outputs $\mathsf{Enc}(\mathsf{msk}, M^b)$, the oracle* LRKey *on input (F^0, F^1) outputs $\mathsf{KeyGen}(\mathsf{msk}, F^b)$.*

The condition (*) is that for any message-challenge (M^0, M^1) sent to the oracle LREnc *and for any function-challenge (F^0, F^1) sent to the oracle* LRKey*,*

$$f_{F^0}(M^0) = f_{F^1}(M^1)$$

We say that a FE scheme is FH-secure if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that,*

$$\mathsf{Adv}^{\mathsf{IND}^b_{\mathsf{FE}}}_{\mathcal{A}}(\kappa) = |\Pr[\mathsf{IND}^b_{\mathsf{FE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{IND}^b_{\mathsf{FE},\mathcal{A}}(\kappa) = 0]| \leq \mathrm{negl}(\kappa)$$

Moreover, it is selectively FH-secure, if all the calls to the oracles LREnc *and* LRKey *has been done all together and before seeing the public-key.*
The weak FH-security (wFH-security) is referred to the case that the constraints on challenges are replaced with $f_{F^0}(M^0) = f_{F^0}(M^1) = f_{F^1}(M^1)$.

Definition 6 (dFH Security). *The dFH security concerns the function-hiding property of a dFE scheme and is defined similar to the FH-security where the adversary can issue the following corruption queries:*
$\mathsf{QCor}(A, i)$*: the response is a secret-key share* ek_i *associated with authority* i*.*
And the other queries are modified as follows.
LREnc *on input* (M^0, M^1) *outputs* $\mathsf{Enc}(\mathsf{mpk}, \mathsf{msk}, M^b)$ *and also a transcription of the associated communications among the client and the authorities.*
LRKey *on input* (F^0, F^1) *outputs* $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, F^b)$ *and also a transcription of the associated communications among the client (the owner of* F*) and the authorities.*

Security of FE (without Function-Hiding). In the above definition if the adversary is restricted to issue functional-keys $F^0 = F^1$, we just say FE scheme is (selectively) secure.

Multi-Client Functional Encryption. Here we present an generalized syntax of MCFE scheme. . Again, comparing with the standard definition of MCFE in [12], we have added new parties called *authorities* where the encryption phase is a protocol among clients and authorities.

Definition 7 (Multi-Client Functional Encryption). *Let* f *be a functionality (indexed by* ρ*), and* $\mathsf{Labels} = \{0,1\}^*$ *or* $\{\bot\}$ *be a set of labels. A multi-client functional encryption scheme (MCFE) for the functionality* f *and the label set* Labels *is a tuple of four algorithms* $\mathsf{MCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(1^\kappa, 1^n, 1^k)$**:** *Takes as input a security parameter* κ*, the number of clients* n*, and the number of authorities* k *(where for the standard definition* k *is zero), then generates public parameters* pp*. The public parameters implicitly define the functionality-index* ρ*. It outputs* n *secret-keys* $\{\mathsf{ek}_i\}_{i\in[n]}$*, the authorities' secret-key* $\mathsf{ask} = \{\mathsf{ek}'_i\}_{i\in[k]}$*, the master secret-key* $\mathsf{msk} = \{\mathsf{ek}_i\}_{i\in[n]}$ *and* pp*.*

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$**:** *Takes as input the public parameters* pp*, the master secret-key* msk *and a function* F*, and outputs a functional-key* sk_F*.*

$\mathsf{Enc}(\mathsf{pp}, \mathsf{ask}, \mathsf{ek}_i, m_i, \ell)$**:** *It is a protocol among authorities and the client* i*, where the client and each authority respectively receive the secret key* ek_i *and* ek'_i*. They all receive the public parameters* pp *and a label* $\ell \in \mathsf{Labels}$*. Moreover, the client receives the message* m_i *to encrypt. They communicate to output the ciphertext* $\mathsf{ct}_{i,\ell}$ *(where* $\mathsf{ct}_{i,\ell}$ *might be a set of ciphertexts indexed by* k*).*

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell})$**:** *Takes as input the public parameters* pp*, a functional-key* sk_F *and* n *ciphertexts under the same label* ℓ *and outputs* $\bot$ *or a value in range* f*.*

A scheme MCFE *is correct, if for all* $\kappa, n, k \in \mathbb{N}$*, functionality* f*,* $\ell \in \mathsf{Labels}$*, message* m_i*, when* $(\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in n}, \mathsf{ask}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^n, 1^k)$*,* $\mathsf{sk}_F \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$*,and* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ask}, \mathsf{ek}_i, m_i, \ell)$ *we have*

$$\Pr[\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_F, \{\mathsf{ct}_{i,\ell}\}_{i\in[n]}) = f_F(m_1, \ldots, m_n)] = 1.$$

$\underline{\mathsf{IND}^b_{\mathsf{FE},\mathcal{A}}(\kappa)}$:

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$

$\alpha \leftarrow \mathcal{A}^{\mathsf{LREnc}(\cdot,\cdot),\mathsf{LRKey}(\cdot,\cdot)}(\mathsf{mpk})$

Output α if condition (*) is satisfied

otherwise output a random bit β.

Fig. 2: Game for adaptive FH-security

Clients-Categorizing. To be clear, the correct decryption for a MCQFE scheme, returns $\mathbf{x}^T F \mathbf{y}$ where $\mathbf{x}^T = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_m)$ and a client may own any possible data x_i or y_j. For the sake of generality, we define two sides of clients: x-side and y-side clients. Where x_i and y_j are respectively assigned to the i-th x-side client and j-th y-side client.

Security Notion. As we noticed in a MCFE scheme each slot i has a different secret key ek_i, which can be individually corrupted. Comparing with the standard security-notion of MCFE [12], in our definition authorities may also be corrupted, while the encryption-quires are only issued w.r.t the clients (since they are the data owners and authorities just help them during the computations/encryption). In the following, we formally define the security notion of a MCFE scheme. Here $\mathsf{id} = \{C, A\}$ where C stands for the clients and A stands for authorities.

Definition 8 (Security of MCFE). *Let* MCFE *be an MCFE scheme and* Labels *a label set. For $\beta \in \{0,1\}$, we define the experiment $IND_\beta^{\mathsf{MCFE}}$ in Fig. 3, where the oracles are defined as:*

Corruption oracle $\mathsf{QCor}(\mathsf{id}, i)$**:** *Outputs the encryption key ek_i of slot i, if $\mathsf{id} = C$, otherwise outputs ek'_i. We denote by $\mathcal{CS}$ the set of corrupted clients at the end of the experiment.*

Left-Right oracle $\mathsf{QLeftRight}(i, m_i^0, m_i^1, \ell)$**:** *On a query (i, m_i^0, m_i^1, ℓ), outputs $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{ask}, \mathsf{ek}_i, m_i^\beta, \ell)$ and a transcription of associated communications among the client i and the authority set.*

Encryption oracle $\mathsf{QEnc}(i, m_i, \ell)$**:** *On a query (i, m_i, ℓ), outputs $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{ask}, \mathsf{ek}_i, m_i, \ell)$ and a transcription of communications among the client i and the authority set.*

Key derivation oracle $\mathsf{QKeyGen}(F)$**:** *Outputs $\mathsf{sk}_F = \mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, F)$.*

and where Condition (*) *holds if all the following conditions hold:*

- *If $i \in \mathcal{CS}$: for any query $\mathsf{QLeftRight}(i, m_i^0, m_i^1, \ell)$, $m_i^0 = m_i^1$.*
- *For any label $\ell \in \mathsf{Labels}$, for any family of queries $\{\mathsf{QLeftRight}(i, m_i^0, m_i^1, \ell)$ or $\mathsf{QEnc}(i, m_i, \ell)\}_{i \in [n] \setminus \mathcal{CS}}$, for any family of inputs $\{m_i \in \mathcal{X}\}_{i \in \mathcal{CS}}$, for any query $\mathsf{QKeyGen}(F)$, we define $m_i^0 = m_i^1 = m_i$ for any slot $i \in \mathcal{CS}$ and any slot queried to $\mathsf{QEnc}(i, m_i, \ell)$, we require that: $f(\boldsymbol{m}^0) = f(\boldsymbol{m}^1)$ where $\boldsymbol{m}^{\boldsymbol{b}} = (m_1^b, \ldots, m_n^b)$ for $b \in \{0,1\}$.*
  We insist that if one index $i \notin \mathcal{CS}$ is not queried for the label ℓ, there is no restriction.

The weaker versions of the security are defined as xx-yy-$IND_\beta^{\mathsf{MCFE}}$ *(*xx, yy *are empty where we don't have their corresponding following restrictions), where,*

- *When* xx = one*: for any slot $i \in [n]$ and $\ell \in \mathsf{Labels}$, the adversary is limited to exactly one encryption/challenge query on each (i, ℓ).*
- *When* yy = sel*: the adversary should output the challenges at the beginning of the game, and it does not have access to the oracle* QLeftRight *after that. This case is referred as the selective security.*

We define the advantage of an adversary $\mathcal{A}$ in the following way:

$$\mathsf{Adv}^{\text{xx-yy-IND}}_{\mathsf{MCFE},\mathcal{A}}(\kappa, n, k) = \big| \Pr[\text{xx-yy-IND}_0^{\mathsf{MCFE}}(\kappa, n, k, \mathcal{A}) = 1] - \Pr[\text{xx-yy-IND}_1^{\mathsf{MCFE}}(\kappa, n, k, \mathcal{A}) = 1] \big|.$$

A multi-client functional encryption scheme MCFE *is* xx-yy*–IND secure, if for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\text{xx-yy-}IND}_{\mathsf{MCFE},\mathcal{A}}(\kappa, n, k) \leq \mathrm{negl}(\kappa)$. *In this paper we mainly work with* one-sel-IND *security.*

$\mathbf{IND}_{\beta}^{\mathsf{MCFE}}(\kappa, n, k, \mathcal{A})$

$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\kappa}, 1^{n}, 1^{k})$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QCor}(\cdot),\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QKeyD}(\cdot)}(\mathsf{pp})$

Output: α if Condition (*) is satisfied,
or a uniform bit otherwise

Fig. 3: Security games for MCFE

2.2 A Review on the FHIP Scheme of Lin[26]

The FHIP scheme of Lin [26] (Lin-FHIP) plays a very important role in all our constructions. She presented an elegant FHIP scheme from IPFE scheme in a double-layer way and based on SXDH assumption. Informally, for the encryption one encrypts the message by the Enc algorithm of the inner-layer IPFE and then applies the KeyGen algorithm of the outer-layer IPFE (see Fig. 6). Similarly, to generate the functional-key, one needs to put an outer-layer Enc over an inner-layer KeyGen. Here we recap the FHIP construction of Lin [26] in Fig. 5. Let $\mathcal{G} = (q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be the description of a bilinear map where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The IPFE scheme $\mathsf{IP}_i = (\mathsf{Setup}_i, \mathsf{Enc}_i, \mathsf{KeyGen}_i, \mathsf{Dec}_i)$ for $i = 1, 2$ is instantiated based on DDH assumption and the encryption and decryption algorithms work in $\mathbb{G}_i$ space. More precisely, the underlying IPFE scheme IP_i is instantiated with the scheme of Abdalla et al. [3] as follows, which we call it as ABDP-IPFE:

$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Setup}(1^{\kappa}, 1^{n_i})$,	$\mathsf{Enc}_i(\mathsf{pk}_i, \mathbf{x}) = [r, \mathbf{x} + r\mathsf{sk}_i]_i = [\mathsf{ct}]_i$
$\mathsf{KeyGen}_i(\mathsf{sk}_i, \mathbf{y}) = (-\langle \mathbf{y}, \mathsf{sk}_i \rangle, \mathbf{y}) = \mathsf{sk}_y$	$\mathsf{Dec}_i([\mathsf{ct}]_i, \mathsf{sk}_y) = \mathsf{DLog}[\langle \mathsf{ct}, \mathsf{sk}_y \rangle]_i$

Fig. 4: ABDP-IPFE [3]

Here we emphasize on properties of their scheme which we widely use in our constructions.

Property 1. In ABDP-IPFE scheme,
i. As long as we don't need to compute DLog, the message can belong to the set $\mathbb{Z}_q$ or $\mathbb{G}_i$. On the other hand, the function may belong to the set $\mathbb{Z}_q$ or $\mathbb{G}_j$, $j \neq i$.
ii. Under the same randomness, it is key and message homomorphic. Namely, $\mathsf{Enc}(\mathsf{pk}_1, \mathbf{x}_1; r) + \mathsf{Enc}(\mathsf{pk}_2, \mathbf{x}_2; r) = \mathsf{Enc}(\mathsf{pk}_1 + \mathsf{pk}_2, \mathbf{x}_1 + \mathbf{x}_2; r)$.

While Fig. 5 presents a more detailed version of Lin-FHIP scheme, we mainly use the compact form of Lin-FHIP presented in Fig. 6.

Theorem 1 (Lin [26]). *If IP_i is selectively secure for $i = 1, 2$, then FHIP scheme of Lin is selectively-wFH-secure.*

Lin and Vaikuntanathan [27] showed that any FHIP scheme with weak function-hiding (wFH) security can be generically "lifted" to a FHIP scheme. Where the vector $\mathbf{x}$ is encrypted as $\mathbf{x}||\mathbf{0}$ (similarly $\mathbf{y}$ as $\mathbf{y}||\mathbf{0}$) by a wFH-secure IP scheme where $\mathbf{0}$ is a zero-vector of dimension $n = |\mathbf{x}|$.

$\underline{\mathsf{Setup}(1^\kappa)}$:
- run $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Setup}_1(1^\kappa, 1^n)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{Setup}_2(1^\kappa, 1^{n+1})$.

Return $\mathsf{msk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and $\mathsf{pp} = (\mathsf{pk}_1, \mathsf{pk}_2)$.

$\underline{\mathsf{Enc}(\mathsf{msk}, \mathbf{x})}$:
- run $\mathsf{ct} \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x})$ and $\mathsf{CT} \leftarrow \mathsf{KeyGen}_2(\mathsf{sk}_2, \mathsf{ct})$. Return CT.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})}$:
- run $\mathsf{sk} \leftarrow \mathsf{KeyGen}_1(\mathsf{sk}_1, \mathbf{y})$ and $\mathsf{SK} \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2, \mathsf{sk})$. Return SK.

$\underline{\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})}$:

Return $\mathsf{Dec}_2(\mathsf{SK}, \mathsf{CT})$

Fig. 5: Lin-FHIP [26]

$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Setup}_1(1^\kappa, 1^n), \quad (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{Setup}_2(1^\kappa, 1^{n+1})$
$\mathsf{CT} = \mathsf{KeyGen}_2(\mathsf{Enc}_1(\mathbf{x})), \quad \mathsf{SK} = \mathsf{Enc}_2(\mathsf{KeyGen}_1(\mathbf{y}))$
$\mathbf{x} \in \mathbb{Z}_q \vee \mathbb{G}_1, \mathbf{y} \in \mathbb{Z}_q \vee \mathbb{G}_2$, as long as $\mathsf{sk}_i \in \mathbb{Z}_q$, for $i = 1, 2$.

Fig. 6: Compact form of Lin-FHIP

Property 2. By applying the Lin's construction over the scheme of ABDP-IPFE one can verify the following properties.
i. The ciphertext is in the form of,

$$\mathsf{CT} = (-\langle \mathsf{sk}_2, \mathsf{ct}_x \rangle, \mathsf{ct}_x) \quad \text{where} \quad \mathsf{ct}_x = \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x})$$

and the functional-key is as,

$$\begin{aligned} \mathsf{SK} &= \mathsf{Enc}_2(\mathsf{pk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; r) || \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r) \\ &= \mathsf{Enc}_2(\mathsf{sk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; [r]_2) || \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r) \end{aligned}$$

ii. Applying the outer-layer does not change the output of the inner-layer and it is just to preserve the security of $\mathbf{y}$ (while the inner-layer preserves the security of $\mathbf{x}$). Namely, by applying $\mathsf{Enc}_1(\mathbf{x})$ over $\mathsf{KeyGen}_1(\mathbf{y})$, the output of the inner-layer is $\langle \mathbf{x}, \mathbf{y} \rangle$. While the output of the outer-layer, by applying CT over SK is also $\langle \mathbf{x}, \mathbf{y} \rangle$.

3 A 2-Input QFE Scheme

Here we present a (2-input) QFE scheme which is also the sketch of our MCQFE scheme (see Fig. 7), assisting us to explain the requirements for the multi-client setting. In this construction $\mathsf{FH} = (\mathsf{FH.Setup}, \mathsf{FH.Enc}, \mathsf{FH.KeyGen}, \mathsf{FH.Dec})$ is Lin-FHIP scheme where $\mathsf{ct}_{x_i} \in \mathbb{G}_1$ and $\mathsf{ct}_{y_j} \in \mathbb{G}_2$. The decryption algorithm $\mathsf{FH.Dec}$ in similar to the decryption algorithm of Lin-FHIP, except that, the DLog-computation is ignored. Note that our general construction is using a FHIP scheme in the encryption stage which means the construction gives a private-key QFE.

Correctness:

$$A_{ij} = [f_{i,j} \cdot \langle x_i || \mathbf{0} || \boldsymbol{\alpha} U_i || U_i', y_j || \mathbf{0} || T_j || T_j' \boldsymbol{\beta} \rangle]_T = [f_{i,j} x_i y_j + \boldsymbol{\alpha} f_{i,j} U_i T_j + f_{i,j} U_i' T_j' \boldsymbol{\beta}]_T$$
$$B = [-\boldsymbol{\alpha} \Sigma f_{i,j} U_i T_j - \Sigma f_{i,j} U_i' T_j' \boldsymbol{\beta}]_T \qquad C = \prod A_{i,j} \cdot B = [\sum f_{i,j} x_i y_j]_T$$

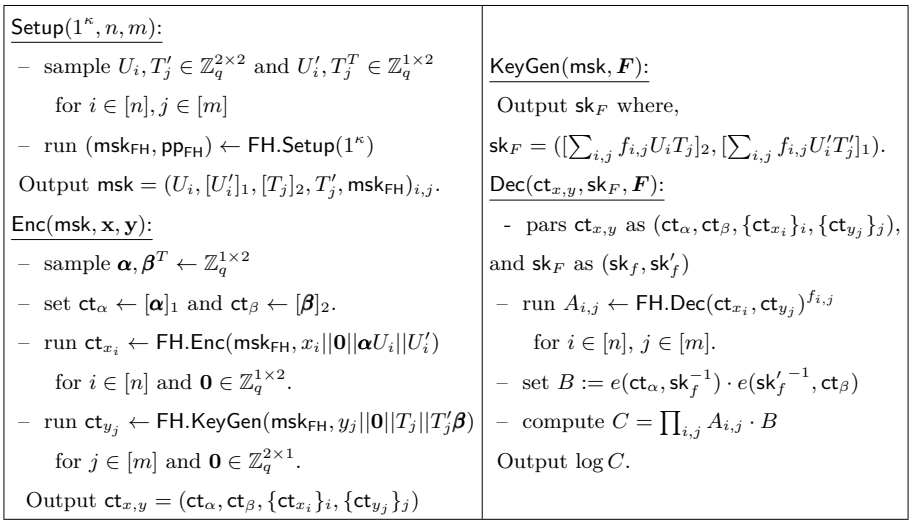

Fig. 7: Our (2-input) QFE scheme (the sketch of our MCQFE scheme)

This construction can not handle corruption or labels and it just supports two clients: one holds the whole vector $\mathbf{x}$ and the other holds $\mathbf{y}$ [5]. Later we extend this construction to a multi-client setting, thus here we ignore the security-proof.

To extend our (2-input) QFE to the general multi-client setting, each ciphertext ct_{x_i} or ct_{y_j} (associated with $\mathsf{ct}_{x,y}$) should be assigned to a separate client. As it was mentioned, we distinct the clients by two sides; x-side and y-side. But this categorization is just to explain the scheme and does not make any limitation on the functionality, as one client can have data on both sides or we can combine several clients in one client.

In a multi-client setting, the client i on the x-side (similarly, the client j on the y-side) should be able to compute ct_{x_i} (similarly ct_{y_j}) only by its own secret-key. Therefore specifically for our QFE, the client i on x-side should have access to U_i, $\boldsymbol{\alpha}$ and $\mathsf{msk_{FH}}$. Since U_i is not appearing in other parts we can consider it as the secret-key for the client i. On the other hand, $\boldsymbol{\alpha}$ has to be the same for all the clients on the x-side, but since the only term involved with $\boldsymbol{\alpha}$ is $[\boldsymbol{\alpha}U_i]_1$, and $U_i \in \mathbb{Z}_q$ is the secret-key of the i-th client, it is possible to replace $[\boldsymbol{\alpha}]_1$ with a random oracle such that all the clients can share the same randomness (i.e., $\mathcal{H}(\ell) = [\boldsymbol{\alpha}]_1$, which is a standard technique to share the randomness [9, 12])

Regarding the secret-key $\mathsf{msk_{FH}}$, the same technique that we used for sharing $\boldsymbol{\alpha}$ can not work here, as in the FHIP scheme we need $\mathsf{msk_{FH}} \in \mathbb{Z}_p$ (and not in $\mathbb{G}_i$).

Another point is about mix-and-match property among different values of ct_{x_i} and ct_{y_j}. This property simply says that FHIP decryption over $(\mathsf{ct}_{x_i}, \mathsf{ct}_{y_j})$ should be possible for any $i \in [n]$ and $j \in [m]$ (over the same label). This is an essential requirement for our QFE scheme and any modification of FHIP scheme should preserve this property.

4 Public-Key FHIP Scheme in Decentralized Setting

To go around the mentioned challenges regarding the master secret-key $\mathsf{msk_{FH}}$, we present a FHIP scheme such that it seems in public-key setting, from clients' point of view, while it still preserves mix-and-match property.

[5] thus, we call it 2-input rather than 2-client.

4.1 Single-Authority FHIP (warm-up)

We introduce a new party called authority to the underlying FHIP scheme. Introducing this authority will somehow change the FHIP from private-key setting to the public-key setting (w.r.t the client-view). While the authority is in charge of some computations holding the secret-key, the client, as the data owner, has access only to the public-key. Fig. 8 depicts our single-authority FHIP scheme. Here $IP_i = (\mathsf{Setup}_i, \mathsf{Enc}_i, \mathsf{KeyGen}_i, \mathsf{Dec}_i)$ for $i = 1, 2$ and $IP'_2 = (\mathsf{Setup}'_2, \mathsf{Enc}'_2, \mathsf{KeyGen}'_2, \mathsf{Dec}'_2)$ are ABDP-IPFE scheme respectively in groups $\mathbb{G}_i$ and $\mathbb{G}_2$. As one sees the difference with a usual FHIP scheme is that in an authority-based IPFE, the Enc and KeyGen algorithms are *protocols* between a client (data-owner) and an authority (secret-key owner).

The underlying idea is as follows. Basically, the goal is to generate the same ciphertext and functional-key of Lin-FHIP without revealing $\mathbf{x}$ or $\mathbf{y}$ to the authorities. By Property2 the ciphertext of Lin-FHIP can be written as,

$$(-\langle \mathsf{sk}_2, \mathsf{ct}_x \rangle, \mathsf{ct}_x) \quad \text{where} \quad \mathsf{ct}_x = \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x}) \tag{1}$$

and the functional-key is as:

$$\mathsf{Enc}_2(\mathsf{pk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; r) || \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r) = \mathsf{Enc}_2(\mathsf{sk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; [r]_2) || \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r) \tag{2}$$

Now for the encryption, the client just needs to publish ct_x. The authority holding sk_2 can compute the target ciphertext (Eq. (1)).

For the functional-key generation, the client publishes $\mathsf{ct}_v \leftarrow \mathsf{Enc}'_2(\mathsf{pk}_v, \mathbf{y})$ and $(\mathsf{ct}_{0y}, \mathsf{ct}_{1y}) \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r)$ where the former is used to generate the first part of the functional-key and the latter for the second part. The authority holding $\mathsf{sk}'_v \leftarrow \mathsf{KeyGen}'_2(\mathsf{sk}_v, \mathsf{sk}_1)$ as its secret-key and given ct_v, can now compute $[\langle \mathsf{sk}_1, \mathbf{y} \rangle]_2$. Finally, it uses $\mathsf{ct}_{0y} = [r]_2$ and its secret-key sk_2^0 to encrypt $[\langle \mathsf{sk}_1, \mathbf{y} \rangle]_2$ under the same randomness that the client has used. Therefore, it can efficiently compute the functional-key (Eq. (2)).

The reason that we need IP'_2 to work in the group $\mathbb{G}_2$, is that in our construction Dec'_2 is computed over some data that their inner-product value can be large and so the decryption would fail (since having small inner-product is a requirement for the ABDP-IPFE scheme [3]). Therefore, the idea is to use IP'_2 over $\mathbb{G}_2$ and Dec'_2 is the decryption algorithm of ABDP-IPFE scheme without discrete-logarithm computation. This means the output of the decryption is in group $\mathbb{G}_2$ which is also compatible with what we need in $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{ek}, \mathbf{y})$.

Correctness. By the correctness of Lin-FHIP (Fig. 5), it is enough to show that $\mathsf{CT} = \mathsf{KeyGen}_2(\mathsf{Enc}_1(\mathbf{x}))$ and $\mathsf{SK} = \mathsf{Enc}_2(\mathsf{KeyGen}_1(\mathbf{y}))$. For CT, the relation is clearly true. For SK, we have $\mathsf{sk}' = \mathsf{Enc}_2(\mathsf{sk}_2^0, -\mathsf{sk}; \mathsf{ct}_{0y}) = \mathsf{Enc}_2(\mathsf{pk}_2^0, \mathsf{sk}; r)$. Thus,

$$\mathsf{SK} = (\mathsf{Enc}_2(\mathsf{pk}_2^0, -\mathsf{sk}; r), \mathsf{ct}_{1y}) = \mathsf{Enc}_2(\mathsf{pk}_2, -\langle \mathsf{sk}_1, \mathbf{y} \rangle || \mathbf{y}; r) = \mathsf{Enc}_2(\mathsf{KeyGen}_1(\mathbf{y}))$$

Security Properties. One can verify that if the adversary does not have access to ek (i.e., $\mathsf{Authority}$ is not corrupted), our single-authority FHIP inherits the FH-security of Lin-FHIP.

4.2 Decentralized FHIP (d-FHIP)

To increase the security, we are interested to relax the condition "uncorrupted authority" by adding more authorities; such that if at least one of the authorities is uncorrupted (while it

$\underline{\mathsf{Setup}(1^\kappa)}$:
- run $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Setup}_1(1^\kappa, 1^n)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{Setup}_2(1^\kappa, 1^{n+1})$.
- parse $\mathsf{sk}_2, \mathsf{pk}_2$ as $\mathsf{sk}_2^0 || \mathsf{sk}_2^1 \in \mathbb{Z}_q \times \mathbb{Z}_q^n$ and $\mathsf{pk}_2^0 || \mathsf{pk}_2^1 \in \mathbb{G}_2 \times \mathbb{G}_2^n$
- run $(\mathsf{sk}_v, \mathsf{pk}_v) \leftarrow \mathsf{Setup}'_2(1^\kappa, 1^n)$ and $\mathsf{sk}'_v \leftarrow \mathsf{KeyGen}'_2(\mathsf{sk}_v, \mathsf{sk}_1)$

Return $\mathsf{ek} = \mathsf{sk}_2 || \mathsf{sk}'_v$ as the secret-key for the Authority, and $\mathsf{pp} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_v)$.

$\underline{\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}, \mathbf{x} \in \mathbb{Z}_q^n)}$:
- $\mathsf{Client}(\mathsf{pp}, \mathbf{x})$: sends $\mathsf{ct}_x \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x})$ to Authority.
- $\mathsf{Authority}(\mathsf{sk}_2, \mathsf{ct}_x)$: returns CT where $\mathsf{CT} \leftarrow \mathsf{KeyGen}_2(\mathsf{sk}_2, \mathsf{ct}_x)$

Return CT

$\underline{\mathsf{KeyGen}(\mathsf{pp}, \mathsf{ek}, \mathbf{y} \in \mathbb{Z}_q^n)}$:
- $\mathsf{Client}(\mathsf{pp}, \mathbf{y})$: sends $\mathsf{ct}_v \leftarrow \mathsf{Enc}'_2(\mathsf{pk}_v, \mathbf{y})$ and ct_y to Authority where $\mathsf{ct}_y = (\mathsf{ct}_{0y}, \mathsf{ct}_{1y}) \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r)$ and $\mathsf{ct}_{0y} = [r]_2$.
- $\mathsf{Authority}(\mathsf{sk}'_v, \mathsf{ct}_y)$: computes $\mathsf{sk} = \mathsf{Dec}'_2(\mathsf{sk}'_v, \mathsf{ct}_v) = [\langle \mathsf{sk}_1, \mathbf{y} \rangle]_2$ and then computes $\mathsf{sk}' \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2^0, -\mathsf{sk}; \mathsf{ct}_{0y})$.

Return $\mathsf{SK} = (\mathsf{ct}_{0y}, \mathsf{sk}', \mathsf{ct}_{1y})$

$\underline{\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})}$: Output $\mathsf{Dec}_2(\mathsf{SK}, \mathsf{CT})$

Fig. 8: Single-authority FHIP

is still curious), the FH-security notion is still satisfied. Note that in our single-authority FHIP scheme the secret-key of Authority is fixed for all the ciphertexts. Thus, one can use a secret-sharing protocol to share the key among n authorities, once in the setup phase. Then each authority should compute its share of the inner-product value holding only its secret-key share. Putting all the inner-product shares together will generate the desired final inner-product value. The underlying idea is similar to our single-authority construction while the ciphertext and the functional-key can be computed in a decentralized way, thanks to the linearity of inner-product and homomorphic property of ABDP-IPFE scheme [3] (Property 1). The resulting construction is depicted in Fig. 9 where n (maximum number of authorities) equals the length of the message-vector (independent of the number of clients).

Correctness. By the correctness of Lin-FHIP (Fig. 5), It is enough to show that $\mathsf{CT}' = \mathsf{KeyGen}_2(\mathsf{Enc}_1(\mathbf{x}))$ and $\mathsf{SK}' = \mathsf{Enc}_2(\mathsf{KeyGen}_1(\mathbf{y}))$
For CT':

$$\mathsf{CT}' = (\prod_i \mathsf{ct}_{i0}, \mathsf{ct}_x) = (-\prod_i \langle \mathsf{sk}_{2,i}, \mathsf{ct}_x \rangle, \mathsf{ct}_x) = (\prod_i \mathsf{ct}_x^{\mathsf{sk}_{2,i}}, \mathsf{ct}_x)$$
$$= (-\langle \sum_i \mathsf{sk}_{2,i}, \mathsf{ct}_x \rangle, \mathsf{ct}_x) = (-\langle \mathsf{sk}_2, \mathsf{ct}_x \rangle, \mathsf{ct}_x) = \mathsf{KeyGen}_2(\mathsf{Enc}_1(\mathbf{x}))$$

For SK': we know that $\sum_i \mathsf{sk}_i = \sum_i \langle \mathsf{sk}_{1,i}, \mathbf{y} \rangle = \langle \mathsf{sk}_1, \mathbf{y} \rangle$ and $\mathsf{sk}'_i = \mathsf{Enc}_2(\mathsf{sk}_{2,i}^0, -\mathsf{sk}_i; \mathsf{ct}_{0y}) = \mathsf{Enc}_2(\mathsf{pk}_{2,i}^0, \mathsf{sk}_i; r)$ therefore,

$$\prod_i \mathsf{sk}'_i = \prod_i \mathsf{Enc}_2(\mathsf{pk}_{2,i}^0, -\mathsf{sk}_i; r) = \mathsf{Enc}(\mathsf{pk}_2^0, -\sum_i \mathsf{sk}_i; r) = \mathsf{Enc}(\mathsf{pk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; r)$$

where the second equality is due to the homomorphic property of underlying IP scheme. We also have, $(\mathsf{ct}_{0y}, \mathsf{ct}_{1y}) = \mathsf{Enc}(\mathsf{pk}_2^1, \mathbf{y}; r)$. Thus,

$$\mathsf{SK}' = (\mathsf{ct}_{0y}, \prod_i \mathsf{sk}'_i, \mathsf{ct}_{1y}) = \mathsf{Enc}_2(\mathsf{pk}_2, -\langle \mathsf{sk}_1, \mathbf{y} \rangle || \mathbf{y}; r) = \mathsf{Enc}_2(\mathsf{KeyGen}_1(\mathbf{y}))$$

$\underline{\mathsf{Setup}(1^\kappa)}$:

- run $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Setup}_1(1^\kappa, 1^n)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{Setup}_2(1^\kappa, 1^{n+1})$.
- run $\{\mathsf{sk}_{1,i}\}_i \leftarrow \mathsf{SecretShare}(\mathsf{sk}_1)$ such that $\mathsf{sk}_1 = \sum_i \mathsf{sk}_{1,i}$.
- run $\{\mathsf{sk}_{2,i}\}_i \leftarrow \mathsf{SecretShare}(\mathsf{sk}_2)$ such that $\mathsf{sk}_2 = \sum_i \mathsf{sk}_{2,i} = \sum_i \mathsf{sk}^0_{2,i} || \mathsf{sk}^1_{2,i}$.
- parse $\mathsf{sk}_2, \mathsf{pk}_2$ as $\mathsf{sk}^0_2 || \mathsf{sk}^1_2 \in \mathbb{Z}_q \times \mathbb{Z}_q^n$ and $\mathsf{pk}^0_2 || \mathsf{pk}^1_2 \in \mathbb{G}_2 \times \mathbb{G}_2^n$
- run $(\mathsf{sk}_v, \mathsf{pp}_v) \leftarrow \mathsf{Setup}'_2(1^\kappa)$ and set $\mathsf{sk}'_{v,i} = \mathsf{KeyGen}'_2(\mathsf{sk}_v, \mathsf{sk}_{1,i})$.

Return $\mathsf{pp} = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_v)$, $\mathsf{ek}_i = \mathsf{sk}_{2,i} || \mathsf{sk}'_{v,i}$ as the secret-key for i-th authority for $i \in [n]$.

$\underline{\mathsf{Enc}(\mathsf{pp}, \{\mathsf{ek}_i\}_i, \mathbf{x})}$:

- $\mathsf{Client}(\mathsf{pp}, \mathbf{x})$: publishes ct_x where $\mathsf{ct}_x \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x}; r')$
- $\mathsf{Authority}_i(\mathsf{ek}_i, \mathsf{ct}_x)$: returns CT_i where $\mathsf{CT}_i \leftarrow \mathsf{KeyGen}_2(\mathsf{sk}_{2,i}, \mathsf{ct}_x) = (-\langle \mathsf{sk}_{2,i}, \mathsf{ct}_x \rangle, \mathsf{ct}_x)$

Return $\mathsf{CT} = \{\mathsf{CT}_i\}_i$.

$\underline{\mathsf{KeyGen}(\mathsf{pp}, \{\mathsf{ek}_i\}_i, \mathbf{y})}$:

- $\mathsf{Client}(\mathsf{pp}, \mathbf{y})$: publishes $\mathsf{ct}_v = \mathsf{Enc}'_2(\mathsf{pk}_v, \mathbf{y})$ and $\mathsf{ct}_y = (\mathsf{ct}_{0y}, \mathsf{ct}_{1y}) \leftarrow \mathsf{Enc}_2(\mathsf{pk}^1_2, \mathbf{y}; r)$, where $\mathsf{ct}_{0y} = [r]_2$.
- $\mathsf{Authority}_i(\mathsf{ek}_i, \mathsf{ct}_y)$: computes $\mathsf{sk}_i = \mathsf{Dec}'_2(\mathsf{sk}'_{v,i}, \mathsf{ct}_v) = [\langle \mathsf{sk}_{1,i}, \mathbf{y} \rangle]_2$ and $\mathsf{sk}'_i \leftarrow \mathsf{Enc}_2(\mathsf{sk}^0_{2,i}, -\mathsf{sk}_i; \mathsf{ct}_{0y})$ and returns $\mathsf{SK}_i = (\mathsf{ct}_{0y}, \mathsf{sk}'_i, \mathsf{ct}_{1y})$

Return $\mathsf{SK} = \{\mathsf{SK}_i\}_i$

$\underline{\mathsf{Dec}(\{\mathsf{CT}_i\}_i, \{\mathsf{SK}_i\}_i)}$:

Aggregation: $\begin{cases} \text{parse } \mathsf{SK}_i \text{ as } (\mathsf{ct}_{0y}, \mathsf{sk}'_i, \mathsf{ct}_{1y}) \text{ and } \mathsf{CT}_i \text{ as } (\mathsf{ct}_{0i}, \mathsf{ct}_x) \\ \text{set } \mathsf{SK}' = (\mathsf{ct}_{0y}, \prod_i \mathsf{sk}'_i, \mathsf{ct}_{1y}) \text{ and } \mathsf{CT}' = (\prod_i \mathsf{ct}_{0i}, \mathsf{ct}_x) \end{cases}$

Return $\mathsf{Dec}_2(\mathsf{SK}', \mathsf{CT}')$.

Fig. 9: our d-FHIP scheme

Remark 1. Note that in our d-FHIP, the maximum number of authorities is n. We emphasize that one can simply extend our construction to any arbitrary maximum number k of authorities by adding zeros to the message as $\mathbf{x}||\mathbf{0} \in \mathbb{Z}_q^k$ and random values to the function as $\mathbf{y}||\boldsymbol{r}^* \in \mathbb{Z}_q^k$. Such extension may have some advantage in MCQFE where the clients can play the role of authorities.

Another point is that, since all the operations for the aggregation are linear, our construction could work also with a threshold sharing of sk_1 and sk_2 instead of a n-out-of-n. Indeed, instead of $\mathsf{sk} = \sum sk_i$, we would have $\mathsf{sk} = \sum \lambda_i \mathsf{sk}_i$ where λ_i is the i-th Lagrange interpolation coefficient for the appropriate subset.

Security Analysis. Here we claim that as long as there exists at least one uncorrupted authority called i^*, our d-FHIP scheme is dFH-secure. Obviously, knowing shares ek_i for $i \neq i^*$, can not directly help the adversary to break the security since by the security of secret-sharing, it can not recover ek_{i^*}. Rather, it may use the inner-product share of i^* (available via encryption queries) to get information about $\mathbf{x}$ or $\mathbf{y}$. Intuitively, the parts which may leak information are mainly ct_x, ct_y and ct_v, sk'_{v,i^*}. The latter is due to the fact that leaking $\langle \mathsf{sk}_{1,i^*}, \mathbf{y} \rangle$ can totally reveal $\mathbf{y}$, since the adversary already has access to the values $\langle \mathsf{sk}_{1,i}, \mathbf{y} \rangle$ for $i \neq i^*$ via corruption queries. By the security of IP'_2, no information about $\mathbf{y}$ is leaked via ct_v as long as the condition $\langle \mathsf{sk}_{1,i}, \mathbf{y}^1 - \mathbf{y}^0 \rangle = 0$ is satisfied (e.g., by allowing maximum (n-1) independent vectors $\mathbf{y}^1 - \mathbf{y}^0$ and (n-1) authorities in the game). Moreover we discuss that in the adversary point of view, the information ct_x and ct_y respectively are equivalent with the ciphertext and functional-key of the Lin-FHIP scheme where the secret-keys associated with the inner-layer and outer-layer are respectively sk_{1,i^*} and sk_{2,i^*}.

From there, we can reduce the security to the FH-security of Lin-FHIP. The sequence of games is summarized in Fig. 10.

G_0	$\mathsf{sk}'_{v,i} \leftarrow \mathsf{KeyGen}_2(\mathsf{sk}_v, \mathsf{sk}_{1,i})$ $\mathsf{ct}_v \leftarrow \mathsf{Enc}_2(\mathsf{pk}_v, \mathbf{y}^0)$	$\mathsf{ct}_x \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x}^0)$ $\mathsf{ct}_y \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}^0; r)$	real game $b = 0$
G_1	$\mathsf{sk}'_{v,i} = \mathsf{KeyGen}_2(\mathsf{sk}_v, \mathsf{sk}_{1,i})$ $\mathsf{ct}_v \leftarrow \mathsf{Enc}_2(\mathsf{pk}_v, \boxed{\mathbf{y}^1})$	$\mathsf{ct}_x \leftarrow \mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x}^0; r')$ $\mathsf{ct}_y \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}^0; r)$	security of IP_v associated with sk_v
G_2	$\mathsf{sk}'_{v,i} = \mathsf{KeyGen}_2(\mathsf{sk}_v, \mathsf{sk}_{1,i})$ $\mathsf{ct}_v \leftarrow \mathsf{Enc}_2(\mathsf{pk}_v, \mathbf{y}^1)$	$\mathsf{ct}_x \leftarrow [r' \sum_{i \neq i^*} \mathsf{sk}_{1,i}]_1 \cdot \mathsf{Enc}_1([\mathsf{sk}_{1,i^*}]_1, \boxed{\mathbf{x}^1}; r')$ $\mathsf{ct}_y \leftarrow [r \sum_{i \neq i^*} \mathsf{sk}_{2,i}^1]_2 \cdot \mathsf{Enc}_2([\mathsf{sk}_{2,i^*}^1]_2, \boxed{\mathbf{y}^1}; r)$	FH-security associated with $(\mathsf{sk}_{1,i^*}, \mathsf{sk}_{2,i^*})$

Fig. 10: overview of games for our d-FHIP scheme

Theorem 2. *If Lin-FHIP scheme is (weakly/fully) FH-secure, Our d-FHIP scheme in Fig. 9 is (weakly/fully) selective dFH-secure, as long as there exists at least one honest authority and the condition* $\langle \mathsf{sk}_{1,i}, \mathbf{y}^1 - \mathbf{y}^0 \rangle = 0$ *is satisfied on all the indices* i *and key queries.*

Proof. We assume the i^*-th authority is honest, namely the simulation samples i^* from $[n]$ as the honest authority, at the very beginning of the game. If at some point the adversary corrupts it, the simulation aborts. The probability that i^* is the honest authority is non-negligible and thus the reduction works with non-negligible probability (resulting in breaking the SXDH assumption with non-negligible probability). We start with the real game when the chosen bit is $b = 0$, while the last game is the real game associated with $b = 1$. The adversary $\mathcal{A}$ is the attacker trying to distinguish two adjacent games. Note that based on the security notion, in the simulation of encryption, we have to simulate the communications among the client i and all the authorities, this means we also have to simulate ct_v (while ct_x, ct_y are already part of the output of the protocols).

$\boxed{\mathsf{G}_0}$: is the real game in dFH-security (Definition 6) when the chosen bit is $b = 0$.

$\boxed{\mathsf{G}_1}$: is similar to the game G_0, except that in ct_v, $\mathbf{y}^0$ is replaced with $\mathbf{y}^1$. We reduce the indistinguishability of G_0 and G_1 to the security of IPFE scheme IP'_2. Since all function-challenges are issued at the beginning of the game, the simulator can sample $\{\mathsf{sk}_{1,i}\}_i$ such that: $\langle \mathsf{sk}_{1,i}, \mathbf{y}^1 - \mathbf{y}^0 \rangle = 0$ for every index i and key queries, and then set $\sum_i \mathsf{sk}_{1,i} = \mathsf{sk}_1$.

The simulator simulates games G_0 or G_1 for $\mathcal{A}$ by running the real algorithms, except for the simulation of $\mathsf{sk}'_{v,i}$ and ct_v, where it sends functional-key queries $\mathsf{sk}_{1,i}$, and the challenges $(\mathbf{y}^0, \mathbf{y}^1)$ to the challenger of IPFE scheme IP'_2.

$\boxed{\mathsf{G}_2}$: is similar to the previous game, except that, in ct_x and ct_y, the values $\mathbf{x}^0$ and $\mathbf{y}^0$ are respectively replaced with $\mathbf{x}^1$ and $\mathbf{y}^1$.

The transition from G_1 to G_2 relies on the FH-security of Lin-FHIP scheme when it is associated with keys sk_{1,i^*} and sk_{2,i^*}. In this transition we are using the following facts which are thanks to the homomorphic properties of ABDP-IPFE scheme [3] (see Fig. 4).

$$\mathsf{Enc}_1(\mathsf{pk}_1, \mathbf{x}; r') = \mathsf{Enc}_1(\sum_i \mathsf{sk}_{1,i}, \mathbf{x}; r') = [r' \sum_{i \neq i^*} \mathsf{sk}_{1,i}]_1 \cdot \mathsf{Enc}_1([\mathsf{sk}_{1,i^*}]_1, \mathbf{x}; r')$$
$$\mathsf{Enc}_2(\mathsf{pk}_2^1, \mathbf{y}; r) = \mathsf{Enc}_2(\sum_i \mathsf{sk}_{2,i}^1, \mathbf{y}; r) = [r \sum_{i \neq i^*} \mathsf{sk}_{2,i}^1]_2 \cdot \mathsf{Enc}_2([\mathsf{sk}_{2,i^*}^1]_2, \mathbf{y}; r) \quad (3)$$

The simulator, simulates games G_1 or G_2 for $\mathcal{A}$ by running the real algorithms (on the concerned inputs), except for $\mathsf{CT}_{i^*} = (\mathsf{ct}_{0,i^*}, \mathsf{ct}_x)$, $\mathsf{SK}_{i^*} = (\mathsf{ct}_{0y}, \mathsf{sk}'_{i^*}, \mathsf{ct}_{1y})$:

- it samples all the keys except, sk_{1,i^*} and sk_{2,i^*}.
- simulation of CT_{i^*}: when $\mathcal{A}$ submits the challenge $(\mathbf{x}^0, \mathbf{x}^1)$, the simulator sends it to its challenger and receives $\mathsf{FH.Enc}((\mathsf{sk}_{1,i^*}, \mathsf{sk}_{2,i^*}), \mathbf{x}^b; r') = (\mathsf{ct}_{0i^*}, [r']_1, k)$ where FH stands for the Lin-FHIP scheme, and by the construction of FH:

$$\mathsf{Enc}_1(\mathsf{sk}_{1,i^*}, \mathbf{x}^b; r') = ([r']_1, k), \quad \mathsf{ct}_{0i^*} = -\langle \mathsf{sk}_{2,i^*}, ([r']_1, k) \rangle$$

  Then it sends $\mathsf{CT}_{i^*} = (\mathsf{ct}_{0i^*}, \mathsf{ct}_x)$ to $\mathcal{A}$ where $\mathsf{ct}_x = ([r']_1, [r' \sum_{i \neq i^*} \mathsf{sk}_{1,i}]_1 \cdot k))$.
- simulation of SK_{i^*}: when $\mathcal{A}$ issues the challenge $(\mathbf{y}^0, \mathbf{y}^1)$, the simulator sends it directly to its challenger to receive $\mathsf{FH.KeyGen}((\mathsf{sk}_{1,i^*}, \mathsf{sk}_{2,i^*}), \mathbf{y}^b; r) = ([r]_2, h', h'')$, where by the construction of FH:

$$\mathsf{Enc}_2(\mathsf{sk}^0_{2,i^*}, -\langle \mathsf{sk}_{1,i^*}, \mathbf{y}^b \rangle; r) = ([r]_2, h'), \quad \mathsf{Enc}_2(\mathsf{sk}^1_{2,i^*}, \mathbf{y}^b; r) = ([r]_2, h'')$$

  Then it simulates ct_y and sk'_{i^*} as $\mathsf{ct}_y = (\mathsf{ct}_{0y}, \mathsf{ct}_{1y}) = ([r]_2, [r \sum_{i \neq i^*} \mathsf{sk}^1_{2,i}]_2 \cdot h'')$ and $\mathsf{sk}'_{i^*} = h'$. And sends $\mathsf{SK}_{i^*} = (\mathsf{ct}_{0y}, \mathsf{sk}'_{i^*}, \mathsf{ct}_{1y})$ to $\mathcal{A}$.

This is now the real game for $b = 1$. □

Theorem 3. *If in our construction we replace* $\mathbf{x}$ *with* $\mathbf{x}||0$ *and* $\mathbf{y}$ *with* $\mathbf{y}||r$ *for a random value* r *(and extend the dimension of keys* $\mathsf{sk}_1, \mathsf{sk}^1_2$ *to* $n+1$*). Then, Theorem 2 holds without condition* $\langle \mathsf{sk}_{1,i}, \mathbf{y}^1 - \mathbf{y}^0 \rangle = 0$.

Proof. By this transformation we can find the keys $\mathsf{sk}_{1,i}$ such that $\langle \mathbf{y}^1||r^1 - \mathbf{y}^0||r^0, \mathsf{sk}_{1,i} \rangle = 0$. Because we have $n(n+1)$ keys $\mathsf{sk}_{1,i}$ as unknowns and n equations associated with maximum n linearly independent vectors $\mathbf{y}^1 - \mathbf{y}^0$, which totally gives n^2 equations. □

5 2-Step MCQFE

In this section we present a 2-MCQFE scheme based on the d-FHIP scheme. Basically, we consider the 2-input QFE scheme (Fig. 7) where FH is replaced with our d-FHIP scheme (Fig. 9) denoted as dFH. We extend such 2-input QFE scheme to a 2-MCQFE with $(n+m)$ clients where n and m are respectively the number of slots of vector $\mathbf{x}$ and $\mathbf{y}$.

Our 2-MCQFE scheme is presented in Fig. 11. Here $\mathcal{H}_\alpha : \mathsf{Labels} \to \mathbb{G}_1^{1\times 2}$ and $\mathcal{H}_\beta : \mathsf{Labels} \to \mathbb{G}_2^{2\times 1}$ are hash function modeled as random oracles. The secret-key ek_k is defined as $\mathsf{ek}_{k,x}$ when the concerning client is on the x-side and it is $\mathsf{ek}_{k,y}$ when the client is on the y-side. Note that by Remark 1 we consider the number of authorities to be an arbitrary integer $t = \mathrm{poly}(\kappa)$ (i.e., one needs to apply the changes from Remark 1). Finally, the decryption algorithm $\mathsf{dFH.Dec}$ is similar to the decryption algorithm of our d-FHIP scheme, except that, the DLog-computation is ignored.

Intuitively as the confidentiality of both messages x_i and y_j should be preserved, we need a kind of symmetric structure. Namely, all one needs to do to preserve the privacy of x_i, the same methodology should be done for y_j. This is possible by operating on groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where the pairing allows us to combine/mix the results. In fact, this is the underlying idea for the Lin-FHIP scheme which is so connected to the requirement for a QFE scheme. Here we additionally have functional-key sk_F and random oracles $\mathcal{H}_\alpha$ and $\mathcal{H}_\beta$ which we try

$\underline{\mathsf{Setup}(1^\kappa)}$:

– sample $U_i, T'_j \xleftarrow{R} \mathbb{Z}_q^{2\times 2}$ and $U'_i, T_j^T \xleftarrow{R} \mathbb{Z}_q^{1\times 2}$ for $i \in [n], j \in [m]$.

– run $(\{\mathsf{ek}_i\}_{i\in[t]}, \mathsf{pp}) \leftarrow \mathsf{dFH.Setup}(1^k)$

Return pp, $\mathsf{ask} = \{\mathsf{ek}_i\}_{i\in[t]}$, $\mathsf{msk} = (\mathsf{ek}_{i,x}, \mathsf{ek}_{j,y})_{i,j}$ and

$\mathsf{ek}_{i,x} = (U_i, [U'_i]_1)$, $\mathsf{ek}_{j,y} = ([T_j]_2, T'_j)$ as the secret keys respectively for the x-side and y-side clients.

$\underline{\mathsf{Enc}(\mathsf{pp}, \mathsf{ask}, \mathsf{ek}_k, m_k, \ell)}$:

– set $H_\alpha(\ell) = [\boldsymbol{\alpha}]_1 =\in \mathbb{G}_1^{1\times 2}, \mathcal{H}_\beta(\ell) = [\boldsymbol{\beta}]_2 \in \mathbb{G}_2^{2\times 1}$, and $\mathbf{0} = (0,0)$.

– if this is a x-side clients: run $\mathsf{ct}_{x_k,\ell} \leftarrow \mathsf{dFH.Enc}(\mathsf{pp}, \mathsf{ask}, m_k||\mathbf{0}||\boldsymbol{\alpha}U_k||U'_i)$

– if this is a y-side client: run $\mathsf{ct}_{y_k,\ell} \leftarrow \mathsf{dFH.KeyGen}(\mathsf{pp}, \mathsf{ask}, m_k||\mathbf{0}^T||T_k||T'_k\boldsymbol{\beta})$

Return $\mathsf{ct}_{x_k,\ell}$ or $\mathsf{ct}_{y_k,\ell}$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{F})}$:

Return $\mathsf{sk}_F = (\mathsf{sk}_f, \mathsf{sk}'_f) = ([\Sigma f_{i,j} U_i T_j]_2, [\Sigma f_{i,j} U'_i T'_j]_1)$.

$\underline{\mathsf{Dec}((\mathsf{ct}_{x_i,\ell}, \mathsf{ct}_{y_j,\ell})_{i,j}, \boldsymbol{F}, \mathsf{sk}_F)}$:

– run $A_{i,j} \leftarrow \mathsf{dFH.Dec}(\mathsf{ct}_{x_i,\ell}, \mathsf{ct}_{y_j,\ell})^{f_{i,j}}$ for $i \in [n]$, $j \in [m]$.

– set $B \leftarrow e(\mathcal{H}_\alpha(\ell), \mathsf{sk}_f^{-1}) \cdot e(\mathsf{sk}'^{-1}_f, \mathcal{H}_\beta(\ell))$ and $C = \prod_{i,j} A_{i,j} \cdot B$

Return $\log C$.

Fig. 11: our 2-step Multi-Client QFE (2-MCQFE) from decentralized FHIP

to keep this symmetric structure for them as well. More precisely, the output of $\mathcal{H}_\alpha$ is in $\mathbb{G}_1$ and its associated part of sk_F is sk_f which belongs to $\mathbb{G}_2$, similarly about $\mathcal{H}_\beta$ and sk'_f. The main challenges in a multi-client setting are the corruption-queries and the separate encryption queries on each slot where mix-and-match over them (possible thanks to the underlying dFH scheme) should not leak additional information. Since in our construction ciphertexts are indexed by labels, and indices i or j, one can expect that the sequence of games includes hybrids on ℓ,i,j. Slightly more in details, in the security proof, to show that one can (indistinguishably) change a message $x_i^{\ell 0}$ to $x_i^{\ell 0}$,

1. at first, a hybrid of games over labels ℓ is defined such that every time we change $x_i^{\gamma 0}$ to $x_i^{\gamma 1}$ for the current label $\ell = \gamma$ while for all other labels nothing would be changed;
2. this discussion over label γ also needs a hybrid on index i, where every time for a specific index i^* we change $x_{i^*}^{\gamma 0}$ to $x_{i^*}^{\gamma 1}$;
3. finally such change also needs a hybrid over index j to show that in hybrid j^* we can (loosely speaking) change $x_{i^*}^{\gamma 0} y_{j^*}^{\ell 0}$ to $x_{i^*}^{\gamma 1} y_{j^*}^{\ell 0}$ while for $j \neq j^*$ nothing would be changed.

For 1., by MDDH assumption we change the structure of the random-oracle $\mathcal{H}_1$ such that for every label, except γ, it has the same form. This allows to change the challenge bit for γ (by relying on dFH-security as well), without being interrupted by other labels $\ell \neq \gamma$. For 2., we change the secret-key U_{i^*}, which makes it possible to change the challenge bit for i^* (again by relying on dFH-security as well), without being interrupted by other index $i \neq i^*$. For 3., by MDDH assumption we change the structure of T_{j^*} such that for every $j \neq j^*$, it has the same form. This allows to change the challenge bit for combinations involved with j^*. We discuss details in the security proof.

Game	Description
G_0	$\mathsf{ct}_{x_i} \leftarrow \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^0 \| \mathbf{0} \| \boldsymbol{\alpha} U_i \| U_i')$ $\quad \mathsf{sk}_F = ([\sum f_{i,j} U_i T_j]_2,\ [\sum f_{i,j} U_i' T_j']_1)$ $\mathsf{ct}_{y_j} \leftarrow \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, y_j^0 \| \mathbf{0} \| T_j \| T_j' \boldsymbol{\beta})$ $\quad \mathcal{H}_\alpha(\ell) = [\boldsymbol{\alpha}]_1,\ \mathcal{H}_\beta(\ell) = [\boldsymbol{\beta}]_2$
Justif.	SXDH
$\mathsf{G}_{0.\gamma}$	$\mathsf{ct}_{x_i} = \begin{cases} \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^1 \| \mathbf{0} \| \boldsymbol{\alpha} U_i) & \ell < \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^0 \| \mathbf{0} \| \boldsymbol{\alpha} U_i) & \ell \geq \gamma \end{cases}$ $\boxed{\boldsymbol{\alpha} = \begin{cases} r_\ell \boldsymbol{a}^\perp & \ell \neq \gamma \\ \boldsymbol{\alpha}_\ell & \ell = \gamma \end{cases}}$
Justif.	dFH-security
$\mathsf{G}_{0.\gamma.i^*}$	$\mathsf{ct}_{x_i} = \begin{cases} \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^1 \| \mathbf{0} \| \boldsymbol{\alpha} U_i) & \ell < \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, \boxed{x_i^1 \| \boldsymbol{\alpha}} \| \boldsymbol{\alpha} U_i) & \ell = \gamma, i = i^* \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^0 \| \mathbf{0} \| \boldsymbol{\alpha} U_i) & \ell > \gamma \vee (\ell = \gamma, i \neq i^*) \end{cases}$ $\boldsymbol{\alpha} = \begin{cases} r_\ell \boldsymbol{a}^\perp & \ell \neq \gamma \\ \boldsymbol{\alpha}_\ell & \ell = \gamma \end{cases}$ $\mathsf{ct}_{y_j} \leftarrow \mathsf{dFH.KeyGen}(\boxed{y_j^0 \| \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0} \| T_j)$
Justif.	SXDH
$\mathsf{G}_{0.\gamma.i^*.j^*}$	$\boxed{T_j = \begin{cases} \mathbf{b}^\perp r_j & j \neq j^* \\ T_j & j = j^* \end{cases}}$
Justif.	dFH-security
$\mathsf{G}'_{0.\gamma.i^*.j^*}$	$\mathsf{ct}_{\boxed{x_{i^*}}} = \begin{cases} \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, \boxed{x_i^1 \| \mathbf{0} \| \boldsymbol{\alpha} \tilde{U}_i}) & \ell < \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, \boxed{x_i^1 \| \boldsymbol{\alpha} \| \boldsymbol{\alpha} \tilde{U}_i}) & \ell = \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, \boxed{x_i^0 \| \mathbf{0} \| \boldsymbol{\alpha} \tilde{U}_i}) & \ell > \gamma \end{cases}$ $\mathsf{ct}_{y_j} = \begin{cases} \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, y_j^0 \| \boldsymbol{\alpha}_\gamma^\perp y_j^0 \| T_j) & j < j* \\ \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, \boxed{y_j^0 \| \boldsymbol{\alpha}_\gamma^\perp y_j^0 \| T_j}) & j = j* \\ \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, y_j^0 \| \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 \| T_j) & j > j* \end{cases}$ $\boxed{U_{i^*} = \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle \cdot (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_{j^*}^{0\gamma} \cdot \mathbf{b}/\langle \mathbf{b}, T_{j^*} \rangle + \tilde{U}_{i^*}}$
$\mathsf{G}'_{0.\gamma.i^*}$	$\mathsf{sk}_f = \sum_{j \in [m], i \neq i^*} f_{i,j} U_i T_j + \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle \sum_{j \in [m]} f_{i^*,j} (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_j^{0\gamma} + \sum_{j \in [m]} f_{i^*,j} \tilde{U}_{i^*} T_j$
Justif.	dFH-security
$\mathsf{G}''_{0.\gamma.i^*}$	$\mathsf{ct}_{x_{i^*}} = \begin{cases} \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^1 \| \boxed{\mathbf{0}} \| \boldsymbol{\alpha} \tilde{U}_i) & \ell \leq \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^0 \| \mathbf{0} \| \boldsymbol{\alpha} \tilde{U}_i) & \ell > \gamma \end{cases}$ $\mathsf{ct}_{y_j} = \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, y_j^0 \| \boxed{\mathbf{0}} \| T_j)$
$\mathsf{G}'_{0.\gamma}$	$\mathsf{sk}_f = \sum_{j \in [m], i \in [n]} f_{i,j} \tilde{U}_i T_j$
G_1	$\mathsf{ct}_{x_i} \leftarrow \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^1 \| \mathbf{0} \| \boldsymbol{\alpha} \boxed{U_i} \| U_i')$ $\quad \mathsf{sk}_F = ([\sum f_{i,j} \boxed{U_i} T_j]_2,\ [\sum f_{i,j} U_i' T_j']_1)$ $\mathsf{ct}_{y_j} \leftarrow \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, y_j^0 \| \mathbf{0} \| T_j \| T_j' \boldsymbol{\beta})$ $\quad \mathcal{H}_\alpha(\ell) = \boldsymbol{\alpha},\ \mathcal{H}_\beta(\ell) = \boldsymbol{\beta}$
G_2	$\mathsf{ct}_{x_i} \leftarrow \mathsf{dFH.Enc}(\mathsf{pp}_{\mathsf{FH}}, \mathsf{msk}_{\mathsf{FH}}, x_i^1 \| \mathbf{0} \| \boldsymbol{\alpha} U_i \| U_i')$ $\quad \mathsf{sk}_F = ([\sum f_{i,j} U_i T_j]_2,\ [\sum f_{i,j} U_i' T_j']_1)$ $\mathsf{ct}_{y_j} \leftarrow \mathsf{dFH.KeyGen}(\mathsf{msk}_{\mathsf{FH}}, \boxed{y_j^1} \| \mathbf{0} \| T_j \| T_j' \boldsymbol{\beta})$ $\quad \mathcal{H}_\alpha(\ell) = \boldsymbol{\alpha},\ \mathcal{H}_\beta(\ell) = \boldsymbol{\beta}$

Fig. 12: Overview of games for 2-MCQFE. Here $\mathsf{msk}_{\mathsf{FH}} = \mathsf{ask}$

In the following theorem we prove the security of our 2-MCQFE for a weaker security notion where the constraint $\mathbf{x}_0^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_1$ is replaced with $\mathbf{x}_0^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_1$ (referred as the weak security). We later give a transformation to turn back to the standard constraints.

Theorem 4. *If* dFH *is dFH-secure, then our 2-MCQFE scheme is weakly one-sel-*IND *secure.*

Proof. The proof proceeds via a sequence of the games which are summarized in Fig. 12. At first we define a hybrid over labels ℓ where two such adjacent hybrids are indistinguishable via a hybrids over index i, and similarly two adjacent hybrids (over i) are indistinguishable via

a hybrids over index j. More detailed, we pass through the games (Fig. 12) in the following order: $\mathsf{G}_0 \overset{\ell}{\rightsquigarrow} \mathsf{G}_1 \overset{\ell}{\rightsquigarrow} \mathsf{G}_2$, where the transition from G_0 to G_1 changes the message x_i^0 to x_i^1 for all the labels and $\overset{\ell}{\rightsquigarrow}$ includes the following hybrids on the labels. Which means each time we change x_i^0 to x_i^1 associated with label $\gamma = 1, \ldots, Q$ where Q is the number of labels:

$$\mathsf{G}_{0.1} \overset{i}{\rightsquigarrow} \mathsf{G}'_{0.1} \ldots \rightsquigarrow \ldots \mathsf{G}_{0.Q} \overset{i}{\rightsquigarrow} \mathsf{G}'_{0.Q}$$

where for label γ, the transition from $\mathsf{G}_{0.\gamma}$ to $\mathsf{G}'_{0.\gamma}$ changes the message x_i^0 to x_i^1 for all indices i. That is, each time we change $x_{i^*}^0$ to $x_{i^*}^1$ for index $i^* = 1, \ldots, n$. Thus in the above sequence, $\overset{i}{\rightsquigarrow}$ stands for the following hybrids on index i.

$$\mathsf{G}_{0.\gamma.1} \overset{j}{\rightsquigarrow} \mathsf{G}'_{0.\gamma.1} \rightsquigarrow \mathsf{G}''_{0.\gamma.1} \ldots \rightsquigarrow \ldots \mathsf{G}_{0.\gamma.n} \overset{j}{\rightsquigarrow} \mathsf{G}'_{0.\gamma.n} \rightsquigarrow \mathsf{G}''_{0.\gamma.n}$$

where for label γ and index i^*, the transition from $\mathsf{G}_{0.\gamma.i^*}$ to $\mathsf{G}''_{0.\gamma.i^*}$, loosely speaking, changes $x_{i^*}^0 y_j^0$ to $x_{i^*}^1 y_j^0$ for all the indices of j (more precisely, it changes $x_{i^*}^0$ to $x_{i^*}^1$ in the leakage from mix-and-matches depending on the indices i, j). Meaning that, each time we change $x_{i^*}^0 y_{j^*}^0$ to $x_{i^*}^1 y_{j^*}^0$ for $j^* = 1, \ldots, m$. Thus in the above sequence, $\overset{j}{\rightsquigarrow}$ includes the following hybrids on index j:

$$\mathsf{G}_{0.\gamma.i^*.1} \rightsquigarrow \mathsf{G}'_{0.\gamma.i^*.1} \ldots \rightsquigarrow \ldots \mathsf{G}_{0.\gamma.i^*.m} \rightsquigarrow \mathsf{G}'_{0.\gamma.i^*.m}$$

The transition from G_1 to G_2 proceeds similarly to change y_j^0 to y_j^1. Here we precisely describe each game.

$\boxed{\mathsf{G}_0}$: is the real game associated with bit $b = 0$.

$\boxed{\mathsf{G}_{0,\gamma}}$: is similar to its previous game, except that, the random oracle queries $\mathcal{H}_\alpha(\ell)$ associated with labels $\ell \neq \gamma$ are answered with the same structure $[r_\ell \mathbf{a}^\perp]_1$ for fresh randoms r_ℓ and a fixed uniformly chosen vector $\mathbf{a}^\perp \in \mathbb{Z}_q^{1\times 2}$. While the random oracle for $\ell = \gamma$ is answered by a uniformly sampled $[\boldsymbol{\alpha}_\gamma]_1 \in \mathbb{G}_1^{1\times 2}$. To prove the indistinguishability of these adjacent games, we rely on the MDDH assumption in the group $\mathbb{G}_1$, w.l.g for the simplicity one can consider the indistinguishability of G_0 and $\mathsf{G}_{0.1}$.

By *Random Self-Reducibility* (RSR) of MDDH assumption (Lemma 1 in [14]), we have, $(g^{\mathbf{a}^\perp}, \{g^{\mathbf{a}^\perp r_\ell}\}_{\ell\in[Q]}) \cong (g^{\mathbf{a}^\perp}, \{g^{R_\ell}\}_{\ell\in[Q]})$ where $\mathbf{a}^\perp, R_\ell \xleftarrow{R} \mathbb{Z}_q^{1\times 2}, r_\ell \xleftarrow{R} \mathbb{Z}_q$.

The simulator receives the challenges $(g_1^{\mathbf{a}^\perp}, \{g_1^{b_\ell}\}_{\ell\in[Q]})$ from the challenger of RSR-MDDH. For $\ell \neq 1$ it sets $[\boldsymbol{\alpha}_\ell]_1 = g_1^{b_\ell}$, and for $\ell = 1$ it samples $[\boldsymbol{\alpha}_1]_1 \xleftarrow{R} \mathbb{G}_1^{1\times 2}$. Now if $b_\ell = \mathbf{a}^\perp r_\ell$, it simulates the game $\mathsf{G}_{0,1}$ otherwise it simulates the game G_0. In fact, it simulates all the queries based on the real algorithms (w.r.t the new values $\boldsymbol{\alpha}$). Since all the queries involved with $\boldsymbol{\alpha}$, can be answered via $[\boldsymbol{\alpha}]_1$, this is a correct reduction (note that $U_i \in \mathbb{Z}_q$, and thus in ct_{x_i}, $\boldsymbol{\alpha} U_i$ can be replaced with $[\boldsymbol{\alpha} U_i]_1$).

$\boxed{\mathsf{G}_{0.\gamma.i^*}}$: is similar to the previous game, except for the simulation of ct_{x_i}, ct_{y_j}:

- in ct_{x_i}, for $\ell = \gamma, i = i^*$ we replace the message-part $X_{i,\ell} = x_i^0 || \mathbf{0}$ with $X'_{i,\ell} = x_i^1 || \boldsymbol{\alpha}_\gamma$.
- in ct_{y_j} for all ℓ, j, $Y_{j,\ell} = y_j^0 || \mathbf{0}$ is replaced with $Y'_{j,\ell} = y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0$.

We emphasize that for $\ell \neq \gamma$ or $i \neq i^*$ message is not changed i.e., $X_{i,\ell} = X'_{i,\ell}$ Also note that in any case, the other parts of the message, namely $\boldsymbol{\alpha} U_i || U'_i$ and $T_j || T'_j \boldsymbol{\beta}$ are not changed.

Now, one can verify that:

$$X'_{i,\ell} \cdot Y'_{j,\ell} = X_{i,\ell} \cdot Y_{j,\ell} \quad \underline{\underline{\forall i, j, \ell}} \tag{4}$$

we consider different cases:

– for $\ell < \gamma$, $i \neq i^*$(same reasoning for all $\ell \in [Q]$, and $i \neq i^*$):

$$X_{i,\ell} \cdot Y_{j,\ell} = \langle x_i^1 || \mathbf{0} \ , \ y_j^0 || \mathbf{0} \rangle = x_i^1 y_j^0 = \langle x_i^1 || \mathbf{0} \ , \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_1 \gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 \rangle = X'_{i,\ell} \cdot Y'_{j,\ell}$$

– for $\ell > \gamma$, $i = i^*$ (same reasoning for $\ell < \gamma$, $i = i^*$):

$$X_{i,\ell} \cdot Y_{j,\ell} = \langle x_i^0 || \mathbf{0}, y_j^0 || \mathbf{0} \rangle = x_i^0 y_j^0 = \langle x_i^0 || \mathbf{0} \ , \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_1 \gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 \rangle = X'_{i,\ell} \cdot Y'_{j,\ell}$$

– for $\ell = \gamma$, $i = i^*$:

$$X_{i,\ell} \cdot Y_{j,\ell} = \langle x_i^0 || \mathbf{0}, y_j^0 || \mathbf{0} \rangle = x_i^0 y_j^0 = \langle x_i^1 || \boldsymbol{\alpha}_\gamma \ , \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 \rangle = X'_{i,\ell} \cdot Y'_{j,\ell}$$

The relation $X'_{i,\ell} \cdot Y'_{j,\ell} = X_{i,\ell} \cdot Y_{j,\ell}$ is exactly the required constraint on the challenges for dFH-security of dFH. Thus, by relying on the dFH-security of dFH, games $\mathsf{G}_{0.\gamma.i^*}$ is indistinguishable from its previous game. Note that in this reduction all other queries are answered by the real algorithms, and corruption on the authorities can be simulated by forwarding them to the challenger of dFH. The detailed simulation is as follows:

– the simulator samples all the keys except $(\mathsf{msk}_{\mathsf{FH}}, \mathsf{mpk}_{\mathsf{FH}}) = (\mathsf{ask}, \mathsf{pp})$.
– when it receives the challenge (x_i^0, x_i^1) for label ℓ, the simulator builds $m_0 = (X_{i,\ell} || \boldsymbol{\alpha} U_i || U'_i)$ and $m_1 = (X'_{i,\ell} || \boldsymbol{\alpha} U_i || U'_i)$ where $X_{i,\ell}$, $X'_{i,\ell}$ are defined as above. Then it sends (m_0, m_1) to its challenger.
– when it receives the challenge (y_j^0, y_j^1) for label ℓ, the simulator builds $F_0 = (Y_{j,\ell} || T_j || T'_j \boldsymbol{\beta})$ and $F_1 = (Y'_{j,\ell} || T_j || T'_j \boldsymbol{\beta})$ where $Y_{j,\ell}$, $Y'_{j,\ell}$ are defined as above. Then it sends (F_0, F_1) to its challenger.
– the other queries are simulated by the real algorithms.

By Eq. (4), we have $\langle m_0, F_0 \rangle = \langle m_1, F_1 \rangle$ which means this is a correct simulation. If the challenger of d-FHIP scheme responds by bit $b = 1$, this is the simulation of game $\mathsf{G}_{0.\gamma.i^*}$. Otherwise it is the simulation of previous game (which is $\mathsf{G}_{0,\gamma}$ or $\mathsf{G}''_{0,\gamma,i^*-1}$).

$\boxed{\mathsf{G}_{0.\gamma.i^*.j^*}}$: is similar to the previous game, except that, the values of $T_j \in \mathbb{G}_2$ for $j \neq j^*$ is replaced with $[\mathbf{b}^\perp r_j]_2$ for a fixed vector $\mathbf{b}^\perp \in \mathbb{Z}_q^{2\times 1}$ and fresh randoms r_j. While for j^* we sample T_{j^*} uniformly from $\mathbb{G}_2^{2\times 1}$. A similar reasoning to the transition from G_0 to $\mathsf{G}_{0,\gamma}$ on group $\mathbb{G}_2$ works here. Since in our construction the values T_j only appear in the group $\mathbb{G}_2$, the reduction to RSR-MDDH assumption in $\mathbb{G}_2$ is doable. In fact, in the construction we have $[T_j]_2$ as the secret-key and $\mathsf{sk}_f = [\sum_{i,j} f_{i,j} U_i T_j]_2$ as the functional-key making the simulation of corruption and functional-key queries completely doable.

$\boxed{\mathsf{G}'_{0.\gamma.i^*.j^*}}$: is similar to the game $\mathsf{G}_{0.\gamma.i^*j^*}$, except for the simulation of ct_{x_i}, ct_{y_j}:

– in ct_{x_i}, for all ℓ and $i = i^*$, we replace the message-part U_{i^*} with $\tilde{U}_{i^*}$ where the following relation holds for a uniformly sampled $\tilde{U}_i$ (Note that here we are looking at U_i as a part of the message in the inputs of $\mathsf{dFH.Enc}(\cdot)$).

$$U_{i^*} = \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle \cdot (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_{j^*}^{0\gamma} \cdot \mathbf{b}/\langle \mathbf{b}, T_{j^*} \rangle + \tilde{U}_{i^*} \tag{5}$$

– in ct_{y_j} for all ℓ and $j = j^*$, we replace the message-part $Y_{j,\ell} = y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 || T_j$ with $Y'_{j,\ell} = y_j^0 || \boldsymbol{\alpha}_\gamma^\perp y_j^0 || T_j$ (Note that for $j \neq j^*$ nothing is changed).

To be clear, this means for ct_{x_i} and ct_{y_j} we have:

$$\mathsf{ct}_{x_i} = \begin{cases} \mathsf{dFH.Enc}(\mathsf{pp_{FH}}, \mathsf{msk_{FH}}, x_i^1||\mathbf{0}||\boldsymbol{\alpha}\widehat{U}_i||U_i') & \ell < \gamma \\ \mathsf{dFH.Enc}(\mathsf{pp_{FH}}, \mathsf{msk_{FH}}, x_i^1||\boldsymbol{\alpha}||\boldsymbol{\alpha}\widehat{U}_i||U_i') & \ell = \gamma, i = i^* \\ \mathsf{dFH.Enc}(\mathsf{pp_{FH}}, \mathsf{msk_{FH}}, x_i^0||\mathbf{0}||\boldsymbol{\alpha}\widehat{U}_i||U_i') & \ell > \gamma \vee (\ell = \gamma, i \neq i^*) \end{cases} \tag{6}$$

where in the previous game $\widehat{U}_{i^*} = U_{i^*}$ and in the current game $\widehat{U}_{i^*} = \tilde{U}_{i^*}$ (for $i \neq i^*$, $\widehat{U}_i$ is the same in both games).

$$\mathsf{ct}_{y_j} = \begin{cases} \mathsf{dFH.KeyGen}(\mathsf{msk_{FH}}, y_j^0||\boldsymbol{\alpha}_\gamma^\perp y_j^0||T_j||T_j'\boldsymbol{\beta}) & j < j* \\ \mathsf{dFH.KeyGen}(\mathsf{msk_{FH}}, \widehat{Y}_{j,\ell}||T_j'\boldsymbol{\beta}) & j = j* \\ \mathsf{dFH.KeyGen}(\mathsf{msk_{FH}}, y_j^0||\mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma\rangle(x_{i^*}^0 - x_{i^*}^1)y_j^0||T_j||T_j'\boldsymbol{\beta}) & j > j* \end{cases} \tag{7}$$

where in the previous game $\widehat{Y}_{j,\ell} = Y_{j,\ell}$ and in the current game $\widehat{Y}_{j,\ell} = Y'_{j,\ell}$ and $Y_{j,\ell}$ and $Y'_{j,\ell}$ are defined as above.

Note that the secret-key U_{i^*} in Eq. (5) is perfectly indistinguishable from the secret-key in the previous game. Moreover, one can verify that:

$$X'_{i,\ell} \cdot Y'_{j,\ell} = X_{i,\ell} \cdot Y_{j,\ell} \quad \underline{\underline{\forall i, j, \ell}}$$

where $Y_{j,\ell}$ and $Y'_{j,\ell}$ (res. for the previous and current game) are generally defined to be the message-part after removing $T_j'\boldsymbol{\beta}$ from the original message in ct_{y_j}. Similarly, $X_{i,\ell}$ and $X'_{i,\ell}$ are the message-part in ct_{x_i} by ignoring U_i' from the original message.

Let us verify the mentioned relation for each case:

- for $\ell > \gamma$, $i \neq i^*$, $j < j^*$ (similar reasoning for all $\ell \in [Q]$, $i \neq i^*$, $j \neq j^*$):

$$X_{i,\ell} \cdot Y_{j,\ell} = \langle x_i^0||\mathbf{0}||\boldsymbol{\alpha}\widehat{U}_i,\ y_j^0||\boldsymbol{\alpha}_\gamma^\perp y_j^0||T_j\rangle = X'_{i,\ell} \cdot Y'_{j,\ell}$$

- for $\ell < \gamma$, $i = i^*$, $j > j^*$ (similar reasoning for $\ell \neq \gamma$, $i = i^*$,$j \neq j^*$):

$$\begin{aligned} &X_{i^*,\ell} \cdot Y_{j,\ell} = \langle x_i^1||\mathbf{0}||\boldsymbol{\alpha}U_i\ ,\ y_j^0||\mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma\rangle(x_{i^*}^0 - x_{i^*}^1)y_j^0||T_j\rangle = \\ &x_i^1 y_j^0 + \boldsymbol{\alpha}U_{i^*}T_j = x_i^1 y_j^0 + \boldsymbol{\alpha}\tilde{U}_{i^*}T_j \qquad (*) \\ &= \langle x_i^1||\mathbf{0}||\boldsymbol{\alpha}\tilde{U}_{i^*}\ ,\ y_j^0||\mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma\rangle(x_{i^*}^0 - x_{i^*}^1)y_j^0||T_j\rangle = X'_{i^*,\ell} \cdot Y'_{j,\ell} \end{aligned}$$

  where the equality in line (*) is due to the fact that $\boldsymbol{\alpha}_\ell \mathbf{a} = 0$, for $\ell \neq \gamma$.

- for $\ell = \gamma$, $i = i^*$, $j < j^*$ (similar reasoning for $\ell = \gamma$, $i = i^*$,$j \neq j^*$):

$$\begin{aligned} &X_{i^*,\ell} \cdot Y_{j,\ell} = \langle x_i^1||\boldsymbol{\alpha}_\gamma||\boldsymbol{\alpha}U_i\ ,\ y_j^0||\boldsymbol{\alpha}_\gamma^\perp y_j^0||T_j\rangle = \\ &x_i^1 y_j^0 + \boldsymbol{\alpha}U_{i^*}T_j = x_i^1 y_j^0 + \boldsymbol{\alpha}\tilde{U}_{i^*}T_j \qquad (**) \\ &= \langle x_i^1||\boldsymbol{\alpha}_\gamma||\boldsymbol{\alpha}\tilde{U}_{i^*}\ ,\ y_j^0||\boldsymbol{\alpha}_\gamma^\perp y_j^0||T_j\rangle = X'_{i^*,\ell} \cdot Y'_{j,\ell} \end{aligned}$$

  where the equality in line (**) is due to the fact that $\mathbf{b}T_j = 0$, for $j \neq j^*$.

– for $\ell < \gamma$, $i \neq i^*$, $j = j^*$ (similar reasoning for all $\ell \in [Q]$, $i \neq i^*$, $j = j^*$):

$$\begin{aligned} X_{i,\ell} \cdot Y_{j,\ell} &= \langle x_i^1 || \mathbf{0} || \boldsymbol{\alpha} \widehat{U}_i \ , \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 || T_j \rangle = \\ &= \langle x_i^1 || \mathbf{0} || \boldsymbol{\alpha} \widehat{U}_i \ , \ y_j^0 || \boldsymbol{\alpha}_\gamma^\perp y_j^0 || T_j \rangle = X'_{i^*,\ell} \cdot Y'_{j,\ell} \end{aligned}$$

– for $\ell > \gamma$, $i = i^*$, $j = j^*$ (similar reasoning for $\ell < \gamma$, $i = i^*$, $j = j^*$):

$$\begin{aligned} & X_{i^*,\ell} \cdot Y_{j,\ell} = \langle x_i^0 || \mathbf{0} || \boldsymbol{\alpha} U_i \ , \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 || T_j \rangle = \\ & x_i^0 y_j^0 + \boldsymbol{\alpha} U_{i^*} T_j = x_i^0 y_j^0 + \boldsymbol{\alpha} \tilde{U}_{i^*} T_j \qquad (*') \\ & = \langle x_i^0 || \mathbf{0} || \boldsymbol{\alpha} \tilde{U}_{i^*} \ , \ y_j^0 || \boldsymbol{\alpha}_\gamma^\perp y_j^0 || T_j \rangle = X'_{i^*,\ell} \cdot Y'_{j,\ell} \end{aligned}$$

where the equality in line $(*')$ is due to the fact that $\boldsymbol{\alpha}_\ell \mathbf{a} = 0$, for $\ell \neq \gamma$.

– for $\ell = \gamma$, $i = i^*$, $j = j^*$:

$$\begin{aligned} & X_{i,\ell} \cdot Y_{j,\ell} = \langle x_i^1 || \boldsymbol{\alpha}_\gamma || \boldsymbol{\alpha} U_i, \ y_j^0 || \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle (x_{i^*}^0 - x_{i^*}^1) y_j^0 || T_j \rangle = \\ & x_i^0 y_j^0 + \boldsymbol{\alpha} U_i T_j = x_i^0 y_j^0 + (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_{j^*}^{0\gamma} + \boldsymbol{\alpha}_\gamma \tilde{U}_{i^*} T_{j^*} \qquad (**') \\ & = x_i^1 y_j^0 + \boldsymbol{\alpha}_\gamma \tilde{U}_{i^*} T_{j^*} = \langle x_i^1 || \boldsymbol{\alpha}_\gamma || \boldsymbol{\alpha} \tilde{U}_i, \ y_j^0 || \boldsymbol{\alpha}_\gamma^\perp y_j^0 || T_j \rangle = X'_{i,\ell} \cdot Y'_{j,\ell} \end{aligned}$$

where the equality in line $(**')$ is due to the Eq. (5) and the fact that $\ell = \gamma$ and $j = j^*$.

Thus, having $X'_{i^*,\ell} \cdot Y_{j,\ell} = X_{i^*,\ell} \cdot Y_{j,\ell} \quad \forall i, j, \ell$, allows us to reduce the indistinguishability of concerned games to the dFH-security of dFH. Note that other queries, including corruption-queries and functional-key queries, can be simulated simply by running the real algorithms while authority-corruptions would be forwarded to the challenger of dFH.

$\boxed{\mathsf{G}'_{0.\gamma.i^*}}$: is the same as game $\mathsf{G}'_{0.\gamma.i^*.m}$ when the functional-key sk_f ir rewritten based on updates in the value of U_{i^*} (so, these games are identical). Note that for each index i^*, we have the following hybrids on index j leading to the updates in the value of U_{i^*}:

$$\mathsf{G}_{0.\gamma.i^*} \longrightarrow \mathsf{G}_{0.\gamma.i^*.1} \rightsquigarrow \mathsf{G}'_{0.\gamma.i^*.1} \rightsquigarrow \ldots \rightsquigarrow \mathsf{G}_{0.\gamma.i^*.m} \rightsquigarrow \mathsf{G}'_{0.\gamma.i^*.m} \longrightarrow \mathsf{G}'_{0.\gamma.i^*}$$

Therefore, to see how the value of sk_f is changed between $\mathsf{G}_{0.\gamma.i^*}$ and $\mathsf{G}'_{0.\gamma.i^*}$, we need to follow the updates during the above path.

At the beginning, the value of sk_f in $\mathsf{G}_{0.\gamma.i^*.1}$ (and $\mathsf{G}_{0.\gamma.i^*}$) is :

$$\mathsf{sk}_f = \sum f_{i,j} U_i T_j = \underbrace{\boxed{\sum_{j \in [m], i \neq i^*} f_{i,j} U_i T_j}}_{K} + \sum_{j \in [m]} f_{i^*,j} U_{i^*} T_j$$

Then sk_f in $\mathsf{G}'_{0.\gamma.i^*.1}$ is updated (by updating U_i^*) to:

$$\mathsf{sk}_f = K + \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle \cdot f_{i^*,1} (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_1^{0\gamma} + \sum_{j \in [m]} f_{i^*,j} \tilde{U}_{i^*} T_j \qquad (8)$$

where the second term is due to the fact that $\mathbf{b} T_j = 0$ for $j \neq j^*$ and $\mathbf{b} T_{j^*} \neq 0$. Similarly going through all the m hybrids, sk_f in the current game $\mathsf{G}'_{0.\gamma.i^*}$ is (note that in Eq. (8), every time only the last term changes):

$$\mathsf{sk}_f = K + \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma \rangle \sum_{j \in [m]} f_{i^*,j} (x_{i^*}^{1\gamma} - x_{i^*}^{0\gamma}) y_j^{0\gamma} + \sum_{j \in [m]} f_{i^*,j} \tilde{U}_{i^*} T_j$$

$\boxed{\mathsf{G}''_{0.\gamma.i^*}}$: is similar to the game $\mathsf{G}'_{0.\gamma.i^*}$, except that, for the simulation of ct_{x_i} and ct_{y_j}:
– in ct_{x_i} for $\ell = \gamma,\ i = i^*$, we replace the message-part $X_{i,\ell} = x_i^1||\boldsymbol{\alpha}$ with $X'_{i,\ell} = x_i^1||\mathbf{0}$.
– in ct_{y_j}: for all ℓ and j, we replace the message-part $Y_{j,\ell} = y_j^0||\boldsymbol{\alpha}_\gamma^\perp y_j^0$ with $Y'_{j,\ell} = y_j^0||\mathbf{0}$.

Similar to the game $\mathsf{G}_{0.\gamma.i^*}$, one can verify that: $X_{i,\ell} \cdot Y_{j,\ell} = X'_{i,\ell} \cdot Y'_{j,\ell} \quad \forall i, j, \ell$ allowing to rely on the dFH-security of dFH.

$\boxed{\mathsf{G}'_{0.\gamma}}$: is the same as game $\mathsf{G}''_{0.\gamma.n}$ when the functional-key sk_f ir rewritten based on the updates in values of U_i. Note that for each label γ, we have the following hybrids on index i:

$$\mathsf{G}_{0.\gamma} \longrightarrow \mathsf{G}_{0.\gamma.1} \rightsquigarrow \mathsf{G}'_{0.\gamma.1} \rightsquigarrow \mathsf{G}''_{0.\gamma.1} \rightsquigarrow \ldots \rightsquigarrow \mathsf{G}_{0.\gamma.n} \rightsquigarrow \mathsf{G}'_{0.\gamma.n} \rightsquigarrow \mathsf{G}''_{0.\gamma.n} \longrightarrow \mathsf{G}'_{0.\gamma}$$

Leading to the following functional-key in the current game (with a similar reasoning to the game $\mathsf{G}'_{0.\gamma.i^*}$):

$$\mathsf{sk}_f = \sum_{j\in[m],i\in[n]} \Big(f_{i,j}\tilde{U}_i T_j + \mathbf{a}/\langle \mathbf{a}, \boldsymbol{\alpha}_\gamma\rangle f_{i,j}(x_i^{0\gamma} - x_i^{1\gamma}) y_j^{0\gamma} \Big) = \sum_{j\in[m],i\in[n]} f_{i,j}\tilde{U}_i T_j$$

where the last equality is due to the constraints $\mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_0^T \boldsymbol{F} \mathbf{y}_0$.

$\boxed{\mathsf{G}_1}$: is similar to $\mathsf{G}'_{0,Q}$,except that, the secret-key $\tilde{U}_i$ is replaced with the key U_i in all the queries and challenges. Here we are using the fact that for a corrupted client/slot i, we have $x_i^1 = x_i^0$ leading to answer the corruption queries by $\tilde{U}_i$ (due to Eq. (5)). Meaning that, in adversary's point of view $\tilde{U}_i$'s are the secret keys.

$\boxed{\mathsf{G}_2}$: is similar to the previous game , except that, y_j^0 is replaced with y_j^1. The transition from game G_1 to G_2 is similar to the transition from G_0 to G_1 considering the hybrids on indices j and then i, and using the constraint $\mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_1$. □

In the following we use a similar transformation used in [27] to lift the security from the weak version to a stronger one with standard constraints over the queries. This transformation is equivalent with adding n dummy clients on the x-side and m dummy clients on the y-side, which encrypt 0 for all the labels.

Theorem 5. *In Theorem 4, if we replace* $\mathbf{x}$ *with* $\mathbf{x}||\mathbf{0} \in \mathbb{Z}_q^{2n}$, $\mathbf{y}$ *with* $\mathbf{y}||\mathbf{0} \in \mathbb{Z}_q^{2m}$ *and* $\boldsymbol{F}$ *with* $F' = \left(\begin{smallmatrix} \boldsymbol{F} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{F} \end{smallmatrix}\right)$, *the security holds w.r.t the standard constraints* $\mathbf{x}_0^T \boldsymbol{F} \mathbf{y}_0 = \mathbf{x}_1^T \boldsymbol{F} \mathbf{y}_1$. *More Precisely, if our 2-MCQFE is weakly secure, then the mentioned transformation results in a 2-MCQFE scheme with standard constraints.*

Proof. Let ct_X be the set of all the x-side ciphertexts associated with message X (and ct_Y is defined similarly) and $\mathsf{sk}_{F'}$ is the functional key associated with the matrix F' defined as above. We proceed through a sequence of hybrids as:

$$\{\mathsf{ct}_{\mathbf{x}^0||0^n}, \mathsf{ct}_{\mathbf{y}^0||0^m}, \mathsf{sk}_{F'}\} \cong \{\mathsf{ct}_{\mathbf{x}^0||\mathbf{x}^1}, \mathsf{ct}_{\mathbf{y}^0||0^m}, \mathsf{sk}_{F'}\} \overset{*}{\cong}$$
$$\{\mathsf{ct}_{\mathbf{x}^1||\mathbf{x}^1}, \mathsf{ct}_{\mathbf{y}^1||0^m}, \mathsf{sk}_{F'}\} \cong \{\mathsf{ct}_{\mathbf{x}^1||0^n}, \mathsf{ct}_{\mathbf{y}^1||0^m}, \mathsf{sk}_{F'}\}$$

Where all the relations $\cong$ holds due to the weak security of 2-MCQFE. Moreover, $\overset{*}{\cong}$ holds by the standard constraints in the theorem. □

5.1 From one to many

Now we upgrade the security from "one" to "many" (see Definition 8). Intuitively, the construction can not support many ciphertexts per label, because for a fixed label with different ciphertexts there is no randomness. Thus, the idea is that we add randomness to the ciphertext, but in a way that it can be removed later. To do so, we add a layer of 2-input QFE (i.e., our scheme in Fig. 7)) over a 2-MCQFE.

For this at first we need to slightly modify our 2-input QFE in a way that the FHIP scheme is replaced with our dFH scheme (Fig. 9). In the following corollary we analyze the security of our extended 2-input QFE. The security notion is as Definition 8 when there are only two clients and the label set Labels has only a single element. As there is only a fixed label, we don't need to explicitly write it.

Corollary 1. *If* dFH *is secure, then our extended 2-input QFE scheme is many-sel-*IND *secure.*

Let $\mathbf{x}_k$ and $\mathbf{y}_t$ be the k-th and t-th queries issued by the first and the second client (res.). The security proof is similar to the security proof of Theorem 4, where instead of the hybrid over the labels we consider a hybrid over the query-number k (to go from G_0 to G_1) and a hybrid over query-number t (to go from G_1 to G_2). In fact, every thing is the same except that instead of working with the random oracles we work with randomnesses chosen by the clients.

Let the ciphertext associated with x_i, in 2-input QFE be as Eq. (9), and in 2-MCQFE be as Eq. (10) (in a general form).

$$\mathsf{ct}_{0,x} = [r]_1,\ \mathsf{ct}_{0,y} = [r']_2, \quad \mathsf{ct}_{x_i} = \mathsf{FH.Enc}(\mathbf{x}_i||V_{i,r}) \quad \mathsf{ct}_{y_j} = \mathsf{FH}(\mathbf{y}_j||V'_{j,r'}) \tag{9}$$

$$\mathsf{ct}^*_{x_{i,\ell}} = \mathsf{dFH.Enc}(\mathbf{x}_i||W_{i,\ell}) \qquad \mathsf{ct}^*_{y_{j,\ell}} = \mathsf{dFH.Enc}(\mathbf{y}_j||W'_{j,\ell}) \tag{10}$$

where $V_{i,r}$ is a combination of some secret keys and randomness r and $W_{i,\ell}$ is a combination of some secret keys and hash functions over ℓ (similarly about $V'_{i,r}$ and $W'_{i,\ell}$). Then we build our many-secure 2-MCQFE (called 2-MCQFE') as follows:

$$\begin{aligned} &\mathsf{ct}_{x_{i,\ell}} = \mathsf{dFH.Enc}(\mathbf{x}_i||\mathbf{0}^{2k}||V_{i,r}||W_{i,\ell}) \quad \mathsf{ct}_{0,x_i} = [r]_1,\ \mathsf{ct}_{0,y_j} = [r']_2 \\ &\mathsf{ct}_{y_{j,\ell}} = \mathsf{dFH.Enc}(\mathbf{y}_j||\mathbf{0}^{2k}||V'_{j,r'}||W'_{j,\ell}) \quad \mathsf{sk}_F = (\{\mathsf{sk}_{ij}\}_{i,j}, \mathsf{sk}^*_F) \end{aligned} \tag{11}$$

Where ct_{0,x_i} is associated with the randomness chosen individually by the x-side client i, sk_{ij} and sk^*_F are the functional key for 2-input QFE and 2-MCQFE schemes respectively, and $F \in \mathbb{Z}_q^{kn\times km}$, $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{Z}_q^k$, where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$.

For the index i, we denote k-th query as $\mathbf{x}_{i_k}$, the leakage from ideal functionality (i.e., $\mathbf{x}^T\boldsymbol{F}\mathbf{y}$) can be considered as:

$$\mathbf{x}^T_{i_k}\boldsymbol{f}_{i,j}\mathbf{y}_{j_t} - \mathbf{x}^T_{i_1}\boldsymbol{f}_{i,j}\mathbf{y}_{j_1} \tag{12}$$

Which means in the security game we have,

$$(\mathbf{x}^0_{i_k})^T\boldsymbol{f}_{i,j}\mathbf{y}^0_{j_t} - (\mathbf{x}^0_{i_1})^T\boldsymbol{f}_{i,j}\mathbf{y}^0_{j_1} = (\mathbf{x}^1_{i_k})^T\boldsymbol{f}_{i,j}\mathbf{y}^1_{j_t} - (\mathbf{x}^1_{i_1})^T\boldsymbol{f}_{i,j}\mathbf{y}^1_{j_1}$$

Remark 2. Here we discus on the leakage of ideal functionality. One can formulate the leakage in different (but equivalent) ways so that it can help the proof.

1. if we start with the first query for index j^* and we just change the query on index i^* to the k-th query. Then the leakage is: $(\mathbf{x}_{i_k^*} - \mathbf{x}_{i_1^*}) \sum_j \boldsymbol{f}_{i^*,j} \mathbf{y}_{j_1}$.

2. in the previous case we assume that we have started with t-th query on the index j^*, so: $(\mathbf{x}_{i_k^*} - \mathbf{x}_{i_1^*})(\sum_{j \neq j^*} \boldsymbol{f}_{i,j} \mathbf{y}_{j_1} + \boldsymbol{f}_{i^*,j^*} \mathbf{y}_{j_t^*})$

3. Putting two previous leakage we get: $(\mathbf{x}_{i_k^*} - \mathbf{x}_{i_1^*}) \boldsymbol{f}_{i^*,j^*} (\mathbf{y}_{j_t^*} - \mathbf{y}_{j_1^*})$ for any i_k^* and j_t^*

4. Now in the security game the adversary can issues queries, if it asks for a query $\mathbf{y}_{j_t^*} = \mathbf{0}$, then it can find $(\mathbf{x}_{i_k^*} - \mathbf{x}_{i_1^*}) \boldsymbol{f}_{i^*,j^*} \mathbf{y}_{j_1^*}$ (via the equation in step 3).

5. Consequently it finds $(\mathbf{x}_{i_k^*} - \mathbf{x}_{i_1^*}) \boldsymbol{f}_{i^*,j^*} \mathbf{y}_{j_{t'}^*}$ for the t'-th query on j^* (since the equation in (3) holds for any j_t^*).

6. Similarly it can find $\mathbf{x}_{i_1^*} \boldsymbol{f}_{i^*,j^*} (\mathbf{y}_{j_{t'}^*} - \mathbf{y}_{j_1^*})$.

7. Putting steps 5 and 6 together, it finds the leakage given in Eq. (12).

Note that we computed a special form of the leakage from the original one, so if the proof is working for this leakage then the scheme is secure for the original case.

Now we address a special property of Lin's FHIP scheme (and consequently our dFH scheme) which is used in the proof.

Property 3. Given the encryption associated with $\mathbf{x}$, one can compute the encryption of $\mathbf{x}||\mathbf{x}'$. Similarly given the functional key associated with $\mathbf{y}$, one can compute the functional key for $\mathbf{y}||\mathbf{y}'$.

To see this, let FH generates the ciphertext and functional key corresponding with $\mathbf{x}$ and $\mathbf{y}$ where $\mathsf{msk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ is the master secret key. Then the ciphertext and functional key are in the form given in Property 2, and by sampling msk', and having the encoded randomness used in ct_x, we can compute.

$$\mathsf{FH.Enc}(\mathsf{msk}||\mathsf{msk}', \mathbf{x}||\mathbf{x}') = \big(\langle \mathsf{sk}_2||\mathsf{sk}_2', \mathsf{ct}_x||\mathsf{ct}_{x'} \rangle,\ \underline{\mathsf{ct}_x}||\mathsf{ct}_{x'}\big)$$

where

$$\langle \mathsf{sk}_2||\mathsf{sk}_2', \mathsf{ct}_x||\mathsf{ct}_{x'} \rangle = \underline{\langle \mathsf{sk}_2, \mathsf{ct}_x \rangle} + \langle \mathsf{sk}_2', \mathsf{ct}_{x'} \rangle.$$

and the underlined part comes from the encryption of $\mathbf{x}$. And similarly,

$$\mathsf{FH.KeyGen}(\mathsf{msk}||\mathsf{msk}', \mathbf{y}||\mathbf{y}') = \Big(\mathsf{Enc}(\mathsf{sk}_2^0 + \mathsf{sk}_2'^0, -\langle \mathsf{sk}_1||\mathsf{sk}_1', \mathbf{y}||\mathbf{y}' \rangle; [r]_2),\ \underline{\mathsf{ct}_{\mathbf{y}}}||\mathsf{ct}_{\mathbf{y}'}\Big)$$

where

$$\begin{aligned} &\mathsf{Enc}(\mathsf{sk}_2^0 + \mathsf{sk}'^0_2, -\langle \mathsf{sk}_1||\mathsf{sk}_1', \mathbf{y}||\mathbf{y}' \rangle; [r]_2) \\ &= \underline{\mathsf{Enc}(\mathsf{sk}_2^0, -\langle \mathsf{sk}_1, \mathbf{y} \rangle; [r]_2)} + \mathsf{Enc}(\mathsf{sk}'^0_2, -\langle \mathsf{sk}_1', \mathbf{y}' \rangle; [r]_2) \end{aligned}$$

Note that the ciphertexts ct_x and $\mathsf{ct}_{x'}$ share the same randomness, thus one can write $\mathsf{ct}_x||\mathsf{ct}_{x'} = \mathsf{ct}_{x||x'}$ (similarly $\mathsf{ct}_y||\mathsf{ct}_{y'} = \mathsf{ct}_{y||y'}$). Putting to gather we get the ciphertext and functional key for $\mathbf{x}||\mathbf{x}'$ and $\mathbf{y}||\mathbf{y}'$ generated by FH associated with the master key $\mathsf{msk}'' = (\mathsf{msk}, \mathsf{msk}') = (\mathsf{sk}_1||\mathsf{sk}_1', \mathsf{sk}_2^0 + \mathsf{sk}'^0_2, \mathsf{sk}_2^1||\mathsf{sk}'^1_2)$.

Now we are ready to prove many-security of our 2-MCQFE′ construction.

Theorem 6. *If dFH is secure, 2-input QFE is many-sel-*IND *secure, and 2-MCQFE is one-sel-secure, our suggested construction 2-*MCQFE′ *is many-sel-secure.*

Proof. We proceed via a sequence of the games, the first and the last games are the real games respectively for $b = 0$ and $b = 1$. Fig. 13 summarizes the required games. In the ciphertexts of our many-secure 2-MCQFE (Eq. (11)), we define $X_{i_k,\ell} = (\mathbf{x}_{i_k,\ell}||\mathbf{0}^{2k})$ and $Y_{j_t,\ell} = (\mathbf{y}_{j_t,\ell}||\mathbf{0}^{2k})$ as the messages associated with the label ℓ.

G_0: is the real game associated with $b = 0$.

G_1: is similar to the previous game, except that in $X_{i_k,\ell}$, the vector $\mathbf{0}^{2k}$ is replaced with $A^0_{i,\ell} = \mathbf{x}^0_{i_1,\ell}||\mathbf{x}^0_{i_1,\ell}$ and in $Y_{j_k,\ell}$, the vector $\mathbf{0}^{2k}$ is replaced with $B^0_{j_t,\ell} = -\mathbf{y}^0_{j_1,\ell}||\mathbf{y}^0_{j_1,\ell}$. One can verifies that:

$$A^0_{i,\ell}B^0_{j_t,\ell'} = 0,\ \forall i, j, \ell, \ell'$$

Thus by relying on the security of dFH, games G_0 and G_1 are indistinguishable.

G_2: is similar to the previous game, except that, we replace the last $\mathbf{x}^0_{i_1,\ell}$ (in $A^0_{i,\ell}$) with $\mathbf{x}^1_{i_1,\ell}$ and correspondingly $\mathbf{y}^0_{i_1,\ell}$ with $\mathbf{y}^1_{i_1,\ell}$. Games G_1 and G_2 are indistinguishable relying on the one-security of 2-MCQFE. The simulator can simulate the encryption queries by Property 3.

G_3: is similar to the previous game, except that, we define $C^0_{i_k,\ell} := \mathbf{x}^0_{i_k,\ell}||\mathbf{x}^0_{i_1,\ell}$ and replace it with $C^1_{i_k,\ell}$, and also $D^0_{j_t,\ell} := \mathbf{y}^0_{j_t,\ell}|| - \mathbf{y}^0_{j_1,\ell}$ with $D^1_{j_t,\ell}$ for all labels (see Fig. 13). This transition needs a simple hybrid on the labels. In each hybrid, the constraints *Eq.* (12) allows us to rely on the security of 2-input QFE. The simulator can simulate the encryption queries by Property 3.

G_4: is similar to the game G_1 and allows us to replace, $A^1_{i,\ell}$ and $B^1_{j_t,\ell'}$ with vectors $\mathbf{0}^{2k}$ and $\mathbf{0}^{2k}$ where $A^1_{i,\ell}$ and $B^1_{j_t,\ell'}$ are defined in the game G_1. This is then the real game associated with $b = 1$. □

Game	Description	justific.
G_0	$X^0_{i_k,\ell} = \mathbf{x}^0_{i_k,\ell}\|\|\mathbf{0} \quad Y^0_{j_t,\ell} = \mathbf{y}^0_{j_t,\ell}\|\|\mathbf{0}$	real game $b = 0$
G_1	$\mathbf{x}^0_{i_k,\ell}\|\|\boxed{\mathbf{x}^0_{i_1,\ell}\|\|\mathbf{x}^0_{i_1,\ell}}$ $\mathbf{y}^0_{j_t,\ell}\|\|\boxed{-\mathbf{y}^0_{j_1,\ell}\|\|\mathbf{y}^0_{j_1,\ell}}$	dFH
G_2	$\mathbf{x}^0_{i_k,\ell}\|\|\mathbf{x}^0_{i_1,\ell}\|\|\boxed{\mathbf{x}^1_{i_1,\ell}}$ $\mathbf{y}^0_{j_t,\ell}\|\| - \mathbf{y}^0_{j_1,\ell}\|\|\boxed{\mathbf{y}^1_{j_1,\ell}}$	one-security 2-MCQFE
G_3	$\boxed{\mathbf{x}^1_{i_k,\ell}\|\|\mathbf{x}^1_{i_1,\ell}}\|\|\mathbf{x}^1_{i_1,\ell}$ $\boxed{\mathbf{y}^1_{j_t,\ell}\|\| - \mathbf{y}^1_{j_1,\ell}}\|\|\mathbf{y}^1_{j_1,\ell}$	2-input QFE
G_4	$X^1_{i_k,\ell} = \mathbf{x}^1_{i_k,\ell}\|\|\boxed{\mathbf{0}} \quad Y^1_{j_t,\ell} = \mathbf{y}^1_{j_t,\ell}\|\|\boxed{\mathbf{0}}$	dFH, real game $b = 1$

Fig. 13: The sequence of games for many-secure 2-MCQFE.

Acknowledgments.

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme FENTEC (Grant Agreement no. 780108), by the European Union's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud), and by the French FUI project ANBLIC.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_13
2. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_19
3. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_33
4. Abdalla, M., Bourse, F., Marival, H., Pointcheval, D., Soleimanian, A., Waldner, H.: Multi-client inner-product functional encryption in the random-oracle model. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 525–545. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_26
5. 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.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_20
6. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_12
7. 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.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_3
8. Barbosa, M., Catalano, D., Soleimanian, A., Warinschi, B.: Efficient function-hiding functional encryption: From inner-products to orthogonality. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 127–148. Springer, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_7
9. Benhamouda, F., Joye, M., Libert, B.: A new framework for privacy-preserving aggregation of time-series data. ACM Trans. Inf. Syst. Secur. **18**(3), 10:1–10:21 (2016). https://doi.org/10.1145/2873069, `https://doi.org/10.1145/2873069`
10. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_30
11. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_13
12. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_24

13. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018), https://eprint.iacr.org/2018/1021
14. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
15. Fan, X., Tang, Q.: Making public key functional encryption function private, distributively. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 218–244. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_8
16. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.13
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.) PKC 2020, Part I. LNCS, vol. 12110, pp. 95–120. Springer, Heidelberg (May 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.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_32
19. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488678
20. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_11
21. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_25
22. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180418, available as Cryptology ePrint Archive Report 2006/309
23. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_9
24. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_4
25. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_18
26. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 599–629. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7_20
27. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: Dinur, I. (ed.) 57th FOCS. pp. 11–20. IEEE Computer Society Press (Oct 2016). https://doi.org/10.1109/FOCS.2016.11
28. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_11
29. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (Aug 1984)