# From Single-Input to Multi-Client Inner-Product Functional Encryption

Michel Abdalla[1,2], Fabrice Benhamouda[3], and Romain Gay[4]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
michel.abdalla@ens.fr
[2] INRIA, Paris, France
[3] Algorand Foundation, New York, NY, USA
fabrice.benhamouda@normalesup.org
[4] Cornell Tech, New York, USA
romain.gay@cornell.edu

**Abstract.** We present a new generic construction of multi-client functional encryption (MCFE) for inner products from single-input functional inner-product encryption and standard pseudorandom functions. In spite of its simplicity, the new construction supports labels, achieves security in the standard model under adaptive corruptions, and can be instantiated from the plain DDH, LWE, and Paillier assumptions. Prior to our work, the only known constructions required discrete-log-based assumptions and the random-oracle model. Since our new scheme is not compatible with the compiler from Abdalla et al. (PKC 2019) that decentralizes the generation of the functional decryption keys, we also show how to modify the latter transformation to obtain a decentralized version of our scheme with similar features.

1 Introduction . . . 1
2 Definitions and Security Models . . . 4
  2.1 Multi-Client Functional Encryption . . . 5
  2.2 Decentralized Multi-Client Functional Encryption . . . 7
  2.3 Inner-Product Functionality . . . 9
  2.4 Pseudorandom Functions (PRF) . . . 9
  2.5 Symmetric-Key Encryption (SE) . . . 10
3 MCFE from Public-Key Single-Input FE . . . 10
  3.1 Construction . . . 10
  3.2 Static Security . . . 13
  3.3 Adaptive Security . . . 16
4 From pos$^+$-IND to any-IND Security . . . 21
5 Decentralized Multi-Client Function Encryption . . . 24
Acknowledgments . . . 26

## 1 Introduction

Functional encryption [BSW11, O'N10] is a generalization of standard encryption which allows for a more fine-grained control over the decryption capabilities of third parties. In these schemes, the owner of a master secret key can derive secret keys for specific functions via a key derivation algorithm. Then, given the encryption of a message $x$, the holder of a secret decryption key $\mathsf{sk}_f$ for a function $f$ can compute $f(x)$ using the decryption algorithm. Informally, a FE scheme is deemed secure if it is infeasible for an adversary to learn any information about $x$ other than what it can be computed using the secret keys at its disposal.

Multi-input functional encryption [GGG+14] is an extension of the functional encryption in which the function can be computed over several different inputs that can be encrypted independently. More precisely, the decryption algorithm of such schemes takes as input a secret key $\mathsf{sk}_f$ for a function $f$ together with $n$ different ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)$ and outputs the value of the function $f$ applied to underlying plaintexts $(x_1, \ldots, x_n)$.

In the setting in which each ciphertext of a multi-input functional encryption scheme is generated by a different party or client $P_i$. we often refer to these schemes as multi-client functional encryption (MCFE) schemes [CDG+18, GGG+14]. In this setting, it is natural to assume that the adversary can corrupt these parties and learn their secret encryption keys. The master secret key, however, is still assumed to be owned by a trusted third party.

Another important property of multi-client functional encryption considered by Chotard et al. [CDG+18] is the inclusion of labels in the encryption process. More precisely, in a labeled MCFE scheme, the individual encryption algorithms each take a label as an additional parameter and decryption should only be possible when using ciphertexts generated with respect to the same label. That is, labels allow the users to have more control over the mix-and-match capabilities, as opposed to MCFE without labels, where the owner of a functional decryption key can mix and match all the ciphertexts.

Note that labels can be obtained without loss of generality for MCFE for all functions; however, this is not the case of the practical constructions for restricted classes of functions, such as inner products, which is the focus of this paper. Reciprocally, any MCFE with labels can be turned into a label-free MCFE for the same functionality, simply by setting the labels used by the encryption algorithm to be always a fixed value $\perp$. Put simply, labels are an extra feature that offers a better control over the information leaked by each generated functional decryption key.

For instance, suppose we want to use MCFE to allow teachers to grade their students in a way that the students can use these grades in different college applications and that colleges only learn the average grades of the students with weights of their choice. In this scenario, each teacher would encrypt the grade of each student for their subject. Each college would have a functional decryption key to compute the weighted average of all the grades of each student. It is very important that the teachers use the student ID as a label, otherwise colleges would be able to compute weighted average of a mix of multiple students (like Maths from student A and Physics from student B), which significantly hinders privacy.

**Prior work**. As remarked in [AGRW17], most of the prior work in the multi-input setting are either feasibility results for general functionalities (e.g,. [GGG+14,BGJS15,AJ15,BKS18]) or efficient constructions for particular functionalities (e.g.,[AGRW17, ACF+18, CDG+18, DOT18, **?**, ABKW19]). In the latter case, which is the setting in which we are interested in this paper, the main functionality under consideration is the inner-product functionality, in which functions are associated to a collection $\boldsymbol{y}$ of $n$ vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. In particular, on input a collection $\boldsymbol{x}$ of $n$ vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, it outputs $f_{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{i=1}^{n} \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. As noted in prior works [ABDP15, AGRW17, CDG+18], inner-product functionalities can be quite useful for computing statistics or performing data mining on encrypted databases.

Among the constructions of multi-input functional inner-product encryption schemes without labels, the work of Abdalla et al. in [ACF+18] is the one requiring the weakest assumptions since it can be built from any single-input functional encryption scheme satisfying some mild properties (recalled in Section 3). In particular, by instantiating it with the public-key functional inner-product encryption schemes in [ALS16], one can obtain constructions based on the DDH, Paillier, and LWE assumptions. Moreover, as recently shown in [ABKW19], their schemes remain secure even when the secret encryption keys can be adaptively corrupted by the adversary. Unfortunately, as we further discuss below, we do not know how to generalize the ACFGU scheme to the labeled setting. In fact, the construction from [ACF+18] relies on an information-theoretic multi-input FE (as they put it, the functional encryption equivalent of a one-time pad) to obtain security in the restricted context of one challenge ciphertext per input slot. Then, they bootstrap security to many challenge ciphertexts using an extra layer of single-input FE. That information-theoretic approach cannot be emulated, since we need to hide messages for arbitrarily many labels in our case. Thus, an entropy argument can be used to show that we need to resort to a computational assumption, even for proving security in the context of one challenge ciphertext per input slot and label. In our case, we use PRFs.

Among the constructions of multi-input functional inner-product encryption schemes with labels, the works of Abdalla et al. [ABKW19] and Chotard et al. [**?**] currently represent the state of the art in this area. In particular, both schemes provide labeled MCFE schemes in the random-oracle model in discrete-log-based groups. The main advantage of the work of Chotard et al. is that its ciphertexts are shorter and that it allows for multiple ciphertexts under the same label. However, it requires pairing groups. The main advantage of the work of Abdalla et al. is that it can be instantiated in pairing-free groups. However, its ciphertexts are longer and it only allows for one ciphertext per label, a restriction inhereted from [CDG+18]. As in the case of other discrete-log-based constructions of functional inner-product encryption schemes (e.g, [ABDP15, BJK15, ALS16, AGRW17]), the size of supported messages is restricted for both schemes since the decryption algorithm needs to compute discrete logarithms.

**Contributions**. In order to address the shortcomings of previous labeled MCFE schemes, the main contribution of this paper is to provide the first construction of labeled MCFE schemes in the standard model from more general assumptions than discrete-logarithm-based ones. As in the work of Abdalla et al. in [ACF+18], our constructions can be built from any single-input public-key functional encryption scheme satisfying some mild properties (recalled in Section 3). In particular, by instantiating it with the schemes in [ALS16], one can obtain constructions based on the DDH, Paillier, and LWE assumptions. Our constructions have no restriction on the number of ciphertexts per label and are proven secure with respect to adaptive corruptions.

In order to achieve our main result, our security proof proceeds in two parts. First, we prove the security of our MCFE scheme in a setting in which the adversary is required to query the encryption oracle in all $n$ positions for each label. Then, in a second step, we apply the compiler suggested in [ABKW19] to remove this requirement. Since the proof for the latter transformation given in [ABKW19] is in the random-oracle model, an additional contribution of our work is to provide an alternative proof for it in Section 4 which does not require random oracles.

Finally, since our main construction is not compatible with the transformation from [ABKW19] that decentralizes the generation of the functional decryption keys, we also show how to modify the latter to obtain a decentralized version of our scheme with similar features. As a result, we obtain the first decentralized labeled MCFE schemes in the standard model based on the DDH, Paillier, and LWE assumptions.

**Independent work**. In a recent work [**?**], the authors define multi-input functional encryption schemes with decentralized key generation and setup, in which users can join the system dynamically. They give a feasibility result for general functions, and also provide a construction for inner products, from a standard assumption (LWE). However, their construction does not handle labels.

**Overview of our construction.** Following the proof strategy first used in [AGRW17] in the context of multi-input FE for inner products, we start with a scheme whose security only holds when there is only one challenge ciphertext per input slot. The novelty compared to multi-input FE is that we have to handle arbitrarily many labels, even if there is only one challenge ciphertext per slot and label.

*One-time security with labels.* We modify the scheme from [ACF+18], where the one-time secure MIFE is simply obtained using a one-time pad of the messages. The functional decryption keys are simply the linear combination of these pads. Namely, for any input slot $i$, we have $\mathsf{ct}_i := \boldsymbol{x}_i + \boldsymbol{t}_i$, and for $\mathsf{sk}_{\boldsymbol{y}} := \sum_{i=1}^{n} \langle \boldsymbol{t}_i, \boldsymbol{y}_i \rangle$, where $\boldsymbol{t}_i \leftarrow \mathbb{Z}_L^m$, $m$ denotes the dimension of individual messages $\boldsymbol{x}_i$, and everything is computed modulo $L$, for some specified integer $L$. Here, we write $\boldsymbol{y} := (\boldsymbol{y}_1 \| \ldots \| \boldsymbol{y}_n)$, the concatenation of $n$ vectors, each of dimension $m$. To decrypt the set of ciphertext $\{\mathsf{ct}_i\}_i$, one simply compute $\sum_i \langle \mathsf{ct}_i, \boldsymbol{y}_i \rangle$, and subtract by the key $\mathsf{sk}_{\boldsymbol{y}}$ to get $\sum_i \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle$. Security follows by a perfect statistical argument.

The technical challenge is to emulate this idea to a setting where ciphertexts can be generated for many labels. Since the number of label is not a priori bounded, we cannot resort to a perfectly statistical argument: the master secret key (which in the previous scheme contains all the vectors $\boldsymbol{t}_i$) is simply too small to contain all possible pads $\boldsymbol{t}_{i,\ell}$ for all labels $\ell \in \mathsf{Labels}$ that would required to perform such an argument. We must resort to a computation argument. A natural but flawed idea would to generate the pads $\boldsymbol{t}_{i,\ell}$ using a PRF

applied on a label $\ell \in \mathsf{Labels}$. This approach faces two issues: first, if one slot is corrupted, then the security of the entire system is compromised, since each input slot needs the PRF key to encrypt. Second, since the labels are only known at encryption time, the generation of functional decryption keys is unable to produce the value $\sum_i \langle \boldsymbol{y}_i, \boldsymbol{t}_{i,\ell} \rangle$.

To circumvent these issues we generate the pads $\boldsymbol{t}_{i,\ell} := \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$, where for all $i < j \in [n]$, the keys $\mathsf{K}_{i,j} \leftarrow \{0,1\}^\lambda$, and $\mathsf{K}_{j,i} = \mathsf{K}_{i,j}$, and $(-1)^{j<i}$ denotes $-1$ if $j < i$, 1 otherwise. This construction has first been used in [KDK11] to decentralize the computation of the sum of private values in a non-interactive way. Each input slot $i \in [n]$ needs the set of keys $\{\mathsf{K}_{i,j}\}_{j \in [n]}$ to encrypt. Assuming the security of the PRF, it produces pseudorandom pads, which will be able to mask the messages $\boldsymbol{x}_i$ simultaneously for all used label $\ell \in \mathsf{Labels}$. Thus, we prove that this holds even when some users $i \in [n]$ are corrupted (in fact, up to $n-2$ can be corrupted). This solves the first issue mentioned above. To solve the second issue, namely, ensuring correctness holds for all possible labels, we use the structure property that holds for all label $\ell \in \mathsf{Labels}$: $\sum_{i \in [n]} \boldsymbol{t}_{i,\ell} = \mathbf{0}$, where $\mathbf{0}$ denotes the zero vector. Otherwise stated, these pads are shares of a perfect $n$ out of $n$ secret sharing of $\mathbf{0}$. We use this by setting the ciphertext for slot $i \in [n]$ and label $\ell \in \mathsf{Labels}$ to be an encryption of the vector $\boldsymbol{w}_{i,\ell} := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i \| \mathbf{0} \| \ldots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell} \in \mathbb{Z}_L^{mn}$. This way, we have $\langle \boldsymbol{w}_{i,\ell}, \boldsymbol{y} \rangle = \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle + \langle \boldsymbol{t}_{i,\ell}, \boldsymbol{y} \rangle$ for all slots $i \in [n]$, therefore: $\sum_{i \in [n]} \langle \boldsymbol{w}_{i,\ell}, \boldsymbol{y} \rangle = \sum_{i \in [n]} \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle$. The last step is to encrypt the vector $\boldsymbol{w}_{i,\ell}$ using any single-input, public-key FE for inner products. The functional decryption key is simply the functional decryption key of the single-input inner-product FE for the associated vector $\boldsymbol{y}$. Correctness is preserved, since the decryption only needs to compute the inner product between $\boldsymbol{w}_{i,\ell}$ and $\boldsymbol{y}$.

*Full-fledged security.* To obtain security with many challenge ciphertexts per input slot and label, we use similar techniques to those used in [ACF+18] in the context of multi-input inner-product FE. However, these can only be applied when the adversary does not make use of the information revealed by partial ciphertexts $\{\mathsf{ct}_{i,\ell}\}_{i \in [n] \setminus \{\text{missing}\}}$, where $\{\text{missing}\}$ denotes the set of missing slots for label $\ell$. Prior works [**?**,ABKW19] provides generic compilers that precisely avoid partial ciphertexts to leak any information about the underlying plaintext (decryption is only successful when ciphertexts for all slots are present), but they are only proven secure in the random oracle model, and for [**?**], use additional assumptions (pairings). Since our focus it to build simple MCFE schemes from weak assumptions, we give a new generic transformation (in Section 4) that avoids the leakage of information of partial ciphertexts, with no extra assumption (only PRFs, in the standard model), and that handles adaptive corruptions.

*Decentralizing MCFE.* In order to decentralize the generation of functional decryption keys, we adapt the construction from [ABKW19]. The main idea is to secret share the master secret key, since computing the functional secret key is a linear operation, it can be done non-interactively from these shares.

*Outline.* The rest of the paper is organized as follows. After giving the relevant technical preliminaries and definitions in Section 2, we give our new construction of MCFE from single-input FE for inner products in Section 3. In Section 4, we show how to generically strengthen the security of our MCFE construction, thereby removing any artificial restrictions on the security model. Finally, in Section 5, we show how to decentralize our MCFE to obtain a DMCFE.

## 2 Definitions and Security Models

**Notation.** We use $[n]$ to denote the set $\{1, \ldots, n\}$. We write $\boldsymbol{x}$ for vectors and $x_i$ for the $i$-th element. For security parameter $\lambda$ and additional parameters $n$, we denote the winning probability of an adversary $\mathcal{A}$ in a game or experiment $\mathsf{G}$ as $\mathsf{Win}^{\mathsf{G}}_{\mathcal{A}}(\lambda, n)$, which is $\Pr[\mathsf{G}(\lambda, n, \mathcal{A}) = 1]$. The probability is taken over the random coins of $\mathsf{G}$ and $\mathcal{A}$.

### 2.1 Multi-Client Functional Encryption

In this section, we recall the definition of MCFE [GGG+14]. It is taken almost verbatim from [ABKW19], with the following differences: the use of a stronger security definition (see Remark 2.3) and the introduction of a master public key $\mathsf{mpk}$, so that *public-key* functional encryption becomes a particular case of MCFE.

**Definition 2.1.** ***(Multi-Client Functional Encryption)*** *Let $\mathcal{F} = \{\mathcal{F}_\rho\}_\rho$ be a family (indexed by $\rho$) of sets $\mathcal{F}_\rho$ of functions $f\colon \mathcal{X}_{\rho,1} \times \cdots \times \mathcal{X}_{\rho,n_\rho} \to \mathcal{Y}_\rho$.*[5] *Let $\mathsf{Labels} = \{0,1\}^*$ or $\{\perp\}$ be a set of labels. A multi-client functional encryption scheme (MCFE) for the function family $\mathcal{F}$ and the label set $\mathsf{Labels}$ is a tuple of five algorithms $\mathsf{MCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$:*

$\mathsf{Setup}(1^\lambda, 1^n)$**:** *Takes as input a security parameter $\lambda$ and the number of parties $n$, and generates public parameters $\mathsf{pp}$. The public parameters implicitly define an index $\rho$ corresponding to a set $\mathcal{F}_\rho$ of $n$-ary functions (i.e., $n = n_\rho$).*

$\mathsf{KeyGen}(\mathsf{pp})$**:** *Takes as input the public parameters $\mathsf{pp}$ and outputs $n$ secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$, a master secret key $\mathsf{msk}$, and a master public key $\mathsf{mpk}$.*

$\mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, f)$**:** *Takes as input the public parameters $\mathsf{pp}$, the master secret key $\mathsf{msk}$ and a function $f \in \mathcal{F}_\rho$, and outputs a functional decryption key $\mathsf{sk}_f$.*

$\mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_i, x_i, \ell)$**:** *Takes as input the public parameters $\mathsf{pp}$, a master public key $\mathsf{mpk}$, a secret key $\mathsf{sk}_i$, a message $x_i \in \mathcal{X}_{\rho,i}$ to encrypt, a label $\ell \in \mathsf{Labels}$, and outputs ciphertext $\mathsf{ct}_{i,\ell}$.*

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell})$**:** *Takes as input the public parameters $\mathsf{pp}$, a functional key $\mathsf{sk}_f$ and $n$ ciphertexts under the same label $\ell$ and outputs a value $y \in \mathcal{Y}_\rho$.*

*A scheme $\mathsf{MCFE}$ is correct, if for all $\lambda, n \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $f \in \mathcal{F}_\rho$, $\ell \in \mathsf{Labels}$, $x_i \in \mathcal{X}_{\rho,i}$, when $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and $\mathsf{sk}_f \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, f)$, we have for $\boldsymbol{x} = (x_1, \ldots, x_n)$:*

$$\Pr\left[\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_1, x_1, \ell), \ldots, \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_n, x_n, \ell)) = f(\boldsymbol{x})\right] = 1.$$

When $\rho$ is clear from context, the index $\rho$ is omitted. Note that the case of (single-input) functional encryption as defined in [BSW11, O'N10] corresponds to the case $n = 1$, and $\mathsf{Labels} = \{\perp\}$. For such schemes, we also consider the *public-key* variant, where $\mathsf{sk}_1 = \perp$, that is, the encryption algorithm only requires the public parameters $\mathsf{pp}$ and the master public key $\mathsf{mpk}$ to encrypt the message $x_1$. In this setting, $\mathsf{sk}_1$ is omitted.

Except for public-key single-input functional encryption, the master public-key can be included in each secret key $\mathsf{sk}_i$ and we omit it.

We follow the notation of [ABKW19] here, where the algorithm $\mathsf{Setup}$ only generates public parameters that determine the set of functions for which functional decryption keys can be created, and the secret/encryption keys and the master secret keys are generated by another algorithm $\mathsf{KeyGen}$, while the functional decryption keys are generated by $\mathsf{KeyDer}$.

In the following, we define security as adaptive left-or-right indistinguishability under both static (sta), and adaptive (adt) corruption. We also consider two variants of these notions $(\mathrm{any}, \mathrm{pos}^+)$ related to the number of encryption queries asked by the adversary for each slot.

**Definition 2.2.** ***(Security of MCFE)*** *Let $\mathsf{MCFE}$ be an MCFE scheme, $\mathcal{F} = \{\mathcal{F}_\rho\}_\rho$ a function family indexed by $\rho$ and $\mathsf{Labels}$ a label set. For $\mathrm{xx} \in \{\mathrm{sta}, \mathrm{adt}\}$, $\mathrm{yy} \in \{\mathrm{any}, \mathrm{pos}^+\}$, and $\beta \in \{0,1\}$, we define the experiment $\mathrm{xx\text{-}yy\text{-}IND}_\beta^{\mathsf{MCFE}}$ in Fig. 1, where the oracles are defined as:*

**Corruption oracle** $\mathsf{QCor}(i)$**:** *Outputs the encryption key $\mathsf{sk}_i$ of slot $i$. We denote by $\mathcal{CS}$ the set of corrupted slots at the end of the experiment.*

**Left-Right oracle** $\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$**:** *Outputs $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, x_i^\beta, \ell)$ on a query $(i, x_i^0, x_i^1, \ell)$. We denote by $Q_{i,\ell}$ the number of queries of the form $\mathsf{QLeftRight}(i, \cdot, \cdot, \ell)$.*

**Encryption oracle** $\mathsf{QEnc}(i, x_i, \ell)$**:** *outputs $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_i, x_i, \ell)$ on a query $(i, x_i, \ell)$.*

[5] All the functions inside the same set $\mathcal{F}_\rho$ have the same domain and the same range.

**Key derivation oracle** $\mathsf{QKeyD}(f)$**:** *Outputs* $\mathsf{sk}_f = \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, f)$.

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

- *If* $i \in \mathcal{CS}$ *(i.e., slot* $i$ *is corrupted): for any query* $\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$, $x_i^0 = x_i^1$.[6]
- *For any label* $\ell \in \mathsf{Labels}$*, for any family of queries* $\{\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$ *or* $\mathsf{QEnc}(i, x_i, \ell)\}_{i \in [n] \setminus \mathcal{CS}}$*, for any family of inputs* $\{x_i \in \mathcal{X}_{\rho,i}\}_{i \in \mathcal{CS}}$*, for any query* $\mathsf{QKeyD}(f)$*, we define* $x_i^0 := x_i$ *and* $x_i^1 := x_i$ *for any slot* $i \in \mathcal{CS}$ *and any slot queried to* $\mathsf{QEnc}(i, x_i, \ell)$*, and we require that:*

$$f(\boldsymbol{x}^0) = f(\boldsymbol{x}^1) \qquad \text{where } \boldsymbol{x}^b = (x_1^b, \ldots, x_n^b) \text{ for } b \in \{0,1\} \ .$$

  *We insist that if one index* $i \notin \mathcal{CS}$ *is not queried for the label* $\ell$*, there is no restriction.*
- *When* $\mathrm{yy} = \mathrm{pos}^+$*: for any slot* $i \in [n]$ *and* $\ell \in \mathsf{Labels}$*, if* $Q_{i,\ell} > 0$*, then for any slot* $j \in [n] \setminus \mathcal{CS}$*,* $Q_{j,\ell} > 0$*. In other words, for any label, either the adversary makes no left-right encryption query or makes at least one left-right encryption query for each slot* $i \in [n] \setminus \mathcal{CS}$*.*

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

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

*A multi-client functional encryption scheme* $\mathsf{MCFE}$ *is* xx-yy-*IND secure, if for any* $n$*, for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\text{xx-yy-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq \mathrm{negl}(\lambda)$.

We omit $n$ when it is clear from the context. We also often omit $\mathcal{A}$ from the parameter of experiments or games when it is clear from the context.

*Remark 2.3 (The role of the oracle* $\mathsf{QEnc}$*).* The security definitions we give are slightly stronger than those given in [ABKW19], since the oracle $\mathsf{QEnc}$ gives out information that is not captured by Condition (*), for $\mathrm{pos}^+$, hence the use of the notation $\mathrm{pos}^+$ instead of pos in [ABKW19]. For any, this addition of $\mathsf{QEnc}$ has no effect, as $\mathsf{QEnc}$ queries can be simulated using $\mathsf{QLeftRight}$. But for $\mathrm{pos}^+$/pos, there is no equivalence in general between the security definition with and without the encryption oracle. We add this oracle $\mathsf{QEnc}$ so that we can reduce the security with respect to one label to the security with respect to multiple queried labels, via a simple hybrid argument (which would not be valid without the $\mathsf{QEnc}$ oracle), as done in [**?**]. This will be used in our generic compiler from $\mathrm{pos}^+$ to any security, in Section 4.

Now we define a seemingly weaker security notion than xx-yy-IND, which we call xx-yy-IND-1-label, since the adversary is restricted to query the oracle $\mathsf{QLeftRight}$ on at most one label, and it cannot query the oracle $\mathsf{QEnc}$ oracle on that label. Using a standard hybrid argument (cf Lemma 2.5), we show that this is equivalent to the original xx-yy-IND security defined above. These restrictions will make the proofs easier in the rest of the paper.

**Definition 2.4.** ***(1-label Security)*** *Let* $\mathsf{MCFE}$ *be an MCFE scheme,* $\mathcal{F} = \{\mathcal{F}_\rho\}_\rho$ *a function family indexed by* $\rho$ *and* $\mathsf{Labels}$ *a label set. For* $\mathrm{xx} \in \{\mathrm{sta}, \mathrm{adt}\}$*,* $\mathrm{yy} \in \{\mathrm{any}, \mathrm{pos}^+\}$*, and* $\beta \in \{0,1\}$*, we define the experiment* $\text{xx-yy-IND-1-label}^{\mathsf{MCFE}}_\beta$ *exactly as in Fig. 1, where the oracles are defined as for Definition 2.2, except:*

**Left-Right oracle** $\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$**:** *Outputs* $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, x_i^\beta, \ell)$ *on a query* $(i, x_i^0, x_i^1, \ell)$*. This oracle can be queried at most on one label. Further queries with distinct labels will be ignored.*

**Encryption oracle** $\mathsf{QEnc}(i, x_i, \ell)$**:** *outputs* $\mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \mathsf{sk}_i, x_i, \ell)$ *on a query* $(i, x_i, \ell)$*. If this oracle is queried on the same label that is queried to* $\mathsf{QLeftRight}$*, the game ends and return* $0$.

[6] We could define a stronger security notion without this restriction. However, in this paper, as in the prior works on MCFE, we add this restriction. In particular, we allow the secret key for the slot $i$ to decrypt ciphertexts for the slot $i$. We leave achieving stronger security as an interesting open problem.

Condition (*) *is defined as for Definition 2.2.*

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

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

**Lemma 2.5 (From one to many labels).** *Let* MCFE *be a scheme that is* xx-yy*-IND-1-label secure, for* $\text{xx} \in \{\text{sta}, \text{adt}\}$ *and* $\text{yy} \in \{\text{pos}^+, \text{any}\}$*. Then it is also secure against PPT adversaries that query* QLeftRight *on many distinct labels (*xx-yy*-IND security). Namely, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that:*

$$\mathsf{Adv}^{\text{xx-yy-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq q_{\mathsf{Enc}} \cdot \mathsf{Adv}^{\text{xx-yy-IND-1-label}}_{\mathsf{MCFE},\mathcal{B}}(\lambda, n),$$

*where* $\mathsf{Adv}^{\text{xx-yy-IND-1-label}}_{\mathsf{MCFE},\mathcal{B}}(\lambda, n)$ *denotes the advantage of $\mathcal{B}$ against an experiment defined as above, except* QLeftRight *can be queried on at most one label and* QEnc *must not be queried on that label. By $q_{\mathsf{Enc}}$ we denote the number of distinct labels queried by $\mathcal{A}$ to* QLeftRight *in the original security game.*

*Proof (Sketch).*

First, let us consider the case of yy = any security. The proof uses a hybrid argument which goes over all the labels $\ell_1, ...\ell_Q$ queried to both the oracles QEnc and QLeftRight. In the $k$'th hybrid, the queries for the first $k$'th labels to the QLeftRight oracle are answered with the right plaintext, and the the last $Q - k$ labels are answered with the left plaintext. To go from hybrid $k - 1$ to $k$, $\mathcal{B}$ uses its own QEnc oracle to answer $\mathcal{A}$'s queries to QLeftRight for labels $\ell_j$ for $j < k$, and $j > k$ (using the right and left plaintext respectively), and uses its own oracle QLeftRight for label $\ell_k$. The queries made by $\mathcal{A}$ to QEnc and QCor are answered straightforwardly by $\mathcal{B}$ from its own oracles. Note that the queries made by $\mathcal{B}$ satisfy the 1-label restriction, since QLeftRight is only queried on $\ell_k$, and QEnc is not queried on $\ell_k$.

For the case of yy = $\text{pos}^+$ security, to go from hybrid $k - 1$ to $k$, $\mathcal{B}$ uses the QEnc oracle to answer QLeftRight queries for labels $\ell_j$ for $j < k$ and $j > k$ (using the right and left plaintext respectively). For the label $\ell_k$, $\mathcal{B}$ uses its own oracle QLeftRight to answer $\mathcal{A}$'s queries to both QLeftRight and QEnc. So far, the reduction works as for the case of yy = any security. However, the difference is yy = $\text{pos}^+$ security requires additional conditions on the queries made to QLeftRight, in particular, if one honest slot is queried to QLeftRight for $\ell_k$, then all honest slots should be queried. Thus, we need to distinguish two cases: case 1) $\ell_k$ is queried to QEnc, but never on QLeftRight, in which case $\mathcal{B}$ uses its own QEnc oracle; case 2) $\ell_k$ is queried to QLeftRight at some point (and by definition of $\text{pos}^+$ security, that means it's queried to all honest slots). In case 2, the queries of $\mathcal{B}$ to QLeftRight will satisfy the condition required by the yy = $\text{pos}^+$ security game, namely, if QLeftRight is queried on $\ell_k$ for some honest input slot, then it has to be queried on the same label $\ell_k$ for all honest input slots. Note that this restriction doesn't apply to the queries made to QEnc. In case 1, we use the fact that the two hybrid games $k - 1$ and $k$ are exactly the same. Therefore, at the end of the simulation, $\mathcal{B}$ checks whether case 1 occurs, and if it does, simply outputs 0 to its own experiment, ignoring $\mathcal{A}$'s output. Otherwise, it means it is case 2, and $\mathcal{B}$ forwards the output from $\mathcal{A}$ to its own experiment.

□

We summarize the relations between the six security notions in Fig. 2, where xx-pos-IND is the notion defined in [ABKW19] (i.e., it is like xx-$\text{pos}^+$-IND without the QEnc oracle).

### 2.2 Decentralized Multi-Client Functional Encryption

Now, we introduce the definition of decentralized multi-client functional encryption (DMCFE) [CDG+18]. As for our definition of MCFE, we separate the algorithm Setup which generates public parameters defining in particular the set of functions, from the algorithm KeyGen. We do not consider public-key variants of DMCFE and hence completely omit the master public key mpk.

$\textbf{sta-yy-IND}_{\beta}^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$

$\mathcal{CS} \leftarrow \mathcal{A}(1^\lambda, 1^n)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$

$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot),\mathsf{QKeyD}(\cdot)}(\mathsf{pp}, \mathsf{mpk}, \{\mathsf{sk}_i\}_{i\in\mathcal{CS}})$

**Output:** $\alpha$ if Condition (*) is satisfied, 0 otherwise.

$\textbf{adt-yy-IND}_{\beta}^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$

$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$

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

**Output:** $\alpha$ if Condition (*) is satisfied, 0 otherwise.

**Fig. 1.** Security games for MCFE

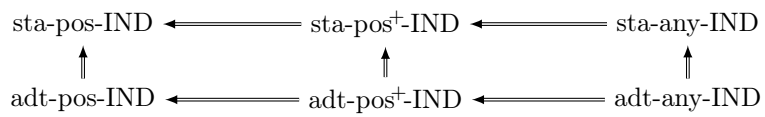


**Fig. 2.** Relations between the MCFE security notions (arrows indicate implication or being "a stronger security notion than")

**Definition 2.6.** ***(Decentralized Multi-Client Functional Encryption)*** *Let $\mathcal{F} = \{\mathcal{F}_\rho\}_\rho$ be a family (indexed by $\rho$) of sets $\mathcal{F}_\rho$ of functions $f\colon \mathcal{X}_{\rho,1} \times \cdots \times \mathcal{X}_{\rho,n_\rho} \to \mathcal{Y}_\rho$. Let $\mathsf{Labels} = \{0,1\}^*$ or $\{\perp\}$ be a set of labels. A decentralized multi-client functional encryption scheme (DMCFE) for the function family $\mathcal{F}$ and the label set $\mathsf{Labels}$ is a tuple of six algorithms $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDerShare}, \mathsf{KeyDerComb}, \mathsf{Enc}, \mathsf{Dec})$:*

$\mathsf{Setup}(1^\lambda, 1^n)$ *is defined as for MCFE in Definition 2.1.*

$\mathsf{KeyGen}(\mathsf{pp})$**:** *Takes as input the public parameters $\mathsf{pp}$ and outputs $n$ secret keys $\{\mathsf{sk_i}\}_{i\in[n]}$.*

$\mathsf{KeyDerShare}(\mathsf{pp}, \mathsf{sk}_i, f)$**:** *Takes as input the public parameters $\mathsf{pp}$, a secret key $\mathsf{sk}_i$ from position $i$ and a function $f \in \mathcal{F}_\rho$, and outputs a partial functional decryption key $\mathsf{sk}_{i,f}$.*

$\mathsf{KeyDerComb}(\mathsf{pp}, \mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f})$**:** *Takes as input the public parameters $\mathsf{pp}$, $n$ partial functional decryption keys $\mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f}$ and outputs the functional decryption key $\mathsf{sk}_f$.*

$\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, x_i, \ell)$ *is defined as for MCFE in Definition 2.1.*

$\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell})$ *is defined as for MCFE in Definition 2.1.*

*A scheme $\mathsf{DMCFE}$ is correct, if for all $\lambda, n \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $f \in \mathcal{F}_\rho$, $\ell \in \mathsf{Labels}$, $x_i \in \mathcal{X}_{\rho,i}$, when $\{\mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\mathsf{sk}_{i,f} \leftarrow \mathsf{KeyDerShare}(\mathsf{sk}_i, f)$ for $i \in [n]$, and $\mathsf{sk}_f \leftarrow \mathsf{KeyDerComb}(\mathsf{pp}, \mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f})$, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_1, x_1, \ell), \ldots, \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_n, x_n, \ell)) = f(x_1, \ldots, x_n)\right] = 1 \ .$$

We remark that there is no master secret key $\mathsf{msk}$. Furthermore, similarly to [CDG+18], our definition does not explicitly ask the setup to be decentralized. Our DMCFE construction based on DDH (Section 5) however has a setup which can be easily decentralized.

We consider a similar security definition for the decentralized multi-client scheme. We point out that contrary to [CDG+18], we do not differentiate encryption keys from secret keys. This is without loss of generality, as corruptions in [CDG+18] only allow to corrupt both keys at the same time.

**Definition 2.7.** ***(Security of DMCFE)*** *The* xx-yy-*IND security notion of an DMCFE scheme (*$\mathrm{xx} \in \{\mathrm{sta}, \mathrm{adt}\}$ *and* $\mathrm{yy} \in \{\mathrm{any}, \mathrm{pos}^+\}$*) is similar to the one of an MCFE (Definition 2.2), except that there is no master secret key* $\mathsf{msk}$ *and the key derivation oracle is now defined as:*

**Key derivation oracle** $\mathsf{QKeyD}(f, i)$**:** *Given as input* $f \in \mathcal{F}$ *and a client* $i \in [n]$*, it returns* $\mathsf{sk}_{i,f} := \mathsf{KeyDerShare}(\mathsf{pp}, \mathsf{sk}_i, f)$*.*

*Remark 2.8 (Weaker security definition for DMCFE).* Some prior works, including [ABKW19], give a weaker security definition for DMCFE, where the adversary can only get access to all the shares of functional decryption keys at once. This fails to capture the scenario where the adversary get some, but not all, shares of functional decryption keys. In that case, the adversary should not be able to recover any meaningful information. For instance, consider the following scenario: all clients except client 1 want to learn the input of client 1. Client 1 will not participate in the generation of such functional decryption key, since that would clearly violate her privacy. Thus, the scheme should not reveal any information on her encrypted value as long there are some missing shares of functional decryption keys. This is not addressed by the security definition of [ABKW19] since the adversary gets all the shares at the same time. Instead, we allow the adversary to corrupt functional decryption keys share by share, as in the security definition originally introduced in [CDG+18].

### 2.3 Inner-Product Functionality

We describe the functionalities supported by the constructions in this paper. The index of the family is defined as $\rho = (\mathcal{R}, n, m, X, Y)$ where $\mathcal{R}$ is either $\mathbb{Z}$ or $\mathbb{Z}_L$ for some integer $L$, and $n, m, X, Y$ are positive integers. If $X, Y$ are omitted, then $X = Y = L$ is used (i.e., no constraint).

This defines $\mathcal{F}^{\mathsf{ip}}_{\rho} = \{f_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n} : (\mathcal{R}^m)^n \to \mathcal{R}\}$ where

$$f_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{i=1}^{n} \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \ ,$$

where the vectors satisfy the following bounds: $\|\boldsymbol{x}_i\|_\infty < X$, $\|\boldsymbol{y}_i\|_\infty < Y$ for $i \in [n]$, and where $\boldsymbol{x} \in \mathcal{R}^{mn}$ and $\boldsymbol{y} \in \mathcal{R}^{mn}$ are the vectors corresponding to the concatenation of the $n$ vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ respectively.

### 2.4 Pseudorandom Functions (PRF)

We make use of a pseudorandom function $\mathsf{PRF}_{\mathsf{K}}(\ell)$, indexed by a key $\mathsf{K} \in \{0,1\}^\lambda$, that takes as input a label $\ell \in \mathsf{Labels}$, and outputs a value in the output space $\mathcal{Z}$. For a uniformly random key $\mathsf{K} \leftarrow \{0,1\}^\lambda$, this function is computationally indistinguishable from a truly random function from $\mathsf{Labels}$ to $\mathcal{Z}$.

**Definition 2.9 (PRF).** *For any* $\mathsf{PRF}$ *from* $\mathsf{Labels}$ *to* $\mathcal{Z}$*, any bit* $\beta \in \{0,1\}$*, any security parameter* $\lambda$*, and any adversary* $\mathcal{A}$*, we define the experiment* $\mathbf{IND}^{\mathsf{PRF}}{}_{\beta}$ *as follows.*

| $\mathbf{IND}^{\mathsf{PRF}}_{\beta}(\lambda, \mathcal{A})$ |
|---|
| $\mathsf{K} \leftarrow \{0,1\}^\lambda$ |
| $\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PRF}}(\cdot)}(1^\lambda)$ |
| **Output:** $\alpha$ |

**Fig. 3.** Security games for PRF. The oracle $\mathcal{O}_{\mathsf{PRF}}(\ell)$ returns $\mathsf{PRF}_{\mathsf{K}}(\ell)$ if $\beta = 0$; $\mathsf{RF}(\ell)$ otherwise, where $\mathsf{RF}$ denotes a random function computed on the fly.

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

$$\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) = \left| \Pr[\mathrm{IND}_0^{\mathsf{PRF}}(\lambda, \mathcal{A}) = 1] - \Pr[\mathrm{IND}_1^{\mathsf{PRF}}(\lambda, \mathcal{A}) = 1] \right| \ .$$

*A PRF is secure, if for any any polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that:* $\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)$.

### 2.5 Symmetric-Key Encryption (SE)

A symmetric encryption with key space $\mathcal{K}$ consists of the following PPT algorithms:

- $\mathsf{Enc}(K, m)$: given a symmetric key $K$ and a message $m$, outputs a ciphertext.
- $\mathsf{Dec}(K, \mathsf{ct})$: given a symmetric key $K$ and a ciphertext $\mathsf{ct}$, outputs a message (or $\perp$ if it fails to decrypt).

For all message in the message space, we have $\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$, where the probability is taken over the random choice of $K \leftarrow \mathcal{K}$. We say a symmetric-key encryption with key space $\mathcal{K}$ is compatible with a PRF with output space $\mathcal{Z}$ if $\mathcal{K} = \mathcal{Z}$.

**Definition 2.10 (SE).** *For any* $\mathsf{SE}$ *with key space $\mathcal{K}$, any bit $\beta \in \{0, 1\}$, any security parameter $\lambda$, and any adversary $\mathcal{A}$, we define the experiment* $\mathbf{IND}^{\mathsf{PRF}}{}_{\beta}$ *as follows.*

| $\mathbf{IND}_{\beta}^{\mathsf{SE}}(\lambda, \mathcal{A})$ |
|---|
| $\mathsf{K} \leftarrow \mathcal{K}$ |
| $\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{SE}}(\cdot)}(1^{\lambda})$ |
| **Output:** $\alpha$ |

**Fig. 4.** Security games for SE. The oracle $\mathcal{O}_{\mathsf{SE}}(m_0, m_1)$ returns $\mathsf{Enc}(K, m_\beta)$.

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

$$\mathsf{Adv}_{\mathsf{SE},\mathcal{A}}(\lambda, n) = \left| \Pr[\mathrm{IND}_0^{\mathsf{PRF}}(\lambda, \mathcal{A}) = 1] - \Pr[\mathrm{IND}_1^{\mathsf{SE}}(\lambda, \mathcal{A}) = 1] \right| \ .$$

*A SE is secure, if for any any polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that:* $\mathsf{Adv}_{\mathsf{SE},\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)$.

## 3 MCFE from Public-Key Single-Input FE

In this section, we build a multi-client FE for inner products generically from any public-key single-input FE and a standard PRF.

### 3.1 Construction

The construction resembles the multi-input FE from [ACF+18], where an inner layer of information-theoretic one-time FE is combined with an outer layer of single-input FE. We manage to extend this paradigm to the setting where the encryption additionally takes a label as input: the one-time pads are replaced by pads which are pseudorandom for all used labels $\ell$, using techniques similar to those used in [ABKW19] to decentralize the generation of functional secret keys.

The underlying single-input FE is required to satisfy simple structural properties, originally defined in [ACF+18] and recalled below (converted to the public-key setting), which are satisfied by all known existing single-input FE for inner products.

$\underline{\mathsf{Setup}(1^\lambda, 1^n):}$

$\mathsf{pp}_{\mathsf{ipfe}} \leftarrow \mathsf{Setup}^\star_{\mathsf{ipfe}}(1^\lambda, 1^n)$, with $L$ implicitly defined from $\mathsf{pp}_{\mathsf{ipfe}}$

Return $\mathsf{pp} = \mathsf{pp}_{\mathsf{ipfe}}$

$\underline{\mathsf{KeyGen}(\mathsf{pp}):}$

$(\mathsf{msk}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}) \leftarrow \mathsf{KeyGen}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}); \mathsf{msk} := \mathsf{msk}_{\mathsf{ipfe}}$

For $i \in [n]$, $j > i : \mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$

Return $\{\mathsf{sk}_i = (\mathsf{mpk}, \{\mathsf{K}_{i,j}\}_{j\in[n]})\}_{i\in[n]}$ and $\mathsf{msk}$

$\underline{\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i \in \mathcal{R}^m, \ell \in \mathsf{Labels}):}$

Parse $\mathsf{sk}_i = (\mathsf{mpk}_{\mathsf{ipfe}}, \{\mathsf{K}_{i,j}\}_{j\in[n]})$

$\boldsymbol{t}_{i,\ell} := \sum_{j\neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) \in \mathbb{Z}_L^{mn}$

$\boldsymbol{w}_i := (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i\|\mathbf{0}\|\ldots\|\mathbf{0}) + \boldsymbol{t}_{i,\ell} \bmod L$

$\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$

Return $\mathsf{ct}_i$

$\underline{\mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y} \in \mathcal{R}^{mn}):}$

Return $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{msk}_{\mathsf{ipfe}}, \boldsymbol{y})$

$\underline{\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\boldsymbol{y}}, \{\mathsf{ct}_i\}_{i\in[n]}):}$

For $i \in [n]$, $\mathcal{E}(\langle \boldsymbol{w}_i, \boldsymbol{y}\rangle \bmod L, \mathsf{noise}_i) \leftarrow \mathsf{Dec}_{\mathsf{ipfe},1}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{sk}_{\boldsymbol{y}}, \mathsf{ct}_i)$

Return $\mathsf{Dec}_{\mathsf{ipfe},2}(\mathsf{pp}_{\mathsf{ipfe}}, \mathcal{E}(\langle \boldsymbol{w}_1, \boldsymbol{y}\rangle \bmod L, \mathsf{noise}_1)) \circ \cdots \circ \mathcal{E}(\langle \boldsymbol{w}_n, \boldsymbol{y}\rangle \bmod L, \mathsf{noise}_n))$

**Fig. 5.** Inner-Product MCFE for $\mathcal{F}_\rho, \rho = (\mathbb{Z}, n, m, X, Y)$ built from a public-key FE $\mathsf{FE} := (\mathsf{Setup}_{\mathsf{ipfe}}, \mathsf{Enc}_{\mathsf{ipfe}}, \mathsf{KeyDer}_{\mathsf{ipfe}}, \mathsf{Dec}_{\mathsf{ipfe}})$ for $\mathcal{F}_{\rho_{\mathsf{ipfe}}}, \rho_{\mathsf{ipfe}} = (\mathbb{Z}, 1, n \cdot m, 2X, Y)$. We assume $\mathsf{FE}$ satisfies the two-step decryption property (see Definition 3.1), hence the existence of PPT algorithms $\mathsf{Setup}^\star_{\mathsf{ipfe}}$, $\mathsf{Dec}_{\mathsf{ipfe},1}$ and $\mathsf{Dec}_{\mathsf{ipfe},2}$. Here, for any $\mathsf{K} \in \{0,1\}^\lambda$, $\mathsf{PRF}_{\mathsf{K}} : \mathsf{Labels} \to \mathbb{Z}_L^{mn}$ is a pseudorandom function (see Section 2.4).

**Definition 3.1 (Two-step decryption [ACF+18]).** *A public-key FE scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ *for the function ensemble* $\mathcal{F}^{\mathsf{ip}}_{\rho}, \rho = (\mathbb{Z}, 1, m, X, Y)$ *satisfies the* two-step decryption *property if it admits PPT algorithms* $\mathsf{Setup}^{\star}$, $\mathsf{Dec}_1, \mathsf{Dec}_2$ *and an encoding function* $\mathcal{E}$ *such that:*

1. *For all* $\lambda \in \mathbb{N}, \mathsf{Setup}^{\star}(1^{\lambda}, 1^{n})$ *outputs* $\mathsf{pp}$ *where* $\mathsf{pp}$ *includes* $\rho = (\mathbb{Z}, 1, m, X, Y)$ *and a bound* $B \in \mathbb{R}^{+}$*, as well as the description of a group* $\mathbb{G}$ *(with group law* $\circ$*) of order* $L > n \cdot m \cdot X \cdot Y$*, which defines the encoding function* $\mathcal{E} : \mathbb{Z}_L \times \mathbb{Z} \to \mathbb{G}$*.*
2. *For all* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \boldsymbol{x} \in \mathbb{Z}^m, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{mpk}, \boldsymbol{x}), \boldsymbol{y} \in \mathbb{Z}^m$*, and* $\mathsf{sk} \leftarrow \mathsf{KeyDer}(\mathsf{msk}, \boldsymbol{y})$*, we have*
$$\mathsf{Dec}_1(\mathsf{pp}, \mathsf{sk}, \mathsf{ct}) = \mathcal{E}(\langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod L, \mathsf{noise}) \ ,$$
*for some* $\mathsf{noise} \in \mathbb{Z}$ *that depends on* $\mathsf{ct}$ *and* $\mathsf{sk}$*. Furthermore, it holds that* $Pr[|\mathsf{noise}| < B] = 1 - \mathrm{negl}(\lambda)$*, where the probability is taken over the random coins of* $\mathsf{KeyGen}$ *and* $\mathsf{KeyDer}$*. Note that there is no restriction on the norm of* $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ *here.*
3. *The encoding* $\mathcal{E}$ *is linear, that is: for all* $\gamma, \gamma' \in \mathbb{Z}_L, \mathsf{noise}, \mathsf{noise}' \in \mathbb{Z}$*, we have*
$$\mathcal{E}(\gamma, \mathsf{noise}) \circ \mathcal{E}(\gamma', \mathsf{noise}') = \mathcal{E}(\gamma + \gamma' \bmod L, \mathsf{noise} + \mathsf{noise}') \ .$$
4. *For all* $\gamma < n \cdot m \cdot X \cdot Y$*, and* $|\mathsf{noise}| < n \cdot B$*,* $\mathsf{Dec}_2(\mathsf{pp}, \mathcal{E}(\gamma, \mathsf{noise})) = \gamma$*.*

**Definition 3.2 (Linear encryption [ACF+18]).** *A secret-key FE scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ *is said to satisfy the linear encryption property if there exists a deterministic algorithm* $\mathsf{Add}$ *that takes as input a ciphertext and a message, such that for all* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}^m$*, the following are identically distributed:*
$$\mathsf{Add}(\mathsf{Enc}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{x}), \boldsymbol{x}'), \quad \textit{and} \quad \mathsf{Enc}\big(\mathsf{pp}, \mathsf{msk}, (\boldsymbol{x} + \boldsymbol{x}' \bmod L)\big) \ .$$
*Recall that the value* $L \in \mathbb{N}$ *is defined as part of the output of the algorithm* $\mathsf{Setup}^{\star}$ *(see the two-step decryption property above).*

**Correctness.** The correctness of the scheme in Fig. 5 follows from (i) the correctness and Definition 3.1 (two-step decryption) of the single-input scheme, and (ii) the fact that for all $\ell \in \mathsf{Labels}$, $\sum_{i \in [n]} \boldsymbol{t}_{i,\ell} = \boldsymbol{0}$, by definition of the vectors $\boldsymbol{t}_{i,\ell}$. Thus, writing $\boldsymbol{w}_i := (\boldsymbol{0} \| \ldots \| \boldsymbol{0} \| \boldsymbol{x}_i \| \boldsymbol{0} \| \ldots \| \boldsymbol{0}) + \boldsymbol{t}_{i,\ell} \bmod L$, we have $\sum_{i \in [n]} \boldsymbol{w}_i \bmod L = \boldsymbol{x} \bmod L \in \mathbb{Z}_L^{mn}$, where $\boldsymbol{x} \in \mathcal{R}^{nm}$ denotes the concatenation of the $n$ vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

More precisely, consider any vector $\boldsymbol{x} := (\boldsymbol{x}_1 \| \cdots \| \boldsymbol{x}_n) \in (\mathbb{Z}^m)^n$, $\boldsymbol{y} \in \mathbb{Z}^{mn}$, such that $\|\boldsymbol{x}\|_{\infty} < X$, $\|\boldsymbol{y}\|_{\infty} < Y$ and let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda})$, $(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y})$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i, \ell)$ for all $i \in [n]$.

By (2) of Definition 3.1, the decryption algorithm $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{\boldsymbol{y}}, \{\mathsf{ct}_i\}_{i \in [n]})$ computes $\mathcal{E}(\langle \boldsymbol{w}_i, \boldsymbol{y} \rangle \bmod L, \mathsf{noise}_i) \leftarrow \mathsf{Dec}_{\mathsf{ipfe},1}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}_i)$ where for all $i \in [n]$, $|\mathsf{noise}_i| < B$ with probability $1 - \mathrm{negl}(\lambda)$, where $B \in \mathbb{R}^{+}$ is the bound output by $\mathsf{Setup}^{\star}_{\mathsf{ipfe}}$.

By (3) of Definition 3.1 (linearity of $\mathcal{E}$) we have:

$$\begin{aligned} &\mathcal{E}(\langle \boldsymbol{w}_1, \boldsymbol{y} \rangle \bmod L, \mathsf{noise}_1) \circ \cdots \circ \mathcal{E}(\langle \boldsymbol{w}_n, \boldsymbol{y} \rangle \bmod L, \mathsf{noise}_n) \\ &= \mathcal{E}\left( \langle \sum_{i \in [n]} \boldsymbol{w}_i, \boldsymbol{y} \rangle, \sum_{i \in [n]} \mathsf{noise}_i \right) = \mathcal{E}\left( \langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod L, \sum_{i \in [n]} \mathsf{noise}_i \right) . \end{aligned}$$

Since $\langle \boldsymbol{x}, \boldsymbol{y} \rangle < n \cdot m \cdot X \cdot Y < L$ and $\left| \sum_{i \in [n]} \mathsf{noise}_i \right| < n \cdot B$, we have

$$\mathsf{Dec}_{\mathsf{ipfe},2}\big(\mathcal{E}(\langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod L, \sum_{i \in [n]} \mathsf{noise}_i)\big) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle,$$

by (4) of Definition 3.1.

### 3.2 Static Security

Now we proceed to prove the sta-pos$^+$-IND-security of the scheme, that is, security with static corruption, which serves as a warm up to the more complicated proof of adt-pos$^+$-IND-security, that we give later. Using the generic transformation in Section 4, we can remove the pos$^+$ restriction, and obtain adt-any-IND security.

**Theorem 3.3 (sta-pos$^+$-IND-security).** *If the FE scheme* $\mathsf{FE} = (\mathsf{Setup}_{\mathsf{ipfe}}, \mathsf{KeyGen}_{\mathsf{ipfe}}, \mathsf{KeyDer}_{\mathsf{ipfe}}, \mathsf{Enc}_{\mathsf{ipfe}}, \mathsf{Dec}_{\mathsf{ipfe}})$ *is an* any*-IND-secure FE scheme for the inner product functionality defined as* $\mathcal{F}^{\mathsf{ip}}_{\rho_{\mathsf{ipfe}}}, \rho_{\mathsf{ipfe}} = (\mathbb{Z}, 1, m, 2X, Y)$*, and* $\mathsf{PRF}$ *is secure, then* $\mathsf{MCFE}$ *from Fig. 5 is* sta-pos$^+$*-IND-secure for the functionality defined as* $\mathcal{F}^{\mathsf{ip}}_{\rho}, \rho = (\mathbb{Z}, n, m, X, Y)$*. Namely, for any PPT adversary* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *such that:*

$$\mathsf{Adv}^{\text{sta-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq 2q_{\mathsf{Enc}} \cdot \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}}(\lambda) + 2(n-1)q_{\mathsf{Enc}} \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}'}(\lambda),$$

*where* $q_{\mathsf{Enc}}$ *denotes the number of distinct labels queried to* $\mathsf{QLeftRight}$*.*

$\mathsf{G}_0$, $\boxed{\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3}$, $\mathsf{G}_4$:

$\mathcal{CS} \leftarrow \mathcal{A}(1^\lambda, 1^n)$
$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
$\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QKeyD}(\cdot)}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in\mathcal{CS}})$
Output: $\alpha$ if Condition (*) is satisfied, or 0 otherwise.

$\mathsf{QKeyD}(\boldsymbol{y})$:
Return $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y})$

$\mathsf{QEnc}(i, \boldsymbol{x}_i^j, \ell)$:
$\boldsymbol{t}_{i,\ell} \leftarrow \mathsf{Gen}(i, \ell)$
$\boldsymbol{w}_i := (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^j\|\mathbf{0}\|\ldots\|\mathbf{0}) + \boldsymbol{t}_{i,\ell} \bmod L$
$\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$
Return $\mathsf{ct}_i$

$\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$:
$\boldsymbol{t}_{i,\ell^\star} \leftarrow \mathsf{Gen}(i, \ell^\star)$
$\boldsymbol{w}_i := (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,0} + \boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0}\|\mathbf{0}\|\ldots\|\mathbf{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L$
$\boldsymbol{w}_i := (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L$
$\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$
Return $\mathsf{ct}_i$

$\mathsf{Gen}(i, \ell)$:
Parse $\mathsf{sk}_i = \{\mathsf{K}_{i,j}\}_{j\in[n]}$
$\boldsymbol{t}_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) \in \mathbb{Z}_L^{mn}$

If $i \in \mathcal{HS} := \{i_1, \ldots, i_h\}$, then:
- If $i = i_1$, $\boldsymbol{t}_{i,\ell} := \sum_{j\in\mathcal{CS}}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) + \sum_{t=2}^{h}\mathsf{RF}(t, \ell)$.
- If $i = i_t$, for $t \in [2, \ldots, h]$, $\boldsymbol{t}_{i,\ell} := \sum_{j\in[n]\setminus\{i_t,i_1\}}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) - \mathsf{RF}(t, \ell)$.

Return $\boldsymbol{t}_{i,\ell}$

**Fig. 6.** Games for the proof of Theorem 3.3. Here, $\mathcal{HS} := [n] \setminus \mathcal{CS}$. Condition (*) is given in Definition 2.1. Here, $\mathsf{RF}$ denotes a random function that is computed on the fly. WLOG, $\mathsf{QLeftRight}$ is only queried on label $\ell^\star$, and $\mathsf{QEnc}$ isn't queried on $\ell^\star$.

*Proof.* For simplicity, we consider the case where $\mathcal{A}$ only queries $\mathsf{QLeftRight}$ on one label $\ell^\star$, and never queries $\mathsf{QEnc}$ on $\ell^\star$. We build PPT adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that: $\mathsf{Adv}^{\text{sta-pos}^+\text{-IND-1-label}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq 2 \cdot \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}}(\lambda) + 2(n-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}'}(\lambda)$, where $\mathsf{Adv}^{\text{sta-pos}^+\text{-IND-1-label}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n)$ is defined as $\mathsf{Adv}^{\text{sta-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n)$, except with the limitations mentioned above, namely, $\mathcal{A}$ can query $\mathsf{QLeftRight}$ on at most one label, which cannot be queried to $\mathsf{QEnc}$. Then we use Lemma 2.5 to obtain the theorem.

First, consider the case where there is only one honest user. In this case, the security follows directly from the any-IND security of $\mathsf{FE}$. Namely, in that case we build a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\text{sta-pos}^+\text{-IND-1-label}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}}(\lambda)$. Given $\mathsf{pp}_{\mathsf{ipfe}}$, $\mathcal{B}$ first samples the keys $\mathsf{K}_{i,j}$ for all $i, j \in [n]$, thanks to which it can compute $\mathsf{pp}$, $\{\mathsf{sk}_i\}_{i\in[n]}$, and send $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in\mathcal{CS}})$ to $\mathcal{A}$. $\mathcal{B}$ can answer all queries to $\mathsf{QEnc}(i, \boldsymbol{x}_i^j, \ell)$, by returning $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i^j, \ell)$, since it know $\mathsf{sk}_i$ for all $i \in [n]$. Call $i^\star$ the only honest slot. $\mathcal{B}$ can answer all queries to $\mathsf{QEnc}(i, \cdot, \cdot, \cdot)$ and $\mathsf{QLeftRight}(i, \cdot, \cdot, \cdot)$ for $i \neq i^\star$, using $\mathsf{pp}$ and $\{\mathsf{sk}_i\}_{i\in[n]}$. Whenever $\mathcal{A}$ queries $\mathsf{QLeftRight}(i^\star, \boldsymbol{x}_{i^\star}^{j,0}, \boldsymbol{x}_{i^\star}^{j,1}, \ell^\star)$, $\mathcal{B}$ queries its own left right oracle on $(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_{i^\star}^{j,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), (\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0})$, to receive $\mathsf{ct}_i := \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, \mathsf{sk}_{i^\star}, (\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,\beta}\|\mathbf{0}\|\ldots\|\mathbf{0}))$, where $\beta \in \{0, 1\}$, depending on the experiment $\mathcal{B}$ is interacting with. Then, $\mathcal{B}$ computes $\boldsymbol{t}_{i^\star,\ell^\star}$ as described in Fig. 5, and returns $\mathsf{Add}(\mathsf{ct}_{i^\star}, \boldsymbol{t}_{i^\star,\ell^\star})$ to $\mathcal{A}$, which, according to the property from Definition 3.2 (linear encryption), is identically distributed to $\mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, (\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,\beta}\|\mathbf{0}\|\ldots\|\mathbf{0}) + \boldsymbol{t}_{i^\star,\ell^\star} \bmod L)$. Whenever $\mathcal{A}$ queries $\mathsf{QKeyD}$ on input $\boldsymbol{y}$, $\mathcal{B}$ queries its own $\mathsf{QKeyD}$ on the same input, and forwards the output to $\mathcal{A}$. For all $\boldsymbol{y}$ queried to $\mathsf{QKeyD}$, we have $\langle(\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle = \langle(\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle$, by Condition (*). Moreover, for all $\beta \in \{0, 1\}$, $\|(\mathbf{0}\|\ldots\|\boldsymbol{x}_{i^\star}^{j,\beta}\|\mathbf{0}\|\ldots\|\mathbf{0})\|_\infty < 2X$. Thus, the queries $\mathcal{B}$ sends to its left-right oracle are legitimate. This concludes the case where there is only one honest user.

Second, we consider the case where there is more than one honest user. For this case, we proceed via a hybrid argument, using the games described in Fig. 6. Note that $\mathsf{G}_0$ corresponds to $\text{sta-pos}^+\text{-IND}_0^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$, and $\mathsf{G}_4$ corresponds to $\text{sta-pos}^+\text{-IND}_1^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$, with the one label restriction. Thus, we have:

$$\mathsf{Adv}^{\text{sta-pos}^+\text{-IND-1-label}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) = \left|\mathsf{Win}^{\mathsf{G}_0}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_4}_{\mathcal{A}}(\lambda, n)\right|.$$

**Game $\mathsf{G}_1$.** In game $\mathsf{G}_1$, we change the way the vectors $\boldsymbol{t}_{i,\ell}$ used by $\mathsf{QEnc}$ and $\mathsf{QLeftRight}$ are generated, switching the values $\mathsf{PRF}_{K_{i_1,i_t}}(\ell)$ to $\mathsf{RF}(t, \ell)$, for all $t \in [2, h]$, where we write the set of honest users $\mathcal{HS} := \{i_1, \ldots, i_h\}$, and $\mathsf{RF}$ denotes a random function, computed on the fly (see Fig. 6). The transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ is justified by the security of the $\mathsf{PRF}$. Namely, in Lemma 3.4, we exhibit a PPT adversary $\mathcal{B}_0$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_0}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_1}_{\mathcal{A}}(\lambda, n)\right| \leq (h-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_0}(\lambda),$$

where $h \leq n$ denotes the number of honest users.

**Game $\mathsf{G}_2$.** In game $\mathsf{G}_2$, the vectors $\boldsymbol{w}_i$ used to generate the challenge ciphertexts contain an additional vector $(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0}\|\mathbf{0}\|\ldots\|\mathbf{0})$. The transition from $\mathsf{G}_1$ to $\mathsf{G}_2$ is justified by the any-IND security of $\mathsf{FE}$. Namely, in Lemma 3.5, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_1}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_2}_{\mathcal{A}}(\lambda, n)\right| \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_1}(\lambda).$$

**Game $\mathsf{G}_3$.** In game $\mathsf{G}_3$, the vectors $\boldsymbol{w}_i$ used in the challenge ciphertexts are of the form: $\boldsymbol{w}_i := (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0})$. The transition from $\mathsf{G}_2$ to $\mathsf{G}_3$ is justified by the any-IND security of $\mathsf{FE}$. Namely, in Lemma 3.6, we exhibit a PPT adversary $\mathcal{B}_2$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_2}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_3}_{\mathcal{A}}(\lambda, n)\right| \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_2}(\lambda).$$

**Game $\mathsf{G}_4$.** This game is $\text{sta-pos}^+\text{-IND}_1^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$. The transition from $\mathsf{G}_3$ to $\mathsf{G}_4$ is symmetric to the transition from $\mathsf{G}_0$ to $\mathsf{G}_1$, justified by the security of the $\mathsf{PRF}$. Namely, it can be proven as in Lemma 3.4 that there exists a PPT adversary $\mathcal{B}_3$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_3}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_4}_{\mathcal{A}}(\lambda, n)\right| \leq (h-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_3}(\lambda),$$

where $h \leq n$ denotes the number of honest users. We defer to the proof of Lemma 3.4 for further details.

Putting everything together, we obtain the theorem. □

**Lemma 3.4 (Transition from $\mathsf{G}_0$ to $\mathsf{G}_1$).** *There exists a PPT adversary $\mathcal{B}'$ such that* $\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_0}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda,n)\right| \leq (h-1)\cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}'}(\lambda)$.

*Proof.* We can use the security of the PRF on all keys $\mathsf{K}_{i,j}$ where $i,j \in \mathcal{HS}$, since these are hidden from the adversary $\mathcal{A}$. We show that using the security of the PRF on $h-1$ carefully chosen such keys is sufficient to transition from $\mathsf{G}_0$ to $\mathsf{G}_1$. Namely, if we write $\mathcal{HS} := \{i_1,\dots,i_h\}$, where the indices $i_1 < i_2 < \dots < i_h$ are ordered, we use the security of the PRF on keys of the form $\mathsf{K}_{i_1,j}$ for all $j \in \mathcal{HS}\setminus\{i_1\}$.

We build the adversary $\mathcal{B}'$ as follows. Given $\mathcal{CS}$ sent by $\mathcal{A}$, it samples $\mathsf{pp}_{\mathsf{ipfe}} \leftarrow \mathsf{Setup}^{\star}_{\mathsf{ipfe}}(1^\lambda,1^n)$ and $\mathsf{msk}_{\mathsf{ipfe}} \leftarrow \mathsf{KeyGen}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}})$. For all $i \in [n]\setminus\{i_1\}$, for all $j > i$, $\mathcal{B}'$ samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$, thanks to which it can compute $\mathsf{sk}_i := \{\mathsf{K}_{i,j}\}_{j\in[n]}$ for all $i \in \mathcal{CS}$ and send them to $\mathcal{A}$. $\mathcal{B}'$ can simulate the oracle QKeyD using $\mathsf{msk}_{\mathsf{ipfe}}$, and answers the queries to $\mathsf{QEnc}(i,\boldsymbol{x}_i^j,\ell)$ for $i \in \mathcal{CS}$, and $\mathsf{QLeftRight}(i,\boldsymbol{x}_i^{j,0},\boldsymbol{x}_i^{j,1},\ell^\star)$ for $i \in \mathcal{CS}$ using $\mathsf{sk}_i$.

To answer $\mathsf{QEnc}(i_1,\boldsymbol{x}_{i_1}^j,\ell)$ or $\mathsf{QLeftRight}(i_1,\boldsymbol{x}_{i_1}^{j,0},\boldsymbol{x}_{i_1}^{j,1},\ell^\star)$, $\mathcal{B}'$ computes

$$\boldsymbol{t}_{i_1,\ell} := \sum_{j\in\mathcal{CS}} (-1)^{j<i_1}\mathsf{PRF}_{\mathsf{K}_{i_1,j}}(\ell) + \sum_{t=2}^{h}\mathsf{RF}(t,\ell).$$

To answer $\mathsf{QEnc}(i_t,\boldsymbol{x}_{i_t}^j,\ell)$ or $\mathsf{QLeftRight}(i_t,\boldsymbol{x}_{i_t}^{j,0},\boldsymbol{x}_{i_t}^{j,1},\ell^\star)$, for $t \in [2,\dots,h]$, $\mathcal{B}'$ computes

$$\boldsymbol{t}_{i_t,\ell} := \sum_{j\in[n]\setminus\{i_t,i_1\}} (-1)^{j<i_t}\mathsf{PRF}_{\mathsf{K}_{i_t,j}}(\ell) - \mathsf{RF}(t,\ell).$$

Here, $\mathsf{RF}(t,\ell)$ is either a truly random function, or $\mathsf{PRF}_{\mathsf{K}_{i_1,i_t}}(\ell)$, depending on the experiment $\mathcal{B}'$ is interacting with. In fact, we implicitly use a hybrid argument which goes over all $t \in [2,\dots,h]$ here, in order to switch the values $\mathsf{PRF}_{\mathsf{K}_{i_1,i_t}}(\ell)$ to $\mathsf{RF}(t,\ell)$. Thus, we obtain $\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_0}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda,n)\right| \leq (h-1)\cdot\mathsf{Adv}_{\mathsf{PRF},\mathcal{B}'}(\lambda)$. □

**Lemma 3.5 (Transition from $\mathsf{G}_1$ to $\mathsf{G}_2$).** *There exists a PPT adversary $\mathcal{B}_1$ such that* $\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda,n)\right| \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_1}(\lambda)$.

*Proof.* The adversary $\mathcal{B}_1$ works as follows. Given $\mathcal{CS}$ sent by $\mathcal{A}$, and $\mathsf{pp}_{\mathsf{ipfe}}$ from its own experiment, $\mathcal{B}_1$ samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$, thanks to which it can send the $\mathsf{sk}_i$ for all $i \in \mathcal{CS}$, together with $\mathsf{pp}_{\mathsf{ipfe}}$ to $\mathcal{A}$. Since $\mathcal{B}_1$ knows the $\mathsf{sk}_i$ for all $i \in [n]$, it can answer the oracle QEnc as described in Fig. 6.

Whenever $\mathcal{A}$ queries QKeyD on input $\boldsymbol{y}$, $\mathcal{B}_1$ queries its own oracle on the same input, and forwards the answer to $\mathcal{A}$.

Since we are considering pos$^+$-IND security, we know $\mathcal{A}$ queries all honest slots on $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$ and we denote by $i_{t^\star}$ the last honest slot queried on $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$. We call $\Delta_{\boldsymbol{x}} := (\boldsymbol{x}_1^{1,1} - \boldsymbol{x}_1^{1,0},\dots,\boldsymbol{x}_n^{1,1} - \boldsymbol{x}_n^{1,0})$, where for all $i \in \mathcal{HS}$, $(i,\boldsymbol{x}_i^{1,0},\boldsymbol{x}_i^{1,1},\ell^\star)$ is the first query of the form $\mathsf{QLeftRight}(i,\cdot,\cdot,\ell^\star)$, and for all $i \in \mathcal{CS}$, we define $\boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0} := \boldsymbol{0} \in \mathbb{Z}^m$ (note that QLeftRight can be queried on a corrupted slot, but by Condition (*), that means the query is of the form $(i,\boldsymbol{x}_i^{1,0},\boldsymbol{x}_i^{1,1},\ell^\star)$).

Whenever $\mathcal{A}$ queries $\mathsf{QLeftRight}(i,\boldsymbol{x}_i^{j,0},\boldsymbol{x}_i^{j,1},\ell^\star)$, $\mathcal{B}_1$ computes the vectors $\boldsymbol{t}_{i,\ell^\star}$ for all $i \in [n]$, using $\mathsf{sk}_i$ and computing the random function RF on the fly, as described in Fig. 6. Then, if $i \neq i_{t^\star}$, it computes $\boldsymbol{w}_i := (\boldsymbol{0}\|\dots\|\boldsymbol{0}\|\boldsymbol{x}_i^{j,0}\|\boldsymbol{0}\|\dots\|\boldsymbol{0}) + \boldsymbol{t}_{i,\ell^\star} \mod L$, and returns $\mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}},\mathsf{mpk}_{\mathsf{ipfe}},\boldsymbol{w}_i)$ to $\mathcal{A}$. If $i = i_{t^\star}$, then $\mathcal{B}_1$ queries its left-right oracle on input $(\boldsymbol{0},\Delta_{\boldsymbol{x}})$ to get $\mathsf{ct}_i := \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}},\mathsf{mpk}_{\mathsf{ipfe}},\boldsymbol{0})$ or $\mathsf{ct}_i := \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}},\mathsf{mpk}_{\mathsf{ipfe}},\Delta_{\boldsymbol{x}})$, depending on the experiment $\mathcal{B}_1$ is interacting with. Note that at this point, $\Delta_{\boldsymbol{x}}$ is entirely known to $\mathcal{B}_1$, since $i_{t^\star}$ is the last honest slot to be queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$. Then, $\mathcal{B}_1$ computes $\boldsymbol{w}_i := (\boldsymbol{0}\|\dots\|\boldsymbol{0}\|\boldsymbol{x}_i^{j,0}\|\boldsymbol{0}\|\dots\|\boldsymbol{0}) + \boldsymbol{t}_{i,\ell^\star} \mod L$ and returns $\mathsf{ct}_i' := \mathsf{Add}(\mathsf{ct}_i,\boldsymbol{w}_i)$, which, according to the property from Definition 3.2 (linear encryption), is identically distributed to $\mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}},\mathsf{mpk}_{\mathsf{ipfe}},\boldsymbol{w}_i \mod L)$ or $\mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}},\mathsf{mpk}_{\mathsf{ipfe}},\boldsymbol{w}_i + \Delta_{\boldsymbol{x}} \mod L)$, (again, depending on which experiment $\mathcal{B}_1$ is interacting with). For

all $\boldsymbol{y}$ queried to $\mathsf{QKeyD}$, we have $\langle \Delta_{\boldsymbol{x}}, \boldsymbol{y} \rangle = 0$, by Condition (*). Moreover, $\|\Delta_{\boldsymbol{x}}\|_\infty < 2X$. Thus, the queries $\mathcal{B}_1$ sends to its left-right oracle are legitimate. Finally, $\mathcal{B}_1$ returns $\mathsf{ct}'_i$ to $\mathcal{A}$.

To conclude, we show that when $\mathcal{B}_1$ is interacting with $\textbf{any-IND}_0^{\mathsf{FE}}(\lambda, 1, \mathcal{A})$, then it simulates the game $\mathsf{G}_1$, whereas it simulates the game $\mathsf{G}_2$ when it is interacting with $\textbf{any-IND}_1^{\mathsf{FE}}(\lambda, 1, \mathcal{A})$. It is clear for the case $\textbf{any-IND}_0^{\mathsf{FE}}(\lambda, 1, \mathcal{A})$. For the case $\textbf{any-IND}_1^{\mathsf{FE}}(\lambda, 1, \mathcal{A})$, we consider the vectors $\{\boldsymbol{u}_t\}_{t \in [h]}$, where we write $\mathcal{HS} := \{i_1, \ldots, i_h\}$ and we denote by $\boldsymbol{u}_1 := -\sum_{t=2}^{h} \mathsf{RF}(t, \ell^\star)$ and $\boldsymbol{u}_t := \mathsf{RF}(t, \ell^\star)$, for all $t \in [2, \ldots, n]$. These are shares of a perfect $h$ out of $h$ secret sharing of $\mathbf{0}$, that is, they are uniformly random conditioned on $\sum_{t \in [h]} \boldsymbol{u}_t = \mathbf{0}$. Thus, $\{\boldsymbol{u}_t\}_{t \in [t] \setminus \{t^\star\}} \cup \{\boldsymbol{u}_{t^\star} + \Delta_{\boldsymbol{x}}\}$ is a set of shares for a secret sharing of the vector $\Delta_{\boldsymbol{x}}$. Thus, the following distributions are identical:

$$\{\boldsymbol{u}_t\}_{t \in [h] \setminus \{t^\star\}} \cup \{\boldsymbol{u}_{t^\star} + \Delta_{\boldsymbol{x}}\}$$

and

$$\{\boldsymbol{u}_t + (\mathbf{0} \| \ldots \| \boldsymbol{x}_{i_t}^{1,1} - \boldsymbol{x}_{i_t}^{1,0} \| \mathbf{0} \| \ldots \| \mathbf{0})\}_{t \in [h]},$$

where for all $t \in [h]$, $\boldsymbol{u}_t \leftarrow \mathbb{Z}_L^{mn}$ such that $\sum_{t \in [h]} \boldsymbol{u}_t = \mathbf{0}$. The uppermost distribution corresponds to the simulation by $\mathcal{B}_1$ when it is interacting with $\textbf{any-IND}_1^{\mathsf{FE}}(\lambda, 1, \mathcal{A})$, while the lowermost distribution corresponds to the game $\mathsf{G}_{1.\rho}$. This concludes the proof. □

**Lemma 3.6 (Transition from $\mathsf{G}_2$ to $\mathsf{G}_3$).** *There exists a PPT adversary $\mathcal{B}_2$ such that* $\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3}(\lambda, n)\right| \leq \mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{any-IND}}(\lambda)$.

*Proof.* We build an adversary $\mathcal{B}_2$ against the any-IND security of $\mathsf{FE}$ as follows.

Given $\mathcal{CS}$ sent by $\mathcal{A}$, and $\mathsf{pp}_{\mathsf{ipfe}}$ from its own experiment, $\mathcal{B}_2$ samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$, thanks to which it can send the $\mathsf{sk}_i$ for all $i \in \mathcal{CS}$, together with $\mathsf{pp}_{\mathsf{ipfe}}$ to $\mathcal{A}$, and answer the oracle queries to $\mathsf{QEnc}$ as described in Fig. 6.

Then, whenever $\mathcal{A}$ queries $\mathsf{QKeyD}$ on input $\boldsymbol{y}$, $\mathcal{B}_2$ queries its own oracle on the same input, and forwards the answer to $\mathcal{A}$. Whenever $\mathcal{A}$ queries $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$, $\mathcal{B}_2$ computes $\boldsymbol{t}_{i,\ell^\star}$ using $\mathsf{sk}_i$ and computing the random function $\mathsf{RF}$ on the fly, as described in Fig. 6. Then, $\mathcal{B}_2$ queries its left-right oracle on input $(\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,0} - \boldsymbol{x}_i^{1,0} \| \mathbf{0} \| \ldots \| \mathbf{0}), (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,1} - \boldsymbol{x}_i^{1,1} \| \mathbf{0} \| \ldots \| \mathbf{0})$ to get

$$\mathsf{ct}_i := \mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}(\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,\beta} - \boldsymbol{x}_i^{1,\beta} \| \mathbf{0} \| \ldots \| \mathbf{0})),$$

where $\beta \in \{0,1\}$, depending on the experiment $\mathcal{B}_2$ is interacting with. Finally, $\mathcal{B}_2$ computes $\boldsymbol{v}_i := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{1,1} \| \mathbf{0} \| \ldots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L$, and returns $\mathsf{ct}'_i := \mathsf{Add}(\mathsf{ct}_i, \boldsymbol{v}_i)$ to $\mathcal{A}$, which, according to the property from Definition 3.2, is identically distributed to $\mathsf{Enc}_{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}, (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,\beta} - \boldsymbol{x}_i^{1,\beta} + \boldsymbol{x}_i^{1,1} \| \mathbf{0} \| \ldots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L)$. For all $\boldsymbol{y}$ queried to $\mathsf{QKeyD}$, Condition (*) implies that $\langle (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,0} - \boldsymbol{x}_i^{1,0} \| \mathbf{0} \| \ldots \| \mathbf{0}), \boldsymbol{y} \rangle = \langle (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,1} - \boldsymbol{x}_i^{1,1} \| \mathbf{0} \| \ldots \| \mathbf{0}), \boldsymbol{y} \rangle$ for all queries $(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$ to $\mathsf{QLeftRight}$. Moreover, for all $\beta \in \{0,1\}$, we have $\|(\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,\beta} - \boldsymbol{x}_i^{1,\beta} \| \mathbf{0} \| \ldots \| \mathbf{0})\|_\infty < 2X$. Thus, the queries $\mathcal{B}_2$ sends to its left-right oracle are legitimate. □

### 3.3 Adaptive Security

Now we proceed to prove the adt-pos$^+$-IND-security of the scheme, that is, security with adaptive corruption. As before, using the generic transformation in Section 4, we can remove the pos$^+$ restriction, and obtain adt-any-IND security.

**Theorem 3.7 (adt-pos$^+$-IND-security).** *If the FE scheme* $\mathsf{FE} = (\mathsf{Setup}_{\mathsf{ipfe}}, \mathsf{KeyGen}_{\mathsf{ipfe}}, \mathsf{KeyDer}_{\mathsf{ipfe}}, \mathsf{Enc}_{\mathsf{ipfe}}, \mathsf{Dec}_{\mathsf{ipfe}})$ *is an* any*-IND-secure FE scheme for the inner product functionality defined as* $\mathcal{F}_{\rho_{\mathsf{ipfe}}}^{\mathsf{ip}}, \rho_{\mathsf{ipfe}} = (\mathbb{Z}, 1, m, 2X, Y)$*, and* $\mathsf{PRF}$ *is secure, then* $\mathsf{MCFE}$ *from Fig. 5 is* adt-pos$^+$*-IND-secure for the functionality defined as* $\mathcal{F}_\rho^{\mathsf{ip}}, \rho = (\mathbb{Z}, n, m, X, Y)$*. Namely, for any PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that:*

$$\mathsf{Adv}_{\mathsf{MCFE},\mathcal{A}}^{\text{adt-pos}^+\text{-IND}}(\lambda, n) \leq 2(n+1)n(n-1)^2 q_{\mathsf{Enc}} \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(\lambda) + 2(n+1) q_{\mathsf{Enc}} \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}'}^{\text{any-IND}}(\lambda) \ ,$$

*where $q_{\mathsf{Enc}}$ denotes the number of distinct labels queried to* $\mathsf{QLeftRight}$.

$\mathsf{G}_0^\star$, $\boxed{\mathsf{G}_1^\star, \mathsf{G}_2^\star, \mathsf{G}_3^\star}$, $\mathsf{G}_4^\star$:

$\kappa^\star \leftarrow \{0, \ldots, n\}$, $\beta \leftarrow \{0, 1\}$, $\boxed{\text{for all } t \in [2, \ldots, \kappa^\star],\ \boldsymbol{u}_t \leftarrow \mathbb{Z}_L^{mn}}$

$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$

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

Output $\alpha$ if Condition (*) is satisfied AND the guess $\kappa^\star$ is correct; 0 otherwise.

$\mathsf{QEnc}(i, \boldsymbol{x}_i^j, \ell)$:

Return $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i^j, \ell)$

$\mathsf{QKeyD}(\boldsymbol{y})$:

Return $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y})$

$\mathsf{QCor}(i)$:

Return $\mathsf{sk}_i$

$\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$:

Parse $\mathsf{sk}_i := \{\mathsf{K}_{i,j}\}_{j \in [n]}$, $\boldsymbol{v}_{i,\ell} := \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) \in \mathbb{Z}_L^{mn}$, $\boldsymbol{t}_{i,\ell} := \boldsymbol{v}_{i,\ell}$.

We write $\{i_1, \ldots, i_\kappa\}$ the set of explicitly honest slots, in the order they are revealed (that is, $i_1$ is the first revealed, $i_2$ is the second, and so forth). If $\kappa^\star \geq 2$ then do the following.

- If $i = i_1$, then $\boldsymbol{t}_{i,\ell} := \boldsymbol{v}_{i,\ell} + \sum_{t=2}^{\kappa^\star} \boldsymbol{u}_t$.
- If $i = i_t$, for $t \in [2, \ldots, \kappa^\star]$, then $\boldsymbol{t}_{i,\ell} := \boldsymbol{v}_{i,\ell} - \boldsymbol{u}_t$.
- If $i = i_t$, for $t > \kappa^\star$, that means $\kappa > \kappa^\star$, the guess was incorrect. Ends the game and output 0.

$\boldsymbol{w}_i := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,0} \| \mathbf{0} \| \cdots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell} \bmod L$

If $\kappa^\star \geq 2$: $\boldsymbol{w}_i := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,0} + \boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0} \| \mathbf{0} \| \cdots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell} \bmod L$

$\boldsymbol{w}_i := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,1} \| \mathbf{0} \| \ldots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell} \bmod L$

$\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$

Return $\mathsf{ct}_i$

**Fig. 7.** Games for the proof of Theorem 3.7. We say the guess $\kappa^\star$ is correct if it equals the size of $\mathcal{Q}$, the set of explicitly honest slots.

*Proof.* WLOG, we can assume that adversary $\mathcal{A}$ only queries $\mathsf{QLeftRight}$ on one label $\ell^\star$, that isn't queried to $\mathsf{QEnc}$. Namely, we show that there exist PPT adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that:

$$\mathsf{Adv}^{\text{adt-pos}^+\text{-IND-1-label}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq 2(n+1)n(n-1)^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(\lambda) + 2(n+1) \cdot \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}'}(\lambda) \ .$$

The theorem then follows from Lemma 2.5. We proceed via a hybrid argument, using the games described in Fig. 7. In this proof, we use the fact that any slot $i$ that is queried on $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{1,0}, \boldsymbol{x}_i^{1,1}, \ell^\star)$ with $\boldsymbol{x}_i^{1,0} \neq \boldsymbol{x}_i^{1,1}$ cannot be corrupted (otherwise Condition (*) from Definition 2.4 would be violated). We call such a slot explicitly honest (note that a slot can be honest without being explicitly honest, if it is only queried on $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$ with $\boldsymbol{x}_i^{j,0} = \boldsymbol{x}_i^{j,1}$ for instance). That is, a slot $i \in [n]$ is explicitly honest if the first query on this slot, $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{1,0}, \boldsymbol{x}_i^{1,1}, \ell^\star)$, is such that $\boldsymbol{x}_i^{1,0} \neq \boldsymbol{x}_i^{1,1}$. In the adaptive setting, the challenger does not know in advance which slot is going to be honest; and simply guessing the set of such slots would incur an exponential security. Instead, we use a more sophisticated proof strategy that relies on a hybrid argument on the number of explicitly honest slots (this number needs to be known in advance by the challenger, which can simply guess it, incurring only a polynomial security loss).

**Game $\mathsf{G}_0^\star$:** is as xx-yy-IND-1-label$_0$, except the size of $\mathcal{Q}$, which denotes the set of explicitly honest slots, is initially guessed by the experiment, by choosing a uniformly random $\kappa^\star \leftarrow \{0, \ldots, n\}$. The game behaves exactly as xx-yy-IND-1-label$_0$, except it ignores the $\mathcal{A}$'s output $\alpha$, and outputs 0 instead, in case the guess $\kappa^\star$ was incorrect. Since this guess is correct with probability $\frac{1}{n+1}$, we have

$$\mathsf{Win}^{\mathsf{G}_0^\star}_{\mathcal{A}}(\lambda, n) = \frac{1}{n+1} \cdot \mathsf{Win}^{\text{xx-yy-IND-1-label}_0}_{\mathcal{A}}(\lambda, n) \ .$$

**Game $\mathsf{G}_1^\star$:** in this game, we change the distribution of the ciphertexts output by $\mathsf{QLeftRight}$, for the case $\kappa^\star \geq 2$. For these, the vector $(\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,0} \| \mathbf{0} \| \ldots \| \mathbf{0})$ to be encrypted is added a share of a perfect $\kappa^\star$ out of $\kappa^\star$ secret sharing of $\mathbf{0}$. This game is similar to the game $\mathsf{G}_1$ from Fig. 6 for the proof of Theorem 3.3. We justify this transition using the security of the $\mathsf{PRF}$, as in Lemma 3.5, with the crucial difference that corruptions are adaptive here. Thus, the set of explicitly honest slots $\mathcal{Q}$ is not known in advance by the reduction. As explained above, guessing the entire set would incur an exponential security loss. Instead we introduce gradually the shares, starting with a 2 out of 2 perfect secret sharing, then 3 out of 3, and so forth, via a hybrid argument, until we reach the $\kappa^\star$ out of $\kappa^\star$ secret sharing among all queried slots. To go from one hybrid to another, we only require to guess a pair of users $(i, j)$ (as opposed to guessing the entire set of honest users) to use the security of the $\mathsf{PRF}$ on the key $\mathsf{K}_{i,j}$. Namely, in Lemma 3.8, we show that there exists a PPT adversary $\mathcal{B}_0$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_0^\star}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_1^\star}_{\mathcal{A}}(\lambda, n)\right| \leq n(n-1)^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_0}(\lambda)$$

**Game $\mathsf{G}_2^\star$:** in this game, the vectors $\boldsymbol{w}_i$ used to generate the ciphertexts output by $\mathsf{QLeftRight}$ contain an additional vector $(\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0} \| \mathbf{0} \| \ldots \| \mathbf{0})$. The transition from $\mathsf{G}_1^\star$ to $\mathsf{G}_2^\star$ is justified by the any-IND security of $\mathsf{FE}$, similarly than the transition from $\mathsf{G}_1$ to $\mathsf{G}_2$ in Fig. 6 for the proof of Theorem 3.3. Namely, in Lemma 3.9, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_1^\star}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_2^\star}_{\mathcal{A}}(\lambda, n)\right| \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_1}(\lambda).$$

**Game $\mathsf{G}_3^\star$:** in this game, the vectors $\boldsymbol{w}_i$ used in the ciphertexts output by $\mathsf{QLeftRight}$ are of the form: $\boldsymbol{w}_i := (\mathbf{0} \| \ldots \| \mathbf{0} \| \boldsymbol{x}_i^{j,1} \| \mathbf{0} \| \ldots \| \mathbf{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L$. The transition from $\mathsf{G}_{\rho-1.2}^\star$ to $\mathsf{G}_{\rho-1.3}^\star$ is justified by the any-IND security of $\mathsf{FE}$, similarly than the transition from $\mathsf{G}_2$ to $\mathsf{G}_3$ in Fig. 6 for the proof of Theorem 3.3. Namely, in Lemma 3.10, we build a PPT adversary $\mathcal{B}_2$ such that:

$$\left|\mathsf{Win}^{\mathsf{G}_2^\star}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_3^\star}_{\mathcal{A}}(\lambda, n)\right| \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_2}(\lambda).$$

**Game $\mathsf{G}_4^\star$.** The transition from $\mathsf{G}_3^\star$ to $\mathsf{G}_4^\star$ is symmetric to the transition from $\mathsf{G}_0^\star$ to $\mathsf{G}_1^\star$, justified by the security of the PRF. Namely, it can be proven as in Lemma 3.8 that there exists a PPT adversary $\mathcal{B}_3$ such that:

$$\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4^\star}(\lambda, n)\right| \leq n(n-1)^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_3}(\lambda).$$

We defer to the proof of Lemma 3.8 for further details. Since $\mathsf{G}_4^\star$ is exactly as the game xx-yy-IND$_0^{\mathsf{MCFE}}$ except it guesses $\kappa^\star \leftarrow \{0, \ldots, n\}$, we have

$$\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4^\star}(\lambda, n) = \frac{1}{n+1} \cdot \mathsf{Win}_{\mathcal{A}}^{\text{xx-yy-IND-1-label}_1}(\lambda, n).$$

Putting everything together, we obtain the theorem. □

**Lemma 3.8 (Transition from $\mathsf{G}_0^\star$ to $\mathsf{G}_1^\star$).** *There exists a PPT adversary $\mathcal{B}_0$ such that* $\left|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_0^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda, n)\right| \leq n(n-1)^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_0}(\lambda)$.

$\underline{\mathsf{G}_{0.\mu}^\star, \text{ for } \mu \in \{0, \ldots, n\}:}$
$\kappa^\star \leftarrow \{0, \ldots, n\}$, for all $t \in [2, \ldots, \kappa^\star]$, $\boldsymbol{u}_t \leftarrow \mathbb{Z}_L^{mn}$
$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
$\alpha \leftarrow \mathcal{A}^{\mathsf{QEnc}(\cdot,\cdot,\cdot,\cdot),\mathsf{QKeyD}(\cdot),\mathsf{QCor}(\cdot)}(\mathsf{pp})$
Output $\alpha$ if Condition (*) is satisfied AND the guess $\kappa^\star$ is correct; output 0 otherwise.

$\underline{\mathsf{QEnc}(i, \boldsymbol{x}_i^j, \ell):}$
Return $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i^j, \ell)$

$\underline{\mathsf{QKeyD}(\boldsymbol{y}):}$
Return $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y})$

$\underline{\mathsf{QCor}(i):}$
Return $\mathsf{sk}_i$

$\underline{\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star):}$
Parse $\mathsf{sk}_i := \{\mathsf{K}_{i,j}\}_{j \in [n]}$, $\boldsymbol{v}_{i,\ell^\star} := \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell^\star) \in \mathbb{Z}_L^{mn}$. We denote by $\{i_1, \ldots, i_\kappa\}$ the set of explicitly honest slots in the order the are revealed, and we write $\theta := \min(\kappa^\star, \mu)$.
If $\theta \geq 2$, then do the following:

- If $i = i_1$, then $\boldsymbol{t}_{i,\ell^\star} := \boldsymbol{v}_{i,\ell^\star} + \sum_{t=2}^{\theta} \boldsymbol{u}_t$.
- If $i = i_t$, for $t \in [2, \ldots, \theta]$, then $\boldsymbol{t}_{i,\ell^\star} := \boldsymbol{v}_{i,\ell^\star} - \boldsymbol{u}_t$.
- If $i = i_t$, for $t \in [\theta + 1, \ldots, \kappa^\star]$, then $\boldsymbol{t}_{i,\ell^\star} := \boldsymbol{v}_{i,\ell^\star}$.
- If $i = i_t$, for $t > \kappa^\star$, that means $\kappa > \kappa^\star$, the guess was incorrect. Ends the game and output 0.

If $\theta < 2$, then $\boldsymbol{t}_{i,\ell^\star} := \boldsymbol{v}_{i,\ell^\star}$.
$\boldsymbol{w}_i := (\boldsymbol{0} \| \ldots \| \boldsymbol{0} \| \boldsymbol{x}_i^{j,0} \| \boldsymbol{0} \| \cdots \| \boldsymbol{0}) + \boldsymbol{t}_{i,\ell^\star} \bmod L$
$\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$
Return $\mathsf{ct}_i$

**Fig. 8.** Games for the proof of Lemma 3.8. We say the guess $\kappa^\star$ is correct if it equals the number of explicitly honest slots.

*Proof.* The proof uses a sequence of hybrid games defined in Fig. 8. Note that $\mathsf{G}_{0.0}^\star$ is the same as $\mathsf{G}_0^\star$, and $\mathsf{G}_{0.n}^\star$ is the same as $\mathsf{G}_1^\star$.

We show that for all $\mu \in [n]$ there exists a PPT adversary $\mathcal{B}_{0.\mu-1}$ such that:

$$|\Pr[\mathsf{G}^\star_{0.\mu-1} = 1] - \Pr[\mathsf{G}^\star_{0.\mu} = 1] \leq n(n-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_{0.\mu-1}}(\lambda).$$

First, we show we show that there exists a PPT adversary $\mathcal{B}'_0$ such that:

$$|\Pr[\mathsf{G}^\star_0 = 1] - \Pr[\mathsf{G}^\star_{0.2} = 1] \leq n(n-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}'_0}(\lambda).$$

Note that we go directly from $\mathsf{G}^\star_0$ to $\mathsf{G}^\star_{0.2}$ since the games $\mathsf{G}^\star_0$ and $\mathsf{G}^\star_{0.1}$ are the same. If $\kappa^\star < 2$, the two games $\mathsf{G}^\star_0$ and $\mathsf{G}^\star_{0.2}$ are the same. We consider the case where $\kappa^\star \geq 2$.

The simulation guesses the first and second explicitly honest slots $i^\star$ and $j^\star$, by sampling random $i^\star, j^\star \leftarrow [n]$, with $i^\star < j^\star$. This guessing incurs a security loss of $\frac{n(n-1)}{2}$. If the guess is incorrect, then, the simulation ends and returns 0. If the guess is correct, then, we can use the security of the $\mathsf{PRF}$ on the key $\mathsf{K}_{i^\star,j^\star}$ to switch $\mathsf{PRF}_{\mathsf{K}_{i^\star,j^\star}}(\ell^\star)$ to a uniformly random value $\mathsf{RF}(\ell^\star)$ over $\mathbb{Z}_L^{mn}$. Then, we argue that $\mathsf{RF}(\ell^\star)$ is identically distributed to $\mathsf{RF}(\ell^\star) + \boldsymbol{u}_2$, where $\boldsymbol{u}_2 \leftarrow \mathbb{Z}_L^{mn}$. Note that the former distribution corresponds to $\mathsf{G}^\star_0$, whereas the latter distribution corresponds to $\mathsf{G}^\star_{0.2}$. Then, we switch $\mathsf{RF}(\ell^\star)$ back to $\mathsf{PRF}_{\mathsf{K}_{i^\star,j^\star}}(\ell^\star)$, using the security of the PRF on $\mathsf{K}_{i^\star,j^\star}$ once again.

We can argue similarly that for all $\mu \in [3, \ldots, n]$, there exists a PPT adversary $\mathcal{B}'_{0.\mu-1}$ such that:

$$|\Pr[\mathsf{G}^\star_{0.\mu-1} = 1] - \Pr[\mathsf{G}^\star_{0.\mu} = 1] \leq n(n-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_{0.\mu-1}}(\lambda).$$

This is proved in the same way as above, except the simulation guesses the first and $\theta$'th explicitly honest slots, where $\theta := \min(\kappa^\star, \mu)$, and uses the security of the PRF to add the vector $\boldsymbol{u}_\theta \leftarrow \mathbb{Z}_L^{mn}$.

Summing up for all $\mu \in [2, \ldots, n]$, we obtain a PPT adversary $\mathcal{B}_0$ such that: $|\Pr[\mathsf{G}^\star_0 = 1] - \Pr[\mathsf{G}^\star_1 = 1] \leq n(n-1)^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_0}(\lambda)$.

□

**Lemma 3.9 (Transition from $\mathsf{G}^\star_1$ to $\mathsf{G}^\star_2$).** *There exists a PPT adversary $\mathcal{B}_1$ such that* $\mathsf{Win}^{\mathsf{G}^\star_1}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}^\star_2}_{\mathcal{A}}(\lambda, n) \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_1}(\lambda)$.

*Proof.* We build a PPT adversary $\mathcal{B}_1$ such that:

$$\Pr\left[\mathsf{G}^\star_1(\lambda, n, \mathcal{A}) = 1\right] - \Pr\left[\mathsf{G}^\star_2(\lambda, n, \mathcal{A}) = 1\right] \leq \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}_1}(\lambda).$$

Given $\mathsf{pp}_{\mathsf{ipfe}}$ from its own experiment, $\mathcal{B}_1$ samples $\kappa^\star \leftarrow \{0, \ldots, n\}$ and $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$. If $\kappa^\star \leq 2$, then $\mathcal{B}_1$ samples $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and simulate the game $\mathsf{G}^\star_1(\lambda, n, \mathcal{A})$ as described in Fig. 7 (note that $\mathsf{G}^\star_1$ is the same as $\mathsf{G}^\star_2$ when $\kappa^\star < 2$).

If $\kappa^\star \geq 2$, $\mathcal{B}_1$ does the following. It answers the queries $\mathsf{QCor}$ and $\mathsf{QEnc}$ using $\{\mathsf{sk}_i\}_{i\in[n]}$, as described in Fig. 7. Whenever $\mathcal{A}$ queries $\mathsf{QKeyD}$ on input $\boldsymbol{y}$, $\mathcal{B}_1$ queries its own oracle on the same input, and forwards the answer to $\mathcal{A}$.

Whenever $\mathcal{A}$ queries $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$, $\mathcal{B}_1$ computes the vectors $\boldsymbol{t}_{i,\ell^\star}$, using $\mathsf{sk}_i$ and computing the vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\kappa^\star}$ on the fly, as described in Fig. 7. Then, if the slot $i$ queried is not the $\kappa^\star$'th explicitly honest slot, then $\mathcal{B}_1$ computes $\boldsymbol{w}_i := (\boldsymbol{0}\|\ldots\|\boldsymbol{0}\|\boldsymbol{x}_i^{j,0}\|\boldsymbol{0}\|\ldots\|\boldsymbol{0}) + \boldsymbol{t}_{i,\ell^\star} \mod L$, and returns $\mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{w}_i)$ to $\mathcal{A}$. If $i$ is the $\kappa^\star$'th explicitly honest slot, then $\mathcal{B}_1$ computes $\Delta_{\boldsymbol{x}} := (\boldsymbol{x}_1^{1,1} - \boldsymbol{x}_1^{1,0}, \ldots, \boldsymbol{x}_n^{1,1} - \boldsymbol{x}_n^{1,0})$, where for all explicitly honest slots $i$, $(i, \boldsymbol{x}_i^{1,0}, \boldsymbol{x}_i^{1,1}, \ell^\star)$ is the first query to $\mathsf{QLeftRight}(i, \cdot, \cdot, \ell^\star)$, and for the others slots $i$, $\boldsymbol{x}_i^{1,1} - \boldsymbol{x}_i^{1,0} = \boldsymbol{0} \in \mathbb{Z}^m$. Note that if the guess $\kappa^\star$ is correct, at this point, all the explicitly honest slots have been revealed. Then, it queries its left-right oracle on input $(\boldsymbol{0}, \Delta_{\boldsymbol{x}})$ to get $\mathsf{ct}_i := \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{0})$ or $\mathsf{ct}_i := \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \Delta_{\boldsymbol{x}})$, depending on the experiment $\mathcal{B}_1$ is interacting with. Then, $\mathcal{B}_1$ computes $\boldsymbol{w}_i := (\boldsymbol{0}\|\ldots\|\boldsymbol{0}\|\boldsymbol{x}_i^{j,0}\|\boldsymbol{0}\|\ldots\|\boldsymbol{0}) + \boldsymbol{t}_{i,\ell^\star} \mod L$ and returns $\mathsf{ct}'_i := \mathsf{Add}(\mathsf{ct}_i, \boldsymbol{w}_i)$, which, according to the property from Definition 3.2, is identically distributed to $\mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{w}_i \mod L)$ or $\mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \boldsymbol{w}_i + \Delta_{\boldsymbol{x}} \mod L)$, (again, depending on which experiment $\mathcal{B}_1$ is interacting with). Finally, $\mathcal{B}_1$ returns $\mathsf{ct}'_i$ to $\mathcal{A}$. If $\mathcal{B}_1$ later discovers that the guess $\kappa^\star$ was incorrect (if there are newly revealed explicitly honest slots), then it ends the simulation

and returns 0 to its own experiment. Since we consider pos$^+$-IND security, all honest slots are queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$. Thus, Condition (*) implies that for all $\boldsymbol{y}$ queried to $\mathsf{QKeyD}$, we have $\langle \Delta_{\boldsymbol{x}}, \boldsymbol{y}\rangle = 0$. Moreover, $\|\Delta_{\boldsymbol{x}}\|_\infty < 2X$. Thus, the queries $\mathcal{B}_1$ sends to its left-right oracle are legitimate.

To conclude, we show that when $\mathcal{B}_1$ is interacting with $\textbf{any-IND}_0^{\mathsf{FE}}(\lambda,1,\mathcal{A})$, then it simulates the game $\mathsf{G}_1^\star$, whereas it simulates the game $\mathsf{G}_2^\star$ when it is interacting with $\textbf{any-IND}_1^{\mathsf{FE}}(\lambda,1,\mathcal{A})$. The case $\textbf{any-IND}_0^{\mathsf{FE}}(\lambda,1,\mathcal{A})$ is clear. To prove the case $\textbf{any-IND}_1^{\mathsf{FE}}(\lambda,1,\mathcal{A})$, we consider the vectors $\{\boldsymbol{u}_t\}_{t\in[\kappa^\star]}$, where $\boldsymbol{u}_t \leftarrow \mathbb{Z}_L^{mn}$ for all $t \in [2,\ldots,\kappa^\star]$, and $\boldsymbol{u}_1 := -\sum_{t=2}^{\kappa^\star} \boldsymbol{u}_t$. These are shares of a perfect $\kappa^\star$ out of $\kappa^\star$ secret sharing of $\mathbf{0}$, that is, they are uniformly random conditioned on $\sum_{t\in[\kappa^\star]} \boldsymbol{u}_t = \mathbf{0}$. Thus, the set $\{\boldsymbol{u}_t\}_{t\in[\kappa^\star-1]} \cup \{\boldsymbol{u}_{\kappa^\star} + \Delta_{\boldsymbol{x}}\}$ is a set of shares for a secret sharing of the vector $\Delta_{\boldsymbol{x}}$. Thus, the following distributions are identical:

$$\{\boldsymbol{u}_t\}_{t\in[\kappa^\star-1]} \cup \{\boldsymbol{u}_{\kappa^\star} + \Delta_{\boldsymbol{x}}\}$$

and

$$\{\boldsymbol{u}_t + (\mathbf{0}\|\ldots\|\boldsymbol{x}_{i_t}^{1,1} - \boldsymbol{x}_{i_t}^{1,0}\|\mathbf{0}\|\ldots\|\mathbf{0})\}_{t\in[\kappa^\star]},$$

where for all $t \in [\kappa^\star]$, $\boldsymbol{u}_t \leftarrow \mathbb{Z}_L^{mn}$ such that $\sum_{t\in[\kappa^\star]} \boldsymbol{u}_t = \mathbf{0}$. The uppermost distribution corresponds to the simulation by $\mathcal{B}_1$ when it is interacting with $\textbf{pos}^+\textbf{-IND}_1^{\mathsf{FE}}(\lambda,1,\mathcal{A})$, while the lowermost distribution corresponds to the game $\mathsf{G}_2^\star$.

Summarizing, we have: $\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda,n) \le \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_1}^{\text{any-IND}}(\lambda)$. □

**Lemma 3.10 (Transition from $\mathsf{G}_2^\star$ to $\mathsf{G}_3^\star$).** *There exists a PPT adversary $\mathcal{B}_2$ such that* $\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3^\star}(\lambda,n) \le \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_2}^{\text{any-IND}}(\lambda)$.

*Proof.* We build an adversary $\mathcal{B}_2$ against the any-IND security of $\mathsf{FE}$ as follows. Given $\mathsf{pp}_{\mathsf{ipfe}}$ from its own experiment, $\mathcal{B}_2$ forwards $\mathsf{pp}_{\mathsf{ipfe}}$ to $\mathcal{A}$ and samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$, thanks to which it can answer the queries to $\mathsf{QCor}$ and $\mathsf{QEnc}$. It also samples $\kappa^\star \leftarrow \{0,\ldots,n\}$.

Then, whenever $\mathcal{A}$ queries $\mathsf{QKeyD}$ on input $\boldsymbol{y}$, $\mathcal{B}_2$ queries its own oracle on the same input, and forwards the answer to $\mathcal{A}$. Whenever $\mathcal{A}$ queries $\mathsf{QLeftRight}(i,\boldsymbol{x}_i^{j,0},\boldsymbol{x}_i^{j,1},\ell^\star)$, $\mathcal{B}_2$ computes $\boldsymbol{t}_{i,\ell^\star}$ as described in Fig. 7. If $\kappa^\star < 2$, then $\mathcal{B}_2$ queries its left-right oracle on input $(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0})$ to get

$$\mathsf{ct}_i := \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,b}\|\mathbf{0}\|\ldots\|\mathbf{0})),$$

where $b \in \{0,1\}$, depending on the experiment $\mathcal{B}_2$ is interacting with. If $\kappa^\star \ge 2$, then $\mathcal{B}_2$ queries its left-right oracle on input $(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,0} - \boldsymbol{x}_i^{1,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1} - \boldsymbol{x}_i^{1,1}\|\mathbf{0}\|\ldots\|\mathbf{0})$ to get

$$\mathsf{ct}_i := \mathsf{Enc}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, (\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,b} - \boldsymbol{x}_i^{1,b}\|\mathbf{0}\|\ldots\|\mathbf{0})),$$

where $b \in \{0,1\}$, depending on the experiment $\mathcal{B}_2$ is interacting with. Then, $\mathcal{B}_2$ returns $\mathsf{ct}_i' := \mathsf{Add}(\mathsf{ct}_i, \boldsymbol{t}_{i,\ell^\star})$ to $\mathcal{A}$. Since we are considering pos$^+$-IND security, all honest slots are queried on $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$, thus for $\kappa^\star = \kappa < 2$, for all $\boldsymbol{y}$ queried to $\mathsf{QKeyD}$, Condition (*) implies that $\langle(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle = \langle(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle$ for all queries $(\boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1})$ to $\mathsf{QLeftRight}(i,\cdot,\cdot,\ell)$; for $\kappa^\star = \kappa \ge 2$, it implies $\langle(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,0} - \boldsymbol{x}_i^{1,0}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle = \langle(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,1} - \boldsymbol{x}_i^{1,1}\|\mathbf{0}\|\ldots\|\mathbf{0}), \boldsymbol{y}\rangle$. Moreover, for all $b \in \{0,1\}$, we have $\|(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,b}\|\mathbf{0}\|\ldots\|\mathbf{0})\|_\infty < X$ and $\|(\mathbf{0}\|\ldots\|\mathbf{0}\|\boldsymbol{x}_i^{j,b} - \boldsymbol{x}_i^{1,b}\|\mathbf{0}\|\ldots\|\mathbf{0})\|_\infty < 2X$. Thus, the queries $\mathcal{B}_2$ sends to its left-right oracle are legitimate. When $b = 0$, $\mathcal{B}_2$ simulates $\mathsf{G}_2^\star$, whereas it simulates $\mathsf{G}_3^\star$ when $b = 1$.

□

## 4 From pos$^+$-IND to any-IND Security

In this section, we give a compiler that generically transforms any adt-pos$^+$-IND secure (D)MCFE into an adt-any-IND secure (D)MCFE. Our construction builds upon the compiler from [ABKW19, Section

4.1], which does not support labels. Our technical contribution is to handle multiple labels, many challenge ciphertexts per label and input slots, and adaptive corruptions, without resorting to the random oracle model, as opposed to [ABKW19, Section 4.2]. This is the first generic transformation to support such features, and when combined with our MCFE from Section 3, it gives the first MCFE for inner products whose adt − any-IND security is proven in the standard model. Our construction is given in Fig. 9. It is stated in terms of DMCFE, but a similar transformation works for MCFE.

| | |
|---|---|
| $\underline{\mathsf{Setup}'(1^\lambda, 1^n):}$ | $\underline{\mathsf{KeyDerShare}'(\mathsf{pp}, \mathsf{sk}'_i, f):}$ |
| Return $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ | Parse $\mathsf{sk}'_i = (\mathsf{sk}_i, \{\mathsf{k}_{i,j}, \mathsf{k}_{j,i}\}_{j\in[n]})$ |
| | Return $\mathsf{sk}'_{i,f} \leftarrow \mathsf{KeyDerShare}(\mathsf{sk}_i, f)$ |
| $\underline{\mathsf{KeyGen}'(\mathsf{pp}):}$ | |
| $\{\mathsf{sk}_i\}_{i\in[n]} \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ | $\underline{\mathsf{KeyDerComb}'(\mathsf{pp}, \{\mathsf{sk}'_{i,f}\}_{i\in[n]}):}$ |
| For $i, j \in [n] : \mathsf{k}_{i,j} \leftarrow \{0,1\}^\lambda$ | $\mathsf{sk}_f := \mathsf{KeyDerComb}(\mathsf{pp}, \{\mathsf{sk}'_{i,f}\}_{i\in[n]})$ |
| Return $\{\mathsf{sk}'_i = (\mathsf{sk}_i, \{\mathsf{k}_{i,j}, \mathsf{k}_{j,i}\}_{j\in[n]})\}_{i\in[n]}$ | Return $\mathsf{sk}_f$ |
| $\underline{\mathsf{Enc}'(\mathsf{pp}, \mathsf{sk}'_i, x_i, \ell):}$ | $\underline{\mathsf{Dec}'(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}''_1, \ldots, \mathsf{ct}''_n):}$ |
| Parse $\mathsf{sk}'_i = (\mathsf{sk}_i, \{\mathsf{k}_{i,j}, \mathsf{k}_{j,i}\}_{j\in[n]})$ | Parse $\{\mathsf{ct}''_i = (\mathsf{ct}'_i, \{\mathsf{k}_{j,i}(\ell)\}_{j\in[n]})\}_{i\in[n]}$ |
| $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, x_i)$ | For $i \in [n]$ : |
| For all $j \in [n] : \mathsf{k}_{i,j}(\ell) := \mathsf{PRF}_{\mathsf{k}_{i,j}}(\ell)$ | $\quad \mathsf{K}_i(\ell) = \oplus_{j\in[n]} \mathsf{k}_{i,j}(\ell)$ |
| $\mathsf{K}_i(\ell) := \oplus_{j\in[n]} \mathsf{k}_{i,j}(\ell)$ | $\quad \mathsf{ct}_i \leftarrow \mathsf{Dec}_{\mathsf{SE}}(\mathsf{K}_i(\ell), \mathsf{ct}'_i)$ |
| $\mathsf{ct}'_i \leftarrow \mathsf{Enc}_{\mathsf{SE}}(\mathsf{K}_i(\ell), \mathsf{ct}_i)$ | Return $\mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$. |
| Return $(\mathsf{ct}'_i, \{\mathsf{k}_{j,i}(\ell)\}_{j\in[n]})$ | |

**Fig. 9.** Compiler from an xx-pos$^+$-IND DMCFE, DMCFE, into an xx-any-IND DMCFE, DMCFE$'$, using an IND-CPA symmetric-key encryption scheme SE.

**Theorem 4.1 (Security).** *Let the tuple* $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDerShare}, \mathsf{KeyDerComb}, \mathsf{Enc}, \mathsf{Dec})$ *be an* adt-pos$^+$*-IND-secure DMCFE scheme for a family of functions* $\mathcal{F}$*. Let* $\mathsf{SE} = (\mathsf{Enc}_{\mathsf{SE}}, \mathsf{Dec}_{\mathsf{SE}})$ *be an IND-CPA symmetric-key encryption scheme. Let* PRF *be a pseudorandom function. Then the DMCFE scheme* $\mathsf{DMCFE}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{KeyDerShare}', \mathsf{KeyDerComb}', \mathsf{Enc}', \mathsf{Dec}')$ *described in Fig. 9 is* adt-any*-IND secure. Namely, for any PPT adversary* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}$*,* $\mathcal{B}'$*, and* $\mathcal{B}''$ *such that:*

$$\mathsf{Adv}^{\text{adt-any-IND}}_{\mathsf{DMCFE}',\mathcal{A}}(\lambda, n) \leq q_{\mathsf{Enc}} \cdot \mathsf{Adv}^{\text{adt-pos}^+\text{-IND}}_{\mathsf{DMCFE},\mathcal{B}}(\lambda, n) + q_{\mathsf{Enc}} n^2 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{SE},\mathcal{B}'}(\lambda) + 2 q_{\mathsf{Enc}} n^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda),$$

*where* $q_{\mathsf{Enc}}$ *denotes the number of distinct labels queried to* QLeftRight*.*

*Proof.* WLOG, we can consider the security where only one label is queried to QLeftRight, and that label is not queried to QEnc. Namely, we show there exist PPT adversaries $\mathcal{B}$, $\mathcal{B}'$ and $\mathcal{B}''$ such that:

$$\mathsf{Adv}^{\text{adt-any-IND-1-label}}_{\mathsf{DMCFE}',\mathcal{A}}(\lambda, n) \leq \mathsf{Adv}^{\text{adt-pos}^+\text{-IND}}_{\mathsf{DMCFE},\mathcal{B}}(\lambda, n) + n^2 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{SE},\mathcal{B}'}(\lambda) + 2n^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda).$$

The theorem follows from Lemma 2.5 (from one to many labels). We call $\ell^\star$ the unique label queried to QLeftRight (if QLeftRight is not queried, the security follows trivially).

Intuitively, the proof uses the adt-pos$^+$-IND security of DMCFE for the case where all honest slots are queried to $\mathsf{QLeftRight}(\cdot, \cdot, \cdot, \ell^\star)$, and the security of the PRF together with the IND-CPA security of SE for the case where not all honest slots are queried to $\mathsf{QLeftRight}(\cdot, \cdot, \cdot, \ell^\star)$.

Formally, for all $b \in \{0,1\}$, we define $\mathsf{G}_b^\star$ as adt-yy-IND$_1^{\mathsf{DMCFE}'}(\lambda, n, \mathcal{A})$, except the game guesses an honest slot that is not going to be queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$, by sampling uniformly at random $i^\star \leftarrow \{0,...,n\}$, where $i^\star = 0$ means that all honest slots are queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$. The output of $\mathsf{G}_b^\star$ is the same adt-yy-IND$_b^{\mathsf{DMCFE}'}(\lambda, n, \mathcal{A})$, unless the guess is unsuccessful, in which case, $\mathsf{G}_b^\star$ outputs 0. Clearly, we have $\Pr[\mathsf{G}_b^\star(\lambda, n, \mathcal{A}) = 1] = \frac{1}{n+1} \cdot \Pr[\text{adt-yy-IND}_b^{\mathsf{DMCFE}'}(\lambda, n, \mathcal{A}) = 1]$.

When $i^\star = 0$, we can rely on the adt-pos$^+$-IND security of $\mathsf{DMCFE}$. Namely, we have a PPT adversary $\mathcal{B}$ such that:

$$\big|\Pr[\mathsf{G}_0^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = 0] - \Pr[\mathsf{G}_1^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = 0]\big| \leq \mathsf{Adv}_{\mathsf{DMCFE},\mathcal{B}}^{\text{adt-pos}^+\text{-IND}}(\lambda, n).$$

For all $j \in [n]$, we prove that there exist PPT adversaries $\mathcal{B}'$ and $\mathcal{B}''$ such that:

$$\big|\Pr[\mathsf{G}_0^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j] - \Pr[\mathsf{G}_1^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j]\big| \leq n \cdot \mathsf{Adv}_{\mathsf{SE},\mathcal{B}'}^{\text{IND-CPA}}(\lambda, n) + 2n \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda, n).$$

To prove the statement above, we use the fact that if there is a query $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$ with $\boldsymbol{x}_i^{j,0} \neq \boldsymbol{x}_i^{j,1}$, then the slot $i \in [n]$ cannot be corrupted without violating the Condition (*) from the security definition given in Definition 2.2. We call such a slot explicitly honest, and such a query explicitly honest. We define hybrid games $\mathsf{H}_\rho$ for all $\rho \in \{0, \ldots, n\}$, defined as $\mathsf{G}_0^\star$, except that every explicitly honest query $\mathsf{QLeftRight}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell^\star)$ is answered by $\mathsf{Enc}'(\mathsf{pp}, \mathsf{sk}_i', \boldsymbol{x}_i^{j,1}, \ell^\star)$ if $i \leq \rho$, and is answered by $\mathsf{Enc}'(\mathsf{pp}, \mathsf{sk}_i', \boldsymbol{x}_i^{j,0}, \ell^\star)$ if $i > \rho$. The game $\mathsf{H}_0$ is the same as $\mathsf{G}_0^\star$, and $\mathsf{H}_n$ is the same as $\mathsf{G}_1^\star$. We prove that for all $j \in [n]$, for all $\rho \in [n]$, there exist PPT adversaries $\mathcal{B}_\rho$ and $\mathcal{B}_\rho'$ such that:

$$\big|\Pr[\mathsf{H}_{\rho-1}(\lambda, n, \mathcal{A}) = 1 | i^\star = j] - \Pr[\mathsf{H}_\rho(\lambda, n, \mathcal{A}) = 1 | i^\star = j]\big| \leq \mathsf{Adv}_{\mathsf{SE},\mathcal{B}_\rho}^{\text{IND-CPA}}(\lambda, n) + 2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_\rho'}(\lambda, n).$$

The transition from $\mathsf{H}_{\rho-1}^\star$ and $\mathsf{H}_{\rho-1}^\star$ is justified as follows. If the slot $\rho$ is never queried on an explicitly honest query, then the two games are the same by definition. Otherwise, we use the security of the $\mathsf{PRF}$ to switch the key $\mathsf{k}_{\rho,i^\star}(\ell^\star)$ to uniformly random (note that we can do so since the slots $\rho$ and $i^\star$ are known beforehand by the reduction). If the guess $i^\star$ is correct (i.e $i^\star$ is honest but never queried to $\mathsf{QLeftRight}$), then the key $\mathsf{k}_{\rho,i^\star}(\ell^\star) := \mathsf{PRF}_{\mathsf{k}_{\rho,i^\star}}(\ell^\star)$ only appears in the output $\mathsf{QLeftRight}(\rho,\cdot,\cdot,\ell^\star)$. So, for these challenge ciphertexts, we have a uniformly random key $\mathsf{K}_\rho(\ell^\star)$, which allows us to use the IND-CPA security of $\mathsf{SE}$, and changes encryption of $\boldsymbol{x}_\rho^{j,0}$ as in $\mathsf{G}_{\rho-1}^\star$ into encryption of $\boldsymbol{x}_\rho^{j,1}$, as in $\mathsf{G}_\rho^\star$. Then we switch back the key $\mathsf{k}_{\rho,i^\star}$ from uniformly random to pseudo-random, using the security of the $\mathsf{PRF}$ once again. Summarizing, we have:

$$\Pr[\mathsf{H}_{\rho-1}^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j] - \Pr[\mathsf{H}_\rho^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j] = \mathsf{Adv}_{\mathsf{SE},\mathcal{B}_\rho}^{\text{IND-CPA}}(\lambda, n) + 2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_\rho'}(\lambda, n).$$

Summing up for all $\rho \in [n]$, we obtain the following for all $j \in [n]$:

$$\big|\Pr[\mathsf{G}_0^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j] - \Pr[\mathsf{G}_1^\star(\lambda, n, \mathcal{A}) = 1 | i^\star = j]\big| \leq n \cdot \mathsf{Adv}_{\mathsf{SE},\mathcal{B}'}^{\text{IND-CPA}}(\lambda, n) + 2n \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda, n).$$

Thus, we have:

$$\begin{aligned}
&\big|\Pr[\mathsf{G}_0^\star(\lambda, n, \mathcal{A}) = 1] - \Pr[\mathsf{G}_1^\star(\lambda, n, \mathcal{A}) = 1]\big| \\
&\leq \frac{1}{n+1}\mathsf{Adv}_{\mathsf{DMCFE},\mathcal{B}}^{\text{adt-pos}^+\text{-IND}}(\lambda, n) + \frac{n^2}{n+1} \cdot \mathsf{Adv}_{\mathsf{SE},\mathcal{B}'}^{\text{IND-CPA}}(\lambda, n) + \frac{2n^2}{n+1} \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda, n) \ .
\end{aligned}$$

Therefore, we obtain:

$$\begin{aligned}
&\big|\Pr[\text{adt-yy-IND}_0^{\mathsf{DMCFE}'}(\lambda, n, \mathcal{A}) = 1] - \Pr[\text{adt-yy-IND}_1^{\mathsf{DMCFE}'}(\lambda, n, \mathcal{A}) = 1]\big| \\
&\leq \mathsf{Adv}_{\mathsf{DMCFE},\mathcal{B}}^{\text{adt-pos}^+\text{-IND}}(\lambda, n) + n^2 \cdot \mathsf{Adv}_{\mathsf{SE},\mathcal{B}'}^{\text{IND-CPA}}(\lambda, n) + 2n^2 \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}''}(\lambda, n) \ .
\end{aligned}$$

□

## 5 Decentralized Multi-Client Function Encryption

In this section, we modify the generic construction of Section 3 to make it decentralized. Since our construction makes generic use of any single-input inner-product FE, we cannot directly use directly the transformation from [ABKW19], because the master secret key $\mathsf{msk}$ may be arbitrary, and not necessarily the concatenation of the parties' secret keys $\mathsf{sk}_i$ (for $i \in [n]$), as required by [ABKW19]. Moreover, the functional decryption keys $\mathsf{sk}_f$ may not be computed just from $\mathsf{sk}_i$. We present a different generic transformation that decentralizes any MCFE assuming the underlying single-input FE has an additional structural property, that is fulfilled by most known constructions of single-input inner FE for inner products. This property is called special key derivation (see below), and is very similar to special key derivation for MCFE defined in [ABKW19].

We obtain a stronger security notion than [ABKW19], in the sense that the adversary is able to corrupt functional decryption key share by share, and can try to infer information from incomplete functional decryption keys (see Remark 2.8 for more details on the difference between the security notions). This is the same security as in the original DMCFE from [CDG+18], except we achieve the decentralization with no extra assumption, simply by performing an additive secret sharing of the master secret key of the underlying single-input FE.

**Definition 5.1 (FE with Special Key Derivation).** *Let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key FE scheme for the inner product functionality* $\mathcal{F}^{\mathsf{ip}}_{\rho}$*, where* $\rho = (\mathbb{Z}, 1, n \cdot m, X, Y)$ *where* $n, m, X, Y$ *are positive integers.* $\mathsf{FE}$ *is said to have special key derivation modulo* $M$ *if:*

- *The algorithm* $\mathsf{KeyGen}(\mathsf{pp})$ *generates a master secret key of the form* $\mathsf{msk} := \mathbf{U} \in \mathbb{Z}_M^{\kappa \times mn}$*, for some constant* $\kappa$ *(which can depend on* $\mathsf{pp}$*).*
- $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{KeyDer}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{y})$ *outputs* $\mathsf{sk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{U} \cdot \boldsymbol{y} \in \mathbb{Z}_M^{\kappa})$.

For our security proof, we require $M$ to be a prime number.

**Instantiations.** All the stateless IPFE constructions in [ALS16] satisfy the special key derivation property. More precisely, the DDH construction has special key derivation modulo $p$, the prime order of the underlying cyclic group, and $\kappa = 2$ (using notations from [ALS16], the matrix $\mathbf{U}$ is defined by $U_{1,i} = s_i$ and $U_{2,i} = t_i$). The Paillier and LWE constructions have special key derivation modulo any large enough prime number $M$ so that $\mathbf{U} \cdot \boldsymbol{y}$ is the same modulo $M$ and over the integers with overwhelming probability over the generation of $\mathsf{msk}$. For Paillier, $\kappa = 1$ and $U_{1,i} = s_i$, while for LWE, $\kappa = m$ and $\mathbf{U} = \mathbf{Z}$ (using notations from [ALS16]).

**Construction.** The construction is provided in Fig. 10.

**Correctness.** The only remaining part of correctness to be proven for the scheme in Fig. 10 is to show that the key computed by the algorithms $\mathsf{KeyDerShare}$ and $\mathsf{KeyDerComb}$ corresponds to the one that would have been computed by $\mathsf{KeyDer}$. This follows from the following fact:

$$\mathsf{sk}_{\boldsymbol{y}} = \sum_{i=1}^{n} \mathsf{sk}_{i,\boldsymbol{y}} = \sum_{i=1}^{n} \mathbf{U}_i \cdot \boldsymbol{y} = \mathbf{U} \cdot \boldsymbol{y} \ .$$

**Theorem 5.2 (adt-pos$^+$-IND-security).** *If the FE scheme* $\mathsf{FE} = (\mathsf{Setup}_{\mathsf{ipfe}}, \mathsf{KeyGen}_{\mathsf{ipfe}}, \mathsf{KeyDer}_{\mathsf{ipfe}}, \mathsf{Enc}_{\mathsf{ipfe}}, \mathsf{Dec}_{\mathsf{ipfe}})$ *is an* any*-IND-secure FE scheme for the inner product functionality defined as* $\mathcal{F}^{\mathsf{ip}}_{\rho_{\mathsf{ipfe}}}, \rho_{\mathsf{ipfe}} = (\mathbb{Z}, 1, m, 2X, Y)$*, if* $\mathsf{FE}$ *has the special key derivation property modulo the prime number* $M$*, and if* $\mathsf{PRF}$ *is secure, then* $\mathsf{DMCFE}$ *from Fig. 10 is* adt-pos$^+$*-IND-secure for the functionality defined as* $\mathcal{F}^{\mathsf{ip}}_{\rho}, \rho = (\mathbb{Z}, n, m, X, Y)$*. Namely, for any PPT adversary* $\mathcal{A}$*, there exist PPT adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *such that:*

$$\mathsf{Adv}^{\text{adt-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq 2n^2(n-1)q_{\mathsf{Enc}} \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(\lambda) + 2q_{\mathsf{Enc}} \cdot \mathsf{Adv}^{\text{any-IND}}_{\mathsf{FE},\mathcal{B}'}(\lambda),$$

*where* $q_{\mathsf{Enc}}$ *denotes the number of distinct labels queried to* $\mathsf{QLeftRight}$.

$\underline{\mathsf{KeyGen}(\mathsf{pp}):}$

$(\mathsf{msk}_{\mathsf{ipfe}}, \mathsf{mpk}_{\mathsf{ipfe}}) \leftarrow \mathsf{KeyGen}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}); \mathsf{msk} := \mathsf{msk}_{\mathsf{ipfe}} := \mathbf{U} \in \mathbb{Z}_M^{\kappa \times mn}$

For $i \in [n]$, $j > i : \mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$

For $i \in [n-1] : \mathbf{U}_i \leftarrow \mathbb{Z}_M^{\kappa \times mn}$

$\mathbf{U}_n := \mathbf{U} - \sum_{i=1}^{n-1} \mathbf{U}_i \in \mathbb{Z}_M^{\kappa \times mn}$

Return $\{\mathsf{sk}_i = (\mathsf{mpk}_{\mathsf{ipfe}}, \mathbf{U}_i, \{\mathsf{K}_{i,j}\}_{j \in [n]})\}_{i \in [n]}$ and $\mathsf{msk}$

$\underline{\mathsf{KeyDerShare}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{y} \in \mathcal{R}^{mn}):}$

Return $\mathsf{sk}_{i,\boldsymbol{y}} := \mathbf{U}_i \cdot \boldsymbol{y} \in \mathbb{Z}_M^\kappa$

$\underline{\mathsf{KeyDerComb}(\mathsf{pp}, \mathsf{sk}_{1,\boldsymbol{y}}, \ldots, \mathsf{sk}_{mn,\boldsymbol{y}}):}$

Return $\mathsf{sk}_{\boldsymbol{y}} := \sum_{i=1}^{n} \mathsf{sk}_{i,\boldsymbol{y}} \in \mathbb{Z}_M^\kappa$

**Fig. 10.** Algorithms KeyGen, KeyDerShare and KeyDerComb making the inner-product MCFE from Fig. 5 a DMCFE, assuming that $\mathsf{FE} := (\mathsf{Setup}^{\mathsf{ipfe}}, \mathsf{Enc}^{\mathsf{ipfe}}, \mathsf{KeyDer}^{\mathsf{ipfe}}, \mathsf{Dec}^{\mathsf{ipfe}})$ has the special key derivation property modulo a prime number $M$.

$\underline{\mathsf{G}_0, \mathsf{G}_3}$:

$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$

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

Output: $\alpha$ if Condition (*) is satisfied, or a 0 otherwise.

$\underline{\mathsf{QCor}(i):}$

Return $\mathsf{sk}_i = (\mathbf{U}_i, \{\mathsf{K}_{i,j}\}_{j \in [n]})$

$\underline{\mathsf{QKeyD}(\boldsymbol{y}):}$

For any $i \in [n]$, $\mathsf{sk}_{i,\boldsymbol{y}} := \mathbf{U}_i \cdot \boldsymbol{y} \in \mathbb{Z}_M^\kappa$

Return $\{\mathsf{sk}_{i,\boldsymbol{y}}\}_{i \in [n]}$

$\underline{\mathsf{G}_0, \mathsf{G}_1, \boxed{\mathsf{G}_2, \mathsf{G}_3}}$:

$\underline{\mathsf{QEnc}(i, \boldsymbol{x}_i^{j,0}, \boldsymbol{x}_i^{j,1}, \ell):}$

$\boldsymbol{x}_i^j := \boldsymbol{x}_i^{j,0}$

$\boxed{\boldsymbol{x}_i^j := \boldsymbol{x}_i^{j,1}}$

Return $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_i, \boldsymbol{x}_i^j, \ell)$

$\underline{\mathsf{G}_1, \mathsf{G}_2}$:

$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ except the $\{\mathbf{U}_i\}_{i \in [n]}$ are not generated

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

$\forall i \in [n], S_i := \emptyset$

Output: $\alpha$ if Condition (*) is satisfied, or 0 otherwise.

$\underline{\mathsf{QCor}(i):}$

Pick $\mathbf{U}_i$ uniformly under the constraint $\forall \boldsymbol{y} \in S_i,\ \mathsf{sk}_{i,\boldsymbol{y}} = \mathbf{U}_i \cdot \boldsymbol{y}$

Return $\mathsf{sk}_i := (\mathbf{U}_i, \{\mathsf{K}_{i,j}\}_{j \in [n]})$

$\underline{\mathsf{QKeyD}(\boldsymbol{y}, i):}$

Add $\boldsymbol{y}$ to $S_i$. Then, do the following.

- If $i \in \mathcal{CS}$, return $\mathsf{sk}_{i,\boldsymbol{y}} := \mathbf{U}_i \cdot \boldsymbol{y}$.
- If $\boldsymbol{y} \in S_j$ for all $j \in [n] \setminus \{i\}$, then compute $\mathsf{sk}_{\boldsymbol{y}} := \mathbf{U} \cdot \boldsymbol{y} = \mathsf{KeyDer}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{msk}_{\mathsf{ipfe}}, \boldsymbol{y})$ and return $\mathsf{sk}_{i,\boldsymbol{y}} = \mathsf{sk}_{\boldsymbol{y}} - \sum_{j \in [n] \setminus \{i\}} \mathsf{sk}_{j,\boldsymbol{y}}$.
- If $\boldsymbol{y} \in \mathrm{Vect}(S_i)$, with $\{\mu_{\boldsymbol{y}'} \in \mathbb{Z}_M\}_{\boldsymbol{y}' \in S_i}$ s.t. $\boldsymbol{y} = \sum_{\boldsymbol{y}' \in S_i} \mu_{\boldsymbol{y}'} \cdot \boldsymbol{y}'$, return $\mathsf{sk}_{i,\boldsymbol{y}} := \sum_{\boldsymbol{y}' \in S_i} \mu_{\boldsymbol{y}'} \cdot \mathsf{sk}_{i,\boldsymbol{y}'}$.
- If $\boldsymbol{y} \notin \mathrm{Vect}(S_i)$, return $\mathsf{sk}_{i,\boldsymbol{y}} \leftarrow \mathbb{Z}_M^{\kappa \times nm}$.

**Fig. 11.** Games for the proof of Theorem 5.2. Condition (*) is given in Definition 2.1.

*Proof.* Let $\mathcal{A}$ be a PPT adversary against the security of $\mathsf{MCFE}$. We proceed via a hybrid argument, using the games described in Fig. 11. Note that $\mathsf{G}_0$ corresponds to the game $\text{adt-pos}^{+}\text{-IND}_0^{\mathsf{DMCFE}}(\lambda, n, \mathcal{A})$, and $\mathsf{G}_3$ corresponds to the game $\text{adt-pos}^{+}\text{-IND}_1^{\mathsf{DMCFE}}(\lambda, n, \mathcal{A})$. Thus, we have: $\mathsf{Adv}^{\text{adt-pos}^{+}\text{-IND}}_{\mathsf{DMCFE},\mathcal{A}}(\lambda, n) = \left|\mathsf{Win}^{\mathsf{G}_0}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_3}_{\mathcal{A}}(\lambda, n)\right|$.

**Game $\mathsf{G}_1$.** In game $\mathsf{G}_1$, we change the way the oracles $\mathsf{QCor}$ and $\mathsf{QKeyD}$ answer: instead of using each individual share $\mathbf{U}_i$, they generate their answers on-the-fly to be consistent with previous answers and $\mathsf{KeyDer}^{\mathsf{ipfe}}(\mathsf{pp}_{\mathsf{ipfe}}, \mathsf{msk}_{\mathsf{ipfe}}, \boldsymbol{y})$ in the case of $\mathsf{QKeyD}$. The transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ is justified by linear algebra: the two games are identically distributed. This comes from the fact that in both games, the partial functional decryption keys $\mathsf{sk}_{i,\boldsymbol{y}}$ are uniformly random over $\mathbb{Z}_M^{\kappa \times nm}$ subject to:

- $\forall i \in [n], \boldsymbol{y}, \boldsymbol{y}' \in \mathbb{Z}^{nm}, \alpha \in \mathbb{Z}_M$: $\mathsf{sk}_{i,\boldsymbol{y}} + \alpha \cdot \mathsf{sk}_{i,\boldsymbol{y}'} = \mathsf{sk}_{i,\boldsymbol{y}+\alpha\cdot\boldsymbol{y}'}$, and
- $\forall \boldsymbol{y} \in \mathbb{Z}^{nm}$, $\sum_{i\in[n]} \mathsf{sk}_{i,\boldsymbol{y}} = \mathbf{U}\boldsymbol{y}$.

These keys can be efficiently sampled on-the-fly subject to these constraints, as described in Fig. 11.

**Game $\mathsf{G}_2$.** In game $\mathsf{G}_2$, the challenge ciphertexts encrypts $\boldsymbol{x}_i^{j,1}$ instead of $\boldsymbol{x}_i^{j,0}$. The transition from $\mathsf{G}_1$ to $\mathsf{G}_2$ is justified by the $\text{adt-pos}^{+}\text{-IND}$ security of $\mathsf{MCFE}$ proven in Theorem 3.7.

**Game $\mathsf{G}_3$.** In game $\mathsf{G}_3$, we change back the way the oracles $\mathsf{QCor}$ and $\mathsf{QKeyD}$ answer to match $\text{adt-pos}^{+}\text{-IND}_1^{\mathsf{DMCFE}}(\lambda, n, \mathcal{A})$. The transition from $\mathsf{G}_2$ to $\mathsf{G}_3$ is similar to the one from $\mathsf{G}_1$ to $\mathsf{G}_0$: $\mathsf{G}_3$ and $\mathsf{G}_2$ are perfectly indistinguishable.

Putting everything together, we obtain the theorem. □

**Acknowledgments.** This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 780108 (FENTEC), by the ERC Project aSCEND (H2020 639554), by the French *Programme d'Investissement d'Avenir* under national project RISQ P141580, and by the French FUI project ANBLIC. The third author was partially supported by a Google PhD Fellowship in Privacy and Security. Part of this work was done while the second author was at IBM Research, Yorktown Heights, USA, and the third author was at École normale supérieure, Paris, France, and UC Berkeley, California, USA.

## References

ABDP15. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015*, *LNCS* 9020, pages 733–751. Springer, Heidelberg, March / April 2015.

ABKW19. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC 2019, Part II*, *LNCS* 11443, pages 128–157. Springer, Heidelberg, April 2019.

ACF+18. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO 2018, Part I*, *LNCS* 10991, pages 597–627. Springer, Heidelberg, August 2018.

AGRW17. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT 2017, Part I*, *LNCS* 10210, pages 601–626. Springer, Heidelberg, April / May 2017.

AJ15. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO 2015, Part I*, *LNCS* 9215, pages 308–326. Springer, Heidelberg, August 2015.

ALS16. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO 2016, Part III*, *LNCS* 9816, pages 333–362. Springer, Heidelberg, August 2016.

BGJS15. S. Badrinarayanan, D. Gupta, A. Jain, and A. Sahai. Multi-input functional encryption for unbounded arity functions. In *ASIACRYPT 2015, Part I*, *LNCS* 9452, pages 27–51. Springer, Heidelberg, November / December 2015.

BJK15. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *ASIACRYPT 2015, Part I*, *LNCS* 9452, pages 470–491. Springer, Heidelberg, November / December 2015.

BKS18. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *Journal of Cryptology*, 31(2):434–520, April 2018.

BSW11. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, *LNCS* 6597, pages 253–273. Springer, Heidelberg, March 2011.

CDG+18. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 703–732. Springer, Heidelberg, December 2018.

DOT18. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *PKC 2018, Part II*, *LNCS* 10770, pages 245–277. Springer, Heidelberg, March 2018.

GGG+14. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT 2014*, *LNCS* 8441, pages 578–602. Springer, Heidelberg, May 2014.

KDK11. K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *PETS 2011*, *LNCS* 6794, pages 175–191. Springer, Heidelberg, July 2011.

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