# Multi-Client Functional Encryption for Separable Functions

Michele Ciampi[1], Luisa Sinischalchi[2], and Hendrik Waldner[1]

[1] The University of Edinburgh, Edinburgh, UK
{mciampi,hendrik.waldner}@ed.ac.uk
[2] Aarhus University, Aarhus, Denmark
lsiniscalchi@cs.au.dk

**Abstract.** In this work, we provide a compiler that transforms a *single-input functional encryption* scheme for the class of polynomially bounded circuits into a *multi-client functional encryption* (MCFE) scheme for the class of *separable functions*. An $n$-input function $f$ is called separable if it can be described as a list of polynomially bounded circuits $f^1, \ldots, f^n$ s.t. $f(x_1, \ldots, x_n) = f^1(x_1) + \cdots + f^n(x_n)$ for all $x_1, \ldots, x_n$. Our compiler extends the works of Brakerski et al. [Eurocrypt 2016] and of Komargodski et al. [Eurocrypt 2017] in which a generic compiler is proposed to obtain *multi-input functional encryption* (MIFE) from single-input functional encryption. Our construction achieves the stronger notion of MCFE but for the less generic class of separable functions. Prior to our work, a long line of results has been proposed in the setting of MCFE for the inner-product functionality, which is a special case of a separable function. We also propose a modified version of the notion of *decentralized* MCFE introduced by Chotard et al. [Asiacrypt 2018] that we call *outsourceable mulit-client functional encryption* (OMCFE). Intuitively, the notion of OMCFE makes it possible to distribute the load of the decryption procedure among at most $n$ different entities, which will return decryption shares that can be combined (e.g., additively) thus obtaining the output of the computation. This notion is especially useful in the case of a very resource consuming decryption procedure, while the combine algorithm is non-time consuming. We also show how to extend the presented MCFE protocol to obtain an OMCFE scheme for the same functionality class.

## 1 Introduction

Compared to traditional public-key encryption, functional encryption (FE) [BSW11, O'N10] enables fine-grained access control of encrypted data. In more detail, a FE scheme is equipped with a key generation algorithm that allows the owner of a master secret key to generate a *functional key* $\mathsf{sk}_f$ associated with a function $f$. Using such a functional key $\mathsf{sk}_f$ for the decryption of a ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{sk}, x)$ yields *only* $f(x)$. Roughly speaking, the security of a functional encryption scheme guarantees that no other information is leaked except for $f(x)$. In the classical notion of FE, the decryption algorithm takes as input just a single ciphertext and a functional key for a single-input (one-variable) function. The more general notion of *Multi-Input Functional Encryption (MIFE)* [GGG+14] allows the evaluation of an $n$-input function on $n$ encrypted inputs. In more detail, the decryption algorithm take as an input $n$ ciphertexts $\mathsf{Enc}(\mathsf{sk}, x_1), \ldots, \mathsf{Enc}(\mathsf{sk}, x_n)$ and a functional key for an $n$-input function $f'$ and outputs $f'(x_1, \ldots, x_n)$.

In this work we consider an even stronger notion than MIFE called *multi-client functional encryption (MCFE)* [GGG+14]. In the MCFE setting, each ciphertext $\mathsf{Enc}(\mathsf{sk}_i, x_i)$ is encrypted using a different secret key $\mathsf{sk}_i$. Moreover, an arbitrary set of secret keys $\mathcal{I} = \{\mathsf{sk}_{i_1}, \ldots, \mathsf{sk}_{i_m}\}$ can be leaked to the adversary. The notion of MCFE, intuitively, says that the adversary cannot learn more on the ciphertexts generated using the disclosed keys than what it can learn by evaluating $f'$. Note that the adversary in this case can evaluate $f'$ using any input that he chooses with respect to the positions $i_1, \ldots, i_m$. In general, we can distinguish between two types of MCFE schemes: labeled and unlabeled [ABKW19, ACF+18]. In the labeled case every ciphertext is encrypted under a label $\ell$. A valid decryption requries that the input ciphertexts have been encrypted under the same label (otherwise the decryption procedure generates an invalid output). Our results are proven secure under the stronger notion of security with labels, which also allows the adversary to obtain multiple ciphertexts under the same label. This additional security requirement has been considered since [CDG+18b, ABKW19].

In this work we focus on MCFE for a specific functionality class called *separable functions* [MS08, MAS06]. A separable function is an efficiently computable function $f$ that can be separated into a list of efficiently computable functions $f^1, \ldots, f^n$ s.t. $f(x_1, \ldots, x_n) = f^1(x_1) + \cdots + f^n(x_n)$ for all $x_1, \ldots x_n$, with $x_i$ contained in the domain of $f^i$. This is not restricted to addition but to any group operation, therefore also multiplication, i.e. $f(x_1, \ldots, x_n) = f^1(x_1) \cdot \ldots \cdot f^n(x_n)$ for all $x_1, \ldots x_n$, with $x_i$ contained in the domain of $f^i$. Separable functions are used in many real-world applications, and a MCFE scheme, covering such a functionality class, would enable privacy in these scenarios. For example, consider the problem of counting a specific word $\mathsf{w}$ in $n$ different files, provided by $n$ different parties, that contain sensitive information. In more detail, assume that we have $n$ parties and each party $P_i$ owns its corresponding file and the right to compute the separated function $f^i_{\mathsf{w}}$ over it, which simply counts the number of occurrences of the word $\mathsf{w}$ in its file. If a different party $P_{\mathsf{w}}$ possesses all the encrypted files and a functional key $\mathsf{sk}_{f_{\mathsf{w}}}$, containing all the partial word counting functions $f^i_{\mathsf{w}}$, it can compute the number of occurrences of $\mathsf{w}$ over all the encrypted files. Even if $P_{\mathsf{w}}$ manages to obtain some of the encryption keys the content of the files remains partially hidden[3].

A second scenario where a MCFE scheme can be useful is the aggregation of *SQL-queries*. In this context, the calculation of sums, counting, and computing averages on a certain field contained in multiple ($n$) encrypted tables that can be accessed by different users (with different qualifications), are possible applications. A third motivating example is voting. Here, for simplicity, we consider a *YES/NO* voting setting. Each voter encrypts its vote using its own secret key. To count the vote, a functional decryption key for the function $f_{\mathsf{vote}}$ is generated. This function counts the overall votes by outputting a boolean expression depending on the vote. More precisely, the function $f_{\mathsf{vote}}$ is defined as follows: $f_{\mathsf{vote}}(\mathsf{vote}_1, \ldots, \mathsf{vote}_n) = f^1_{\mathsf{vote}}(\mathsf{vote}_1) +, \ldots, + f^n_{\mathsf{vote}}(\mathsf{vote}_n)$ where $f^i_{\mathsf{vote}}(\mathsf{vote}_i)$ outputs 1 if $\mathsf{vote}_i$ is equal to YES and 0 otherwise[4].

*Decentralized MCFE.* Both, the notions of MIFE and MCFE, subsume the existence of a central trusted authority that generates and distributes the secret and functional keys. This is undesirable in some scenarios,

[3] For example, in the worst case, where the adversary has all but the key $\mathsf{sk}_j$, he should be able to compute the number of times that the word $\mathsf{w}$ appears in $F_i$, but nothing more than this.

[4] This voting function is rather simple but one can think to adopt more complicated ballots and tally-functions

given that an adversarial trusted authority can compromise the security of the MCFE scheme (note that the trusted authority can generate any functional key, hence also the functional key for the identity function). A first step toward removing the trusted authority has been done in [CDG+18a] by introducing the concept of *decentralized multi-client functional encryption* (DMCFE). In this setting, the generation of the functional keys happens in a decentralized way. In this work, we formally consider a notion of DMCFE for the case of separable functions where both, the secret key and the functional key generation happens in a distributed manner.

### 1.1 Our Contribution

In this paper we investigate the feasibility of constructing MCFE for separable functions starting from *any* general-purpose FE scheme. In more detail, we provide a compiler that takes as input any private-key FE scheme and outputs a MCFE scheme for separable functions that is *selectively* secure[5] and supports an a priori bounded (but still polynomial) number of encryption and an unbounded number of $n$-input functional key queries (where $n$ is polynomially related to the security parameter).

We show how to extend the above scheme to the case of *adaptive* security[6] (where the adversary can request an a priori bounded number of encryptions and functional keys at any time). We now state our theorems informally.

**Theorem 1** (informal). *Assuming the existence of an any selective secure private-key FE scheme that supports an a priori bounded number of encryption queries and an unbounded number of functional-key queries, then there exists a fully selective secure MCFE scheme for separable functions that supports a bounded number of encryption queries and an ubounded number of functional-key queries.*

**Theorem 2** (informal). *Assuming the existence of an any adaptive secure private-key FE scheme that supports an a priori bounded number of encryption and functional-key queries, then there exists a fully-secure adaptive MCFE scheme for separable functions that supports a bounded number of encryption queries and functional-key queries.*

We prove our constructions for the so-called *pos*$^+$ security notion [ABG19, CDG+18b]. In a pos$^+$ security game an adversary is required to ask a left-or-right query under a specific label in either every or none position. A second notion called *any* security [ABG19, CDG+18b] allows the adversary to ask a left-or-right encryption query on as many positions as he wants without any restrictions. To achieve the notion of any security, we make use of a slightly modified version of a black box compiler presented in [ABG19] which amplifies any pos$^+$ secure MCFE scheme to an any secure MCFE scheme.

In the next step, we discuss how to modify our constructions in order to obtain a DMCFE scheme for separable functions and prove the following theorem.

**Theorem 3** (informal). *Assuming the existence of any selective (adaptive) secure private-key FE scheme that supports an a priori bounded number of encryptions queries (and a bounded number of functional-key queries), then there exists a fully selective (adaptive) secure DMCFE scheme for separable functions that supports a bounded number of encryption queries (and a bounded number of functional-key queries).*

*Outsourceable MCFE.* As an additional contribution, we introduce a new notion called outsourceable mulit-client functional encryption (OMCFE). Intuitively, the notion of OMCFE makes it possible to outsource the load of the decryption procedure among $n$ different entities. In more detail, let $f$ be the $n$-input separable function that we want to evaluate, then the key-generation algorithm of an OMCFE scheme generates $n$ partial functional keys $\mathsf{sk}_{f,1}, \ldots, \mathsf{sk}_{f,n}$ (one for each input-slot of $f$), instead of generating one functional key

[5] We actually mean static-selective, i.e. the adversary has to submit all its message and corruption queries at the beginning of the game.

[6] We consider adaptive-adaptive security, which means that the adversary is allowed to query all the oracles, i.e. message and corruption oracles, throughout the whole game.

$\mathsf{sk}_f$ for $f$.s Each of the functional keys $\mathsf{sk}_{f,i}$ can be applied on a ciphertext $\mathsf{ct}_{i,\ell}$ (a ciphertext that contains the $i$-th input of the function) to obtain a decryption share $\varphi_{i,\ell}$. An evaluator that has all the $n$ share (one for each input slot), can compute the final output by running a *combine* algorithm taking the shares as an input.

This notion becomes important in the case where the combine algorithm is significantly more efficient than the *partial* decryption procedure. More formally, we require that the computational complexity of the combine algorithm is independent from the computational complexity of the function $f$.

Coming back to the word count example, it is possible to give $\mathsf{sk}_{f^i_{\mathsf{w}}}$ and an encryption of the $i$'th part of a huge file, to an entity $P_i$ (for each $i \in [n]$) and let $P_i$ generate the decryption share by executing the decryption procedure. In this way an evaluator $P_{\mathsf{w}}$, would receive the decryption shares from $P_1, \ldots, P_n$, and executes the (light) combine algorithm to obtain the final output of the computation. The word count example can be also seen as a special case of a class of problems that can be parallelized using the *MapReduce* paradigm [DG08]. This parallelization paradigm consists of a *map* phase which divides the problem into sub-problems and a *reduce* phase which parallelizes the aggregation of the partial solutions. It is easy to see that if the reduce phase consists of addition/multiplication operations then our OMCFE scheme could be particularly useful to implement a layer of privacy on top of these parallelization paradigm.

The security definition of this notion is almost identical to the security definition of MCFE. The main difference is in the correctness (since the key generation algorithm and the decryption algorithm are different). We show how to obtain an OMCFE for the class of separable functions. In particular we have the following informal theorem.

**Theorem 4** (informal). *Assuming the existence of any selective (adaptive) secure private-key FE scheme that supports an a priori bounded number of encryptions queries (and a bounded number of functional-key queries), then there exists a fully selective (adaptive) secure OMCFE scheme for separable functions that supports a bounded number of encryption queries (and a bounded number of functional-key queries).*

**Instantiations.** Our constructions can be instantiated from various assumptions. There exists a general-purpose private-key FE scheme from indistinguishability obfuscation or multilinear maps [BKS18]. We can obtain our adaptive secure MCFE scheme (and the decentralized one) from learning with errors [GKP+13], one-way functions or low-depth pseudorandom generators [GVW12]. In more detail, as already mentioned in [BKS18], based on the results of Ananth et al. [ABSV15] and Brakerski et al. [BS18], it is possible to generically obtain a fully-secure scheme by relying on any selectively secure and message-private functional encryption scheme. This implies that fully secure schemes for any number of encryption and key-generation queries can be based on indistinguishability obfuscation [GGH+13,Wat15], differing-input obfuscation [BCP14, ABG+13], and multilinear maps [GGHZ16]. Besides this, it is possible to construct fully secure schemes for a bounded number ($q$) of encryption and key-generation queries by relying on the Learning with Errors (LWE) assumption (where the length of ciphertexts grows with $q$ and with a bound on the depth of allowed functions) [GKP+13,CVW+18], or on pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with $T$ and with an upper bound on the circuit size of the functions) [GVW12], and based on one-way functions (for $T = 1$) [GVW12].

### 1.2 Overview of our Techniques

**Our Compiler.** We present a compiler that transforms any selectively secure single-input FE scheme $\mathsf{FE}$ into a selectively secure MCFE scheme $\mathsf{MCFE}$ for the class of $n$-input separable functions. We provide an incremental description how our compiler works.

In the setup procedure of $\mathsf{MCFE}$ we run $n$ times the setup of $\mathsf{FE}$ thus obtaining $n$ master secret keys $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$. We define the $i$'th secret key for $\mathsf{MCFE}$ as $\mathsf{sk}_i := \mathsf{msk}_i$ for $i = 1, \ldots, n$, whereas the master secret key of $\mathsf{MCFE}$ is represented by all the secret keys $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}$. To encrypt a message $x_i$ for the position $i$ we simply run the encryption algorithm of $\mathsf{FE}$ using the secret key $\mathsf{sk}_i$ and the message $x_i$ thus obtaining the ciphertext $\mathsf{ct}_i$. To generate a functional key for a separable function $f := \{f^1, \ldots, f^n\}$ the key generation algorithm randomly samples a secret sharing of 0: $r_1 + \cdots + r_n = 0$ (we refer to this values as $r$-values) and runs, using the master secret key $\mathsf{msk}_i$ (which corresponds to $\mathsf{sk}_i$) of $\mathsf{FE}$ the key generation algorithms for $\mathsf{FE}$

to generate a functional key $\mathsf{sk}_{f^i_{r_i}}$ for $f^i_{r_i}$. The function $f^i_{r_i}$ takes as an input $x_i$ and outputs $f^i(x_i) + r_i$. The output of the key generation algorithm is then represented by $\{\mathsf{sk}_{f^1_{r_1}}, \ldots, \mathsf{sk}_{f^n_{r_n}}\}$. The decryption algorithm of MCFE, on input the ciphertext $\mathsf{ct} := \{\mathsf{ct}_1, \ldots, \mathsf{ct}_n\}$ and the functional keys $\{\mathsf{sk}_{f^1_{r_1}}, \ldots, \mathsf{sk}_{f^n_{r_n}}\}$ runs the decryption algorithm for FE on input $\mathsf{sk}_{f^i_{r_i}}$ and $\mathsf{ct}_i$ thus obtaining $\varphi_i$ for $i = 1, \ldots, n$. The output of the decryption procedure is then given by $\varphi_1 + \cdots + \varphi_n$ which is equal to $f(x_1, \ldots, x_n)$ due to the property of $f$ and the way the values $r_1, \ldots, r_n$ are sampled. Intuitively, the security of this scheme comes from the fact that a functional key $\mathsf{sk}_{f^i_{r_i}}$ for FE hides the description of the function[7], hence it hides the value $r_i$. The fact that the value $r_i$ is protected allows us to argue that $\varphi_i$ encrypts the partial output $f^i(x_i)$ (that the adversary is not supposed see). Indeed, $\varphi_i$ can be seen as the one-time pad encryption of $f^i(x_i)$ using the key $r_i$.

We show that for the class of separable functions the described one-time pad encryption is sufficient for several encryption queries. This is possible by exploiting the fact that the security game for functional encryption requires that $f(x^0_1, \ldots, x^0_n) = f(x^1_1, \ldots, x^1_n)$ for all the challenge queries $(x^0_i, x^1_i)$ and all the functional key queries $f$. This means, in the case of separable functions, that $\sum_{i\in[n]} f^i(x^0_i) = \sum_{i\in[n]} f^i(x^1_i)$, which is equivalent to $f^{i^*}(x^0_{i^*}) - f^{i^*}(x^1_{i^*}) = \sum_{i\in[n]\setminus\{i^*\}} f^i(x^1_i) - f^i(x^0_i)$. This restriction enforces the security of the information-theoretic encryption under many queries (we show this using a simple reduction).

To extend our scheme to the labeled setting, we borrow a technique from [KDK11, ABG19] and modify our scheme as follows: Assuming that we know an upper-bound on the number of labels $q$, during the setup phase, we generate $q$ random secret sharings of 0: $t_{1,j} + \cdots + t_{n,j} := 0$, with $j \in [q]$ (we refer to this values as the $t$-values) and the $i$'th secret key now becomes $\mathsf{sk}_i := (\mathsf{msk}_i, \{t_{i,j}\})_{j\in[q]}$. To encrypt a message $x_i$ under the label $j$ the encryptor runs the encryption algorithm of FE on input $\mathsf{sk}_i$ and the concatenation of $x_i$ with $t_{i,j}$, thus obtaining $\mathsf{ct}_{i,j}$.

To generate a functional key for the function $f$ a secret sharing of 0 is generated as before, $r_1 + \cdots + r_n = 0$,, but this time we generate the functional key $\mathsf{sk}_{\tilde{f}^i_{r_i}}$ of FE for the function $\tilde{f}^i_{r_i}$. The function $\tilde{f}^i_{r_i}$ takes as and input $(x_i, t_{i,j}, j)$ and outputs $f^i(x_i) + r_i + t_{i,j}$. The output of the key generation algorithm is then represented by $\{\mathsf{sk}_{\tilde{f}^1_{r_1}}, \ldots, \mathsf{sk}_{\tilde{f}^n_{r_n}}\}$. The decryption procedure for this new scheme works exactly as before. Let $\varphi_{i,j}$ be the output of the decryption algorithm of FE on input $\mathsf{ct}_{i,j}$ (the ciphertext computed with respect to the label $j$) and $\mathsf{sk}_{f^i_{r_i}}$. Intuitively, our new scheme allows encrypting multiple messages under different labels, since the partial decryption $\varphi_{i,j}$ is now encrypted using a fresh one-time pad key which corresponds to the combination of the $r$-value $r_i$ (hidden inside the function) and the $t$-value $t_{i,j}$ (hidden inside the ciphertext) for every label $j$. It is worth noting that even if new $t$-values are generated for each encryption we still need to rely on the $r$-values hidden inside the function since an adversary could use the same ciphertext $\mathsf{ct}_{i,j}$ as the input of multiple functions, and this creates an issue similar to the one discussed above.

Even if this scheme is secure under the generation of multiple encryptions and functional keys it has the drawback that the size of each secret key growths with $q$ (the upper-bound to the number of encryptions). To tackle this problem we borrow a technique from the work of Abdalla et al. [ABG19][8], that allows multiple parties to generate a secret sharing of 0 non-interactively by agreeing on a set (of size $n$) of pseudo-random function (PRF) keys during the setup. We refer to Section 5 for more details. The adaptive $q$-message $q$-function bounded MCFE scheme works in a similar way, the main differences are regarding the size of the ciphertext and the size of the functional keys. For the selective scheme only the size of the functional keys depends on $q$, whereas in the adaptive scheme also the ciphtertexts grow with $q$. The details for this proof can be found in Section 6.

**Decentralized Multi-Client Functional Encryption.** For completeness, we also propose a construction of a DMCFE scheme. In the notion of DMCFE, as proposed in [CDG+18a], the key-generation phase is decentralized in the sense that each secret keys owner should be able to output a *partial functional key* for a function $f$, such that the combination of all these partial functional keys allows the generation of a valid functional key for $f$. Our MCFE scheme seems to be easy translatable into a decentralized one except

[7] We recall that the notion of fully secure FE guarantees also *function hiding*.

[8] This technique has first been used in [KDK11] in the context of privacy-friendly aggregation.

for one detail: the key generation phase of MCFE requires the computation of a new set of $r$-values such that $r_1 + \cdots + r_n = 0$ for each function $f$. To obtain a decentralized scheme, we adopt again the technique proposed in [ABG19] that allows the generation of a secret sharing of 0 in a distributed manner. The idea of decentralizing a MCFE scheme in this way has first been proposed in [ABKW19]. In the definition of DMCFE in this paper, we also consider also distributed setup phase in which the parties interact with each other to obtain the secret keys. This idea has been informally discussed in , but it has never been formally defined before.

**Outsourceable Multi-Client Functional Encryption.** We show that we obtain, with minor modifications to the presented compiler, an OMCFE scheme. The proof works, as already mentioned in the previous sections, by relying on the fact that the values $\varphi_{i,\ell}$ do not reveal any information on the encrypted messages.

*Remark 1.1.* Without loss of generality, in the remainder of this paper, we only refer to the case of additive separability. However, our compiler also works for the case of multiplicative separability. To achieve multiplicative separability all the additive operators need to be replaced by its multiplicative counterparts (i.e. addition with multiplication and subtraction with division). Also the group we need to operate in needs to be changed from an additive group to a multiplicative group, e.g. from $\mathbb{Z}_p$ to $\mathbb{Z}_p^*$

### 1.3 Related Work

Since the introduction of multi-input and multi-client functional encryption [GGG+14] several contributions have been made to provide constructions in these areas. The main difference between the former and the latter is that in a MIFE scheme a single encryption key can be used to generate ciphertexts for every position, whereas in multi-client functional encryption every position is associated with its own encryption key. The main techniques that have been proposed to construct MIFE schemes are "liftings" from single-input functional encryption into the multi-input setting. The first foundational work in this area has been done by Brakerski et al. [BKS16]. In this work, the authors manage to transform a single-input selectively secure functional encryption scheme into a fully adaptively secure multi-input functional encryption scheme which supports a constant number of inputs. In [KS17] the authors, among other results, improve the result of [BKS16] by obtaining a MIFE scheme that supports functions with $2^t = (\log \lambda)^\delta$ inputs, where $0 < \delta < 1$. Both of these transformations require a single-input functional encryption scheme for the class of polynomially bounded circuits as input.

The schemes that cover the class of polynomially bounded circuits can be divided into two categories. The first category is only able to handle a bounded number of plaintexts (a so called message-bounded scheme) and (or) a bounded number of functional keys, whereas the second class is able to handle an unbounded number of queries and functional keys. A construction that falls into the first category is given by Gorbunov, Vaikuntanathan and Wee [GVW12]. Their construction relies only on the existence of one-way functions. Constructions based on the Learning with Errors (LWE) assumption hav been proposed by Goldwasser et al. [GKP+13] and by Chen et al. [CVW+18]. The second construction is based on traitor tracing (instantiated using LWE) as presented in [GKW18].

In the case of unbounded message security most of the known constructions are based on less standard assumptions like [Wat15, GGH+13] that is based on indistinguishable obfuscation, [GGHZ16], based on multilinear maps and [ABG+13, BCP14], based on differing-input obfuscation. All of the mentioned scheme are also covering the functionality class of polynomially bounded circuits.

Beside the class of polynomially bounded circuits, it is also possible to construct multi-input functional encryption schemes for more specific functionality classes, like inner-product. The first multi-input functional encryption scheme for inner-product functions has been provided by Abdalla et al. [AGRW17]. The construction they present relies on pairings. A follow up work [ACF+18] proposes a compiler that takes as input a single-input functional encryption scheme that fulfills some special properties and outputs a MIFE scheme for inner-product functions. This construction does not require pairings and only relies on standard assumptions. It turns out that the construction of Abdalla et al. [ACF+18] also fulfills the stronger notion of multi-client

| | # of inputs | Functions | Setting | Assumptions |
|---|---|---|---|---|
| **Brakerski et al. Eurocrypt 2016** | Constant | Generic | MIFE | SK Single-input FE |
| **Komargodski et al. Eurocrypt 2017** | $\log(\lambda)^\delta$ $0 < \delta < 1$ | Generic | MIFE | SK Single-input FE |
| **Abdalla et al. Crypto 2018** | poly($\lambda$) | Inner-product | (D)MCFE (no labels) | SK Single-input FE for inner-product |
| **Abdalla et al. Asiacrypt 2019** | poly($\lambda$) | Inner-product | (D)MCFE | SK Single-input FE for inner-product |
| **This work** | poly($\lambda$) | Separable functions | (D)MCFE | SK Single-input FE |

Table 1: Comparison with the most relevant compilers. $\lambda$: the security parameter, SK: secret key.

functional encryption (without labels) which has been proven [ABKW19]. In the case of multi-client functional encryption, it can be distinguished between two cases, the labeled and the unlabeled case. Labels enforce an additional restriction on the decryption procedure. Namely, it is only possible to decrypt tuples of ciphertexts that are encrypted under the same label, otherwise the decryption procedure outputs an invalid value. The first labeled scheme for the inner-product functionality has been proposed in [CDG+18a]. Its security is proven in the random oracle model an relies on DDH. The authors also present a decentralized version of their construction, which additionally requires pairings. A follow up work by the same authors [CDG+18b] improves on these results by presenting a compiler that decentralizes a multi-client functional encryption scheme without relying on pairings but on standard assumptions. Another compiler, that only uses information-theoretic arguments, has been presented in [ABKW19]). The most recent work in this area, the work of Abdalla et al. [ABG19] which presents a construction for multi-client functional encryption in the labeled setting based only the existence of a secure public key FE scheme for the inner product functionality (that can be based on standard assumptions) together with another compiler to achieve decentralization. In the Table 1 we provide a short comparison between the most relevant compilers that turn a single-input FE scheme into a MIFE or MCFE scheme.

## 2 Preliminaries

**Notation.** We denote the security parameter with $\lambda \in \mathbb{N}$. A randomized algorithm $\mathcal{A}$ is running in *probabilistic polynomial time* (PPT) if there exists a polynomial $p(\cdot)$ such that for every input $x$ the running time of $\mathcal{A}(x)$ is bounded by $p(|x|)$. We call a function $\text{negl} : \mathbb{N} \to \mathbb{R}^+$ *negligible* if for every positive polynomial $p(\lambda)$ a $\lambda_0 \in \mathbb{N}$ exists, such that for all $\lambda > \lambda_0 : \epsilon(\lambda) < 1/p(\lambda)$. We denote by $[n]$ the set $\{1, \ldots, n\}$ for $n \in \mathbb{N}$. We use "=" to check equality of two different elements (i.e. $a = b$ then...) and ":=" as the assigning operator (e.g. to assign to $a$ the value of $b$ we write $a := b$). A randomized assignment is denoted with $a \leftarrow A$, where $A$ is a randomized algorithm and the randomness used by $A$ is not explicit. If the randomness is explicit we write $a := A(x; r)$ where $x$ is the input and $r$ is the randomness. 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 $\mathcal{A}$. We define the distinguishing advantage between games $\mathsf{G}_0$ and $\mathsf{G}_1$ of an adversary $\mathcal{A}$ in the following way: $\mathsf{Adv}^{\mathsf{G}}_{\mathcal{A}}(\lambda, n) = \left|\mathsf{Win}^{\mathsf{G}_0}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_1}_{\mathcal{A}}(\lambda, n)\right|$. The notation $(-1)^{j<i}$ denotes $-1$ if $j < i$ and 1 otherwise.

### 2.1 Secret-Key Functional Encryption

In this section, we define the notion of secret-key functional encryption (SK-FE) [BS15]. They are an adaption of the indistinguihsable notion from [BSW11, O'N10].

**Definition 2.1 (Secret-Key Functional Encryption).** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a collection of function families (indexed by $\lambda$), where every $f \in \mathcal{F}_\lambda$ is a polynomial time function $f\colon \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. A secret-key functional encryption scheme (SK-FE) for the function family $\mathcal{F}_\lambda$ is a tuple of four algorithms* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(1^\lambda)$**:** *Takes as input a unary representation of the security parameter $\lambda$ and generates a master secret key* $\mathsf{msk}$.

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** *Takes as input the master secret key* $\mathsf{msk}$ *and a function $f \in \mathcal{F}_\lambda$, and outputs a functional key* $\mathsf{sk}_f$.

$\mathsf{Enc}(\mathsf{msk}, x)$**:** *Takes as input the master secret key* $\mathsf{msk}$*, a message $x \in \mathcal{X}_\lambda$ to encrypt, and outputs a ciphertext* $\mathsf{ct}$.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$**:** *Takes as input a functional key* $\mathsf{sk}_f$ *and a ciphertext* $\mathsf{ct}$ *and outputs a value $y \in \mathcal{Y}_\lambda$.*

*A scheme* $\mathsf{FE}$ *is correct, if for all $\lambda \in \mathbb{N}$,* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$*, $f \in \mathcal{F}_\lambda$, $x \in \mathcal{X}_\lambda$, when* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{msk}, x)) = f(x)\right] = 1 \ .$$

We define the security of a SK-FE scheme using a left-or-right oracle. We distinguish between selective and adaptive submission of the encryption challenges. We consider a fully secure SK-FE scheme, which, intuitively, means that the SK-FE scheme guarantees privacy for both, the description of the functions and the encrypted messages. We will recall now the formal definition.

**Definition 2.2 (Full Security of SK-FE).** *Let* $\mathsf{FE}$ *be an SK-FE scheme, $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ a collection of function families indexed by $\lambda$. For* $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$ *and $\beta \in \{0,1\}$, we define the experiment* $\text{xx-FULL}_\beta^{\mathsf{FE}}$ *in Fig. 1, where the oracles are defined as:*

**Left-or-Right oracle** $\mathsf{QLeftRight}(x^0, x^1)$**:** *Outputs* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x^{\beta,j})$ *on a query $(x^0, x^1)$. We denote by $Q_{\mathsf{LeftRight}}$ the set containing the queries $(x^0, x^1)$.*

**Key generation oracle** $\mathsf{QKeyG}(f^0, f^1)$**:** *Outputs* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f^\beta)$ *on a query $(f^0, f^1)$. We denote by $Q_f$ the queries of the form* $\mathsf{QKeyG}(\cdot,\cdot)$.

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

– *For every query $(f^0, f^1)$ to* $\mathsf{QKeyG}$*, and every query $(x^0, x^1) \in Q_{\mathsf{LeftRight}}$, we require that:*

$$f^0(x^0) = f^1(x^1) \ .$$

*We define the advantage of an adversary $\mathcal{A}$ for $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$ in the following way:*

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

*A secret-key functional encryption scheme* $\mathsf{FE}$ *is* xx-FULL *secure, if for any polynomial-time adversary $\mathcal{A}$, there exists a negligible function* $\mathrm{negl}$ *such that:* $\mathsf{Adv}^{\text{xx-FULL}}_{\mathsf{FE},\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)$*. In addition, we call a scheme $q$-message bounded, if $|Q_{\mathsf{LeftRight}}| < q$ and $q$-message-and-key bounded, if $|Q_{\mathsf{LeftRight}}| < q$ and $|Q_f| < q$, with $q = \mathrm{poly}(\lambda)$.*

### 2.2 Multi-Client Functional Encryption

Now, we introduce multi-client functional encryption (MCFE) as in [GGG+14, ABKW19, ABG19]. In a multi-client functional encryption scheme, every client can encrypt its own input (corresponding to a slot) and the evaluation of a functional key is executed over the ciphertexts of all the clients.

**Definition 2.3 (Multi-Client Functional Encryption).** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a collection of function families (indexed by $\lambda$), where every $f \in \mathcal{F}_\lambda$ is a polynomial time function $f\colon \mathcal{X}_{\lambda,1} \times \cdots \times \mathcal{X}_{\lambda,n} \to \mathcal{Y}_\lambda$. Let* $\mathsf{Labels} = \{0,1\}^*$ *or $\{\perp\}$ be a set of labels. A multi-client functional encryption scheme (MCFE) for the function family $\mathcal{F}_\lambda$ supporting $n$ users, is a tuple of four algorithms* $\mathsf{MCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

| $\textbf{sel-FULL}^{\mathsf{FE}}_{\beta}(\lambda, \mathcal{A})$ | $\textbf{ad-FULL}^{\mathsf{FE}}_{\beta}(\lambda, \mathcal{A})$ |
|---|---|
| $Q_{\mathsf{LeftRight}} \leftarrow \mathcal{A}(1^\lambda)$ | $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ |
| $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ | $\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot),\mathsf{QKeyG}(\cdot,\cdot)}(1^\lambda)$ |
| $\mathsf{ct}^j \leftarrow \mathsf{QLeftRight}(x^{j,0}, x^{j,1})$, for all $(x^{j,0}, x^{j,1}) \in Q_{\mathsf{LeftRight}}$ | **Output:** $\alpha$ if Condition (*) is satisfied, or a uniform bit otherwise |
| $\alpha \leftarrow \mathcal{A}^{\mathsf{QKeyG}(\cdot,\cdot)}(\{\mathsf{ct}^j\}_{j\in[Q_{\mathsf{Enc}}]})$ | |
| **Output:** $\alpha$ if Condition (*) is satisfied, or a uniform bit otherwise | |

Fig. 1: Full Security Games for SK-FE

$\mathsf{Setup}(1^\lambda, n)$**:** *Takes as input a unary representation of the security parameter* $\lambda$*, and the number of parties* $n$ *and generates* $n$ *secret keys* $\{\mathsf{sk}_i\}_{i\in[n]}$*, and a master secret key* $\mathsf{msk}$*.*

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** *Takes as input the master secret key* $\mathsf{msk}$ *and a function* $f \in \mathcal{F}_\lambda$*, and outputs a functional key* $\mathsf{sk}_f$*.*

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

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

*A scheme* MCFE *is correct, if for all* $\lambda, n \in \mathbb{N}$*,* $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$*,* $f \in \mathcal{F}_\lambda$*,* $x_i \in \mathcal{X}_{\lambda,i}$*, when* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$*, we have*

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

A scheme can either be *without labels*, in this case $\mathsf{Labels} = \{\bot\}$ or *with labels/labeled*, where $\mathsf{Labels} = \{0,1\}^*$. In this work, we only consider schemes that are labeled, i.e. $\mathsf{Labels} = \{0,1\}^*$. Where the latter case implies the former.

The security definition is the initial definition from Goldwasser et al. [GGG+14], whereas we also allow the adversary to determine under which label he wants to query the left-or-right oracle and, in addition, we give the adversary access to an encryption oracle. Besides this, we also allow the adversary to query a single label several times. This security definition has initially been considered in [CDG+18b, ABG19]. As also noted in [ABKW19, ABG19] the security model of multi-client functional encryption is similar to the security model of standard multi-input functional encryption, whereas in the latter only a single master secret key $\mathsf{msk}$ is used to generate encryptions for every slot $i$. In comparison to the standard multi-input functional encryption model, we also consider static and adaptive corruption of the different slots and selective and adaptive left-or-right and encryption oracle queries in the multi-client case. In more detail, in the selective case the adversary is required to ask all his left-or-right, encryption and corruption queries in the beginning of the game. In the adaptive case, the adversary is allowed to ask left-or-right, encryption and corruption queries throughout the whole game.

**Definition 2.4 (Security of MCFE).** *Let* MCFE *be an MCFE scheme,* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *a collection of function families indexed by* $\lambda$ *and* $\mathsf{Labels}$ *a label set. For* $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$*,* $\mathrm{yy} \in \{\mathrm{pos}^+, \mathrm{any}\}$ *and* $\beta \in \{0,1\}$*, we define the experiment* $\text{sel-yy-IND}^{\mathsf{MCFE}}_{\beta}$ *in Fig. 2 and* $\text{ad-yy-IND}^{\mathsf{MCFE}}_{\beta}$ *in Fig. 3, 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-or-Right oracle** $\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$**:** *Outputs* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{sk}_i, x_i^\beta, \ell)$ *on a query* $(i, x_i^0, x_i^1, \ell)$*. We denote the queries of the form* $\mathsf{QLeftRight}(i, \cdot, \cdot, \ell)$ *by* $Q_{i,\ell}$ *and the set of queried labels by* $QL$*.*

**Encryption oracle** $\mathsf{QEnc}(i, x_i, \ell)$ *Outputs* $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{sk}_i, x_i, \ell)$ *on a query* $(i, x_i, \ell)$*. We denote the queries of the form* $\mathsf{QEnc}(i, \cdot, \ell)$ *by* $Q'_{i,\ell}$ *and the set of queried labels by* $QL'$*.*

**Key generation oracle** $\mathsf{QKeyG}(f)$**:** *Outputs* $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ *on a query* $f$*. We denote by* $Q_f$ *the queries of the form* $\mathsf{QKeyG}(\cdot)$*.*

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

- *If* $i \in \mathcal{CS}$ *(i.e., slot* $i$ *is corrupted): for any query* $\mathsf{QLeftRight}(i, x_i^0, x_i^1, \ell)$*,* $x_i^0 = x_i^1$*.*
- *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* $\{x_i \in \mathcal{X}_{\lambda,i}\}_{i \in \mathcal{CS}}$*, we define* $x_i^0 = x_i^1 = x_i$ *for any slot* $i \in \mathcal{CS}$ *and any slot queried to* $\mathsf{QEnc}(i, x_i, \ell)$*, and we require that for any query* $\mathsf{QKeyG}(f)$*:*

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

- *When* $\mathrm{yy} = \mathrm{pos}^+$*: If there exists a slot* $i \in [n]$ *and a* $\ell \in \mathsf{Labels}$*, such that* $|Q_{i,\ell}| > 0$*, then for any slot* $k \in [n] \setminus \mathcal{CS}, |Q_{k,\ell}| > 0$*. In other words, for any label, either the adversary makes no left-or-right encryption query or makes at least one left-or-right encryption query for each slot* $i \in [n] \setminus \mathcal{CS}$*.*
- *When* $\mathrm{yy} = \mathrm{any}$*: there is no restriction in the left-or-right queries of the adversary.*

*We define the advantage of an adversary* $\mathcal{A}$ *for* $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$*,* $\mathrm{yy} \in \{\mathrm{pos}^+, \mathrm{any}\}$ *in the following way:*

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

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

*In addition, we call a scheme* $q$-message bounded*, if* $\sum_{i \in [n]}(\sum_{\ell \in QL} |Q_{i,\ell}| + \sum_{\ell \in QL'} |Q'_{i,\ell}|) < q$ *and* $q$-message-and-key bounded*, if* $\sum_{i \in [n]}(\sum_{\ell \in QL} |Q_{i,\ell}| + \sum_{\ell \in QL'} |Q'_{i,\ell}|) < q$ *and* $|Q_f| < q$*, with* $q = \mathrm{poly}(\lambda)$*.*

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

$(\mathcal{CS}, \{Q_{i,\ell}\}_{i \in [n], \ell \in QL}, \{Q'_{i,\ell}\}_{i \in [n], \ell \in QL'}) \leftarrow \mathcal{A}(1^\lambda, n)$

$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$

$\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{QLeftRight}(i, x_i^{j,0}, x_i^{j,1}, \ell)$, for all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$, for all $i \in [n]$ and $\ell \in QL$.

$\mathsf{ct}'^j_{i,\ell} \leftarrow \mathsf{QEnc}(i, x_i^j, \ell)$, for all $x_i^j \in Q'_{i,\ell}$, for all $i \in [n]$ and $\ell \in QL'$.

$\alpha \leftarrow \mathcal{A}^{\mathsf{QKeyG}(\cdot)}(\{\mathsf{sk}_i\}_{i \in \mathcal{CS}}, \{\mathsf{ct}^j_{i,\ell}\}_{i \in [n], \ell \in QL, j \in [|Q_{i,\ell}|]}, \{\mathsf{ct}'^j_{i,\ell}\}_{i \in [n], \ell \in QL', j \in [|Q'_{i,\ell}|]})$

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

Fig. 2: Selective Security Games for MCFE

$\mathbf{ad\text{-}yy\text{-}IND}_{\beta}^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$

$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QCor}(\cdot),\mathsf{QKeyG}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot)}(1^\lambda)$

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

Fig. 3: Adaptive Security Games for MCFE

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.

Multi-input functional encryption (MIFE) and functional encryption (FE) are special cases of MCFE. MIFE is the same as MCFE without corruption, and FE is the special case of $n = 1$ (in which case, MIFE and MCFE coincide as there is no non-trivial corruption). In the case of single-input functional encryption, we only consider the two security definitions of sel-FULL and ad-FULL. For simplicity, in the notion of MCFE security, we denote by sel the case of static corruption, and selective left-or-right and encryption queries are required. By ad we denote the case in which all three, corruption, left-or-right and encryption queries, are adaptive.

### 2.3 Security Compiler

As already mentioned in previous works [ABG19, ABKW19, AGRW17, ACF+18, CDG+18b] there exist two different types of security, namely pos$^+$ and any security. In the case of pos$^+$, the adversary is forced to ask a left-or-right query for every slot $i \in [n]$. The any security definition does not enforce any requirements on the slots that are asked to the left-or-right oracle by the adversary. For the case of the inner-product and the separable function functionalities schemes do not usually directly fulfill the notion of any security, since it is possible to ask left-or-right oracle queries in a few slots, such that Condition (*) cannot be evaluated, but the adversary is able to use its queries and its (partial) functional keys to distinguish if the left or right challenge message has been encrypted. Since these types of attacks are not possible in the setting of pos$^+$ security, a common approach is to construct a MCFE schemes that is pos$^+$ secure and then a compiler [ABKW19, ABG19, CDG+18b] is applied to achieve the desired notion of any security. In this paper we follow the same approach.

In recent works [ABG19], an additional encryption oracle, besides the left-or-right oracle, has been considered. As already mentioned in [ABG19, Remark 2.3], the security definition without the encryption oracle $\mathsf{QEnc}$, as defined in [ABKW19, CDG+18a], is only equivalent to the security notion with the encryption oracle in the case of any security but not in the case of pos$^+$ security. If we want to simulate the encryption oracle in the case of pos$^+$ security, we would simulate it by asking the message the adversary queries in both positions to the left or right oracle, but since pos$^+$ enforces the reduction to ask a message in all the remaining positions it might not be possible to find such a message. Therefore the definition with the additional encryption oracle is slightly stronger.

Now, we recap the recent "pos$^+$" to "any" security compiler as introduced in [ABG19] w.r.t. a decentralize MCFE scheme that follows the definition in [ABG19] (in Section 7 we will provide a slightly different notion of decentralize MCFE scheme). We now provide a proof sketch that shows that the result also holds in the case of a selectively secure (key and) message bounded multi-client functional encryption scheme.

**Theorem 2.5.** *Let* $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGenShare}, \mathsf{KeyGenComb}, \mathsf{Enc}, \mathsf{Dec})$ *be an* xx-pos$^+$-IND*-secure (key and) message bounded DMCFE scheme for a family of functions* $\mathcal{F}$*. Let* $\mathsf{SE} = (\mathsf{Gen}^{\mathsf{SE}}, \mathsf{Enc}^{\mathsf{SE}}, \mathsf{Dec}^{\mathsf{SE}})$ *be an* IND-CPA *secure symmetric key encryption scheme and let* $\mathsf{PRF}$ *be a* IND *secure pseudorandom function. Then the DMCFE scheme* $\mathsf{DMCFE}' = (\mathsf{Setup}', \mathsf{KeyGenShare}', \mathsf{KeyGenComb}', \mathsf{Enc}', \mathsf{Dec}')$ *described in Fig. 4 is (key and) message bounded* xx-pos$^+$-IND *secure.*

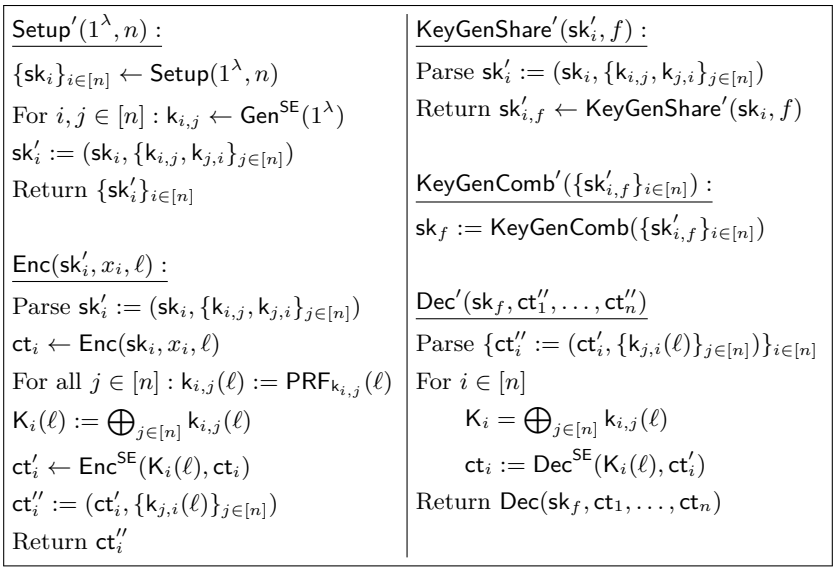


Fig. 4: Compiler from an xx-pos$^+$-IND-secure DMCFE scheme, DMCFE, into an xx-any-IND-secure DMCFE scheme, DMCFE$'$

*Proof (Sketch).* The proof uses the xx-pos$^+$-IND security of the scheme DMCFE for the case where all honest slots are queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$, for a single label $\ell^\star$, and the security of the PRF together with the IND-CPA security of SE for the case where all honest slots are queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$.

We define the game $\mathsf{G}^\star_\beta$ as the ad-pos$^+$-IND$^{\mathsf{DMCFE}'}_\beta(\lambda, n, \mathcal{A})$ game, except that the game guesses uniformly random an honest slot $i^\star \leftarrow \{0,\ldots,n\}$, where $i^\star = 0$ means that all honest slots are queried, that is not going to be queried to $\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\ell^\star)$. In the case that the guess $i^\star$ is unsuccessful, $\mathsf{G}^\star_\beta$ outputs 0. This guessing is not necessary in the case of selective security. In the case of selective security, we can just pick an honest slot, since the honest slots are directly disclosed by the adversary in the beginning of the game.

If $i^\star = 0$, we are in the case of pos$^+$ and therefore, we can directly reduce the security to the security of DMCFE.

To prove the security for all $i^\star \in [n]$, we use the fact that if there is a left-or-right oracle query $\mathsf{QLeftRight}(i, x_i^{j,0}, x_i^{j,1})$ with $x_i^{j,0} \neq x_i^{j,1}$, then the slot cannot be corrupted anymore after Condition (*) of Definition 7.1. Such a slot and the corresponding query is called explicitly honest.

We define hybrid games $\mathsf{G}^\star_{0.\rho}$ for all $\rho \in \{0,\ldots,n\}$ as $\mathsf{G}^\star_0$ except that every explicitly honest query $\mathsf{QLeftRight}(i, x_i^{j,0}, x_i^{j,1}, \ell^\star)$ is answered by $\mathsf{Enc}'(\mathsf{sk}'_i, x_i^{j,1}, \ell^\star)$ for $i \leq \rho$ and by $\mathsf{Enc}'(\mathsf{sk}_i, x_i^{j,0}, \ell^\star)$ for $i > \rho$. It follows that $\mathsf{G}^\star_{0,0} = \mathsf{G}^\star_0$ and $\mathsf{G}^\star_{0.n} = \mathsf{G}^\star_1$. We note that, again, in the case of selective security all the honest slots are known from the beginning and therefore we directly know how to answer which slots.

To go from hybrid $\mathsf{G}^\star_{0.\rho-1}$ to $\mathsf{G}^\star_{0.\rho}$, we distinguish between two different cases. First, slot $\rho$ is never queried on an explicitly honest slot, in this case the two games are the same by definition. Otherwise, we rely on the security of the PRF to make the key $\mathsf{k}_{\rho,i^\star}(\ell^\star)$ uniformly random. This switch is possible, since we know the slots $i^\star$ and $\rho$. If the guess $i^\star$ is correct, the key $\mathsf{k}_{\rho,i^\star}(\ell^\star)$ only appears in the output of $\mathsf{QLeftRight}(\rho,\cdot,\cdot,\ell^\star)$. This results in the fact that we have a uniformly random key $\mathsf{K}_\rho(\ell^\star)$, which allows us to rely on the IND-CPA security of the the symmetric encryption scheme, since $\mathsf{Gen}^{\mathsf{SE}}$ just generates a random element as the encryption key, and change the encryptions of $x_i^{j,0}$ in $\mathsf{G}^\star_{0.\rho-1}$ to encryptions of $x_i^{j,1}$ in $\mathsf{G}^\star_{0.\rho}$. Afterwards, we switch back the key $\mathsf{k}_{\rho,i^\star}$ from uniformly random to pseudorandom by relying on the security of the PRF an additional time.

Applying this reduction for all the $n$ different slots and all the queried labels yields the theorem. □

For more details on this proof, we refer to [ABG19].

### 2.4 Separable Functionalities

In this work, we focus on the class of additive separable functions. We recap the definition of a separable function and the corresponding functionality:

**Definition 2.6 (Separable Functions [MAS06]).** *A function $f : \mathcal{X}_{\lambda,1} \times \cdots \times \mathcal{X}_{\lambda,n} \to \mathcal{Y}_\lambda$, is called separable, if there exists a function $f^i : \mathcal{X}_{\lambda,i} \to \mathcal{Y}_\lambda$ for all $i \in [n]$, such that*

$$f(x_1, \ldots, x_n) = \sum_{i \in [n]} f^i(x_i), \text{ with } x_i \in \mathcal{X}_{\lambda,i} \text{ for all } i \in [n] \ .$$

*Separable Functionality.* We define the functionality class for separable functions as $\mathcal{F}_n^{\mathsf{sep}} := \{f(x_1, \ldots, x_n) = f^1(x_1) + \cdots + f^n(x_n), \text{ with } f^i : \mathcal{X}_{\lambda,i} \to \mathcal{Y}_\lambda\}$.

In this work, we consider the class of separable functions over the group $\mathbb{Z}_p$. Since the separability of a function $f$ is not necessarily unique, we require the adversary to submit its functional key generation query as a set of the separated functions $\{f^i\}_{i \in [n]}$.

### 2.5 Pseudorandom Functions (PRF)

The details on pseudorandom functions can be found in **??**.

## 3 Symmetric Encryption and One-Time Pad Extension

In this section, we recap some definitions regarding symmetric encryption. This consists of the security definition and the one-time pad. We start by formally defining symmetric encryption.

**Definition 3.1 (Symmetric Encryption).** *A symmetric encryption scheme (SE) for the key space $\mathcal{K}$ and the message space $\mathcal{M}$ is a couple of algorithms $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$:*

$\mathsf{Enc}(\mathsf{K}, m)$**:** *Takes as input the symmetric key $\mathsf{K}$, a message $m \in \mathcal{M}$ to encrypt, and outputs a ciphertext $\mathsf{ct}$.*
$\mathsf{Dec}(\mathsf{K}, \mathsf{ct})$**:** *Takes as input the symmetric key $\mathsf{K}$ and a ciphertext $\mathsf{ct}$ and outputs a message or $\perp$ if decryption fails.*

*A scheme $\mathsf{SE}$ is correct, if for all $\lambda \in \mathbb{N}$, $\mathsf{K} \leftarrow \mathcal{K}$, $m \in \mathcal{M}$, we have*

$$\Pr\left[\mathsf{Dec}(\mathsf{K}, \mathsf{Enc}(\mathsf{K}, m)) = m\right] = 1 \ .$$

Before formally introducing the one-time pad, we recap the security definition for a symmetric encryption scheme.

**Definition 3.2 (IND-CPA Security of SE).** *Let $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$ be an SE scheme, for the message space $\mathcal{M}$. We define the experiment $\text{IND-CPA}_\beta^{\mathsf{SE}}$ in Fig. 5, where the oracle is defined as:*

**Left-or-Right oracle** $\mathsf{QLeftRight}(m^{j,0}, m^{j,1})$**:** *Outputs $\mathsf{ct}^j = \mathsf{Enc}(\mathsf{K}, m^{j,\beta})$ on a query $(m^{j,0}, m^{j,1})$. We denote by $Q_{\mathsf{LeftRight}}$ the set containing the queries $(m^0, m^1)$.*

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

$$\mathsf{Adv}_{\mathsf{SE},\mathcal{A}}^{\text{IND-CPA}}(\lambda) = |\Pr[\text{IND-CPA}_0^{\mathsf{SE}}(\lambda, \mathcal{A}) = 1] - \Pr[\text{IND-CPA}_1^{\mathsf{SE}}(\lambda) = 1]| \ .$$

*A symmetric encryption scheme $\mathsf{SE}$ is called IND-CPA secure, if for any PPT adversary $\mathcal{A}$ it holds that $\mathsf{Adv}_{\mathsf{SE},\mathcal{A}}^{\text{IND-CPA}}(\lambda) \leq \text{negl}(\lambda)$.*

$\textbf{IND-CPA}^{\mathsf{SE}}_{\beta}(\lambda, \mathcal{A})$

$\mathsf{K} \leftarrow \mathcal{K}$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot)}(1^{\lambda})$

**Output:** $\alpha$

Fig. 5: IND-CPA Security Game for a symmetric encryption scheme

**One-Time Pad.** Now, we recap a specific symmetric encryption scheme, the one-time pad, and the definition of perfect security, a restricted version of the IND-CPA security defined above.

$\underline{\mathsf{Enc}(\mathsf{K}, m \in \mathcal{M}) :}$

$\mathsf{ct} := m + \mathsf{K}$

Return $\mathsf{ct}$

$\underline{\mathsf{Dec}(\mathsf{K}, \mathsf{ct}) :}$

$m := \mathsf{ct} - \mathsf{K}$

Return $m$

Fig. 6: The One-Time Pad

$\textbf{PERF-IND}^{\mathsf{SE}}_{\beta}(\lambda, \mathcal{A})$

$\mathsf{K} \leftarrow \mathcal{K}$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot)}(1^{\lambda})$

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

Fig. 7: Perfect Security Game for a symmetric encryption scheme

**Definition 3.3 (Perfect Security of SE).** *Let* $\mathsf{SE}$ *be an SE scheme, for the message space* $\mathcal{M}$*. We define the experiment* $\text{PERF-IND}^{\mathsf{SE}}_{\beta}$ *in Fig. 7, where the oracle is defined as:*

**Left-or-Right oracle** $\mathsf{QLeftRight}(m^0, m^1)$**:** *Outputs* $\mathsf{ct} = \mathsf{Enc}(\mathsf{K}, m^{\beta})$ *on a query* $(m^0, m^1)$*. We denote by* $Q_{\mathsf{LeftRight}}$ *the set containing the queries* $(m^0, m^1)$*.*

*and where* Condition (*) *holds if the left-or-right oracle* $\mathsf{QLeftRight}$ *has been only queried once.*

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

$$\mathsf{Adv}^{\text{PERF-IND}}_{\mathsf{SE},\mathcal{A}}(\lambda) = |\Pr[\text{PERF-IND}^{\mathsf{SE}}_{0}(\lambda, \mathcal{A}) = 1] - \Pr[\text{PERF-IND}^{\mathsf{SE}}_{1}(\lambda) = 1]| \ .$$

*A symmetric encryption scheme* $\mathsf{SE}$ *is called perfectly secure, if for any adversary* $\mathcal{A}$ *it holds that* $\mathsf{Adv}^{\text{PERF-IND}}_{\mathsf{SE},\mathcal{A}}(\lambda) = 0$*.*

It has been proven in [Sha01] that the one-time pad is perfectly secure under the XOR operation. An adaption of this proof to finite group is straightforward and can be found for example in [Wic15].

**Theorem 3.4 (One-Time Pad).** *The scheme* $\mathsf{SE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Fig. 6 is perfectly secure. Namely, for any* $\mathcal{A}$ *it holds that* $\mathsf{Adv}^{\text{PERF-IND}}_{\mathsf{SE},\mathcal{A}} = 0$*.*

After introducing the one time pad and recapping that it fulfills perfect security, we introduce a new notion called conditional perfect security.

**Definition 3.5 (Conditional Perfect Security of SE).** *Let* $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$ *be an SE scheme, for the message space* $\mathcal{M}$*. We define the experiment* $\text{CON-PERF-IND}^{\mathsf{SE}}_{\beta}$ *in Fig. 8, where the oracle is defined as:*

**Left-or-Right oracle** $\mathsf{QLeftRight}(m^{j,0}, m^{j,1})$**:** *Outputs* $\mathsf{ct}^j = \mathsf{Enc}(\mathsf{K}, m^{j,\beta})$ *on a query* $(m^{j,0}, m^{j,1})$*. We denote by* $Q_{\mathsf{LeftRight}}$ *the set containing the queries* $(m^0, m^1)$*.*

$\mathbf{CON\text{-}PERF\text{-}IND}^{\mathsf{SE}}_{\beta}(\lambda, \mathcal{A})$

$\mathsf{K} \leftarrow \mathcal{K}$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot)}(1^{\lambda})$

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

Fig. 8: Conditional Perfect Security Game

*and where* Condition (*) *holds if for all couple of queries* $(m^{j,0}, m^{j,1}), (m^{k,0}, m^{k,1}) \in Q_{\mathsf{LeftRight}}$ *we have that*

$$m^{j,0} - m^{j,1} = m^{k,0} - m^{k,1} \ .$$

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

$$\mathsf{Adv}^{\text{CON-PERF-IND}}_{\mathsf{SE},\mathcal{A}}(\lambda) = |\Pr[\text{CON-PERF-IND}^{\mathsf{SE}}_{0}(\lambda, \mathcal{A}) = 1] - \Pr[\text{CON-PERF-IND}^{\mathsf{SE}}_{1}(\lambda) = 1]| \ .$$

*A symmetric encryption scheme* $\mathsf{SE}$ *is called conditional perfectly secure, if for any adversary* $\mathcal{A}$ *it holds that* $\mathsf{Adv}^{\text{CON-PERF-IND}}_{\mathsf{SE},\mathcal{A}}(\lambda) = 0$.

Now, we show that the one-time pad also fulfills conditional perfect security.

**Lemma 3.6.** *Let* $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$ *be the perfectly secure one-time pad, then* $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$ *is also conditional perfectly secure. Namely, for any adversay* $\mathcal{A}$*, there exists an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\text{CON-PERF-IND}}_{\mathsf{SE},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\text{PERF-IND}}_{\mathsf{SE},\mathcal{B}}(\lambda)$$

*Proof.* We build an adversary $\mathcal{B}$ that simulates the CON-PERF-IND$^{\mathsf{SE}}_{\beta}$ game to $\mathcal{A}$, when interacting with the PERF-IND$^{\mathsf{SE}}_{\beta}$ experiment.

When $\mathcal{A}$ submits its first encryption query $(m^{1,0}, m^{1,1})$ to $\mathcal{B}$, $\mathcal{B}$ forwards it to its experiment, receives $\mathsf{ct}^1$ as an answer and sends $\mathsf{ct}^1$ to $\mathcal{A}$. For every further query $(m^{j,0}, m^{j,1})$ that $\mathcal{A}$ asks, $\mathcal{B}$ computes $c_j := m^{j,1} - m^{1,1}$ and sends $\mathsf{ct}^j := \mathsf{ct}^1 + c_j$ as a reply to $\mathcal{A}$.

To complete the proof, we show that $\mathsf{ct}^j$ corresponds to an encryption of $m^{j,\beta}$. This results in a perfect simulation and therefore the theorem follows.

In the first step, $\mathcal{B}$ receives $\mathsf{ct}^1 = m^{1,\beta} + \mathsf{K}$ from its experiment. For all the following queries made by $\mathcal{A}$, it holds that $m^{j,1} - m^{j,0} = m^{1,1} - m^{1,0}$. Therefore we can write the two different queries $m^{j,0}$ and $m^{j,1}$ as follows:

$$m^{j,0} = m^{1,0} + m^{j,1} - m^{1,1}$$
$$m^{j,1} = m^{1,1} + m^{j,0} - m^{1,0}$$

For the message $m^{j,1}$, we can also write $m^{j,1} = m^{1,1} + m^{j,1} - m^{1,1}$ through zero addition. By setting $c_j = m^{j,1} - m^{1,1}$ and calculation $\mathsf{ct}^1 + c_j$, we get an encryption of $\mathsf{ct}^j$ and therefore the theorem follows. □

## 4 Multi-Client Functional Encryption for Separable Functions

In this section, we present our compiler, described in Fig. 9, that turns a single-input functional encryption scheme for class $\mathcal{F}^{\mathsf{sep}}_1$ into a multi-client functional encryption scheme $\mathsf{MCFE}$ with labels $\mathsf{Labels}$ for the class of separable functions $\mathcal{F}^{\mathsf{sep}}_n$, by relying on a $\mathsf{PRF}$ instantiated with the keyspace $\mathcal{K} := \{0,1\}^{\lambda}$, the domain $\mathcal{V} := \mathsf{Labels}$ and the range $\mathcal{W} := \mathcal{Y}_{\lambda}$, where $\mathcal{Y}_{\lambda}$ is the range of the functions $f \in \mathcal{F}^{\mathsf{sep}}_n$.

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

Generate $n$ different single input functional encryption instances

$\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$, for all $i \in [n]$.

And PRF keys for $i \in [n], j > i$:

$\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$

$\mathsf{msk} := (\{\mathsf{msk}_i\}_{i\in[n]}, \{\mathsf{K}_{i,j}\}_{i,j\in[n],i\neq j})$

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

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

$\underline{\mathsf{KeyGen}^{\mathsf{mc}}(\mathsf{msk}, \{f^i\}_{i\in[n]}):}$

Parse $\mathsf{msk} := (\{\mathsf{msk}_i\}_{i\in[n]}, \{\mathsf{K}_{i,j}\}_{i,j\in[n],i\neq j})$

Sample $n-1$ random values $r_i \leftarrow \mathcal{Y}_\lambda$ and set $r_n := -\sum_{i\in[n-1]} r_i$.

Generate a functional key for $f^i_{r_i}$ (defined in Fig. 10a Fig. 10b) for all $i \in [n]$:

$\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$, for all $i \in [n]$, where the size of $\mathsf{sk}_{f^i_{r_i}}$ is bounded by $q$.

Return $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$

$\underline{\mathsf{Enc}^{\mathsf{mc}}(\mathsf{sk}_i, x_i, \ell):}$

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

$t_{i,\ell} := \sum_{j\neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$

$\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i, \bot, t_{i,\ell}, \ell))$, where the size of $\mathsf{ct}_{i,\ell}$ is bounded by $q$

Return $\mathsf{ct}_{i,\ell}$

$\underline{\mathsf{Dec}^{\mathsf{mc}}(\mathsf{sk}_f, \{\mathsf{ct}_{i,\ell}\}_{i\in[n]}):}$

Compute $\mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{f^i_{r_i}}, \mathsf{ct}_{i,\ell}) = f^i(x_i) + t_{i,\ell} + r_i$

Return $f(x_1, \ldots, x_n) = \sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_i$

Fig. 9: The generic construction of $q$-message bounded sel-pos$^+$-IND-secure MCFE and $q$-message-and-key bounded ad-pos$^+$-IND-secure MCFE multi-client functional encryption from single-input functional encryption.

$\underline{f^i_{r_i}(x, t_{i,\ell}, \ell):}$

Output: $f^i(x) + t_{i,\ell} + r_i$

(a) Selective Security

$\underline{f^i_{r_i}(x, \bot, t_{i,\ell}, \ell):}$

Output: $f^i(x) + t_{i,\ell} + r_i$

(b) Adaptive Security

Fig. 10: Description of the function that is used for the key generation under the different security definitions

The construction works in the following way: In the setup procedure, $n$ different instances of the single-input functional encryption scheme $\{\mathsf{msk}_i\}_{i\in[n]}$ and shared keys $\mathsf{K}_{i,j}$ (shared between slot $i$ and $j$) for all $i, j \in [n], i \neq j$, with $\mathsf{K}_{i,j} = \mathsf{K}_{j,i}$ are generated. These keys are used as PRF keys in the encryption procedure. The setup procedure outputs a master secret key $\mathsf{msk}$ containing all the different master secret keys from the different single-input instances and a secret key $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ for every slot $i \in [n]$. We continue by describing the behavior of the remaining algorithms.

To encrypt a message for position $i$, the encryption algorithm takes as input the secret key $\mathsf{sk}_i$, a message $x_i$ and a label $\ell$. In the first step, a padding $t_{i,\ell}$ will be generated using the PRF keys $\{\mathsf{K}_{i,j}\}_{j\in[n]}$ contained in the secret key $\mathsf{sk}_i$. This padding is different for every label $\ell$ and ensures that ciphertexts created under different labels cannot be combined. In more detail, for every padding. it holds that $\sum_{i\in[n]} t_{i,\ell} = 0$ for each label, but if paddings for different labels are combined they do not add up to 0. To generate the ciphertext $\mathsf{ct}_{i,\ell}$, the message $x_i$ concatenated with the padding $t_{i,\ell}$ and the label $\ell$ is encrypted using $\mathsf{msk}_i$.

The key generation procedure, takes as input the master secret key $\mathsf{msk}$ and a function $f \in \mathcal{F}_n^{\mathsf{sep}}$ separated into the functions $f^1, \ldots, f^n$ with $f^i \in \mathcal{F}_1^{\mathsf{sep}}$ for all $i \in [n]$. In the first step of the key generation, $n$ different random values $r_i$ are sampled in such a way that $\sum_{i\in[n]} r_i = 0$, these values are used to ensure that different functional keys cannot be combined. In the next step, a functional key $\mathsf{sk}_{f^i_{r_i}}$ for the function $f^i_{r_i}$ is generated for every single-input instance $i \in [n]$. The function $f^i_{r_i}$ takes as input the message $x_i$ and the padding $t_{i,\ell}$ and outputs the addition of these values together with the hardcoded value $r_i$, i.e. $f^i_{r_i}(x_i, t_{i,\ell}, \ell) = f^i(x_i) + t_{i,\ell} + r_i$. The functional key $\mathsf{sk}_f$ is defined as the set of all the functional keys generated by the single-input instances $\{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$.

To decrypt a set of ciphertexts $\{\mathsf{ct}_{i,\ell}\}_{i\in[n]}$ using a decryption key $\mathsf{sk}_f = \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$, the decryptions of all the instances are generated and the final output is computed by adding up all of the decryptions. In more detail, $\mathsf{Dec}(\mathsf{sk}_{f^i_{r_i}}, \mathsf{ct}_{i,\ell}) = f^i(x_i) + t_{i,\ell} + r_i$ is computed for all $i \in [n]$ and the final output $f(x_1, \ldots, x_n)$ is equal to $\sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_i$.

The output of the decryption of a single-input instance, i.e. $f^i(x_i) + t_{i,\ell} + r_i$ ensures that it is not possible to combine ciphertexts encrypted under different labels or functional keys generated in different key generation procedures. If one of the ciphertexts in the decryption procedure is generated under a different label such that the sum of all the paddings is unequal to 0 or a different partial functional key has been used such that the sum of the values added by the functional key is unequal to 0, the decryption procedure will not output the correct $f(x_1, \ldots, x_n)$.

**Correctness.** The correctness of the multi-client scheme follows from the correctness of the single input scheme, the fact that $\sum_{i\in[n]} t_{i,\ell} = 0$ and $\sum_{i\in[n]} r_i = 0$. Let us consider in more detail the decryption of the correctly generated ciphertexts $\mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell}$ under a correctly generated functional key $\mathsf{sk}_f = \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$. Due to the correctness of the single-input scheme it holds that $f^i(x_i) + t_{i,\ell} + r_i = \mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{f^i_{r_i}}, \mathsf{ct}_{i,\ell})$ and together with the properties of the $t_{i,\ell}$ values and the $r_i$ values it follows that $\sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_i = \sum_{i\in[n]} f^i(x_i)$. Together with the separability property of the function $\sum_{i\in[n]} f^i(x_i) = f(x_1, \ldots, x_n)$ correctness follows.

## 5 Selective Security

To prove the selective security of the proposed construction, we proceed via a hybrid argument. In the first hybrid, we replace the PRF's with random functions between a selected honest party $i^*$ and all the remaining honest parties $i \in \mathcal{HS} \setminus i^*$ such that the padding values $t_{i,\ell}$ are randomly generated. Our goal is to encode all the function evaluations of the left submitted challenges, i.e. $f^i(x_i^0) + t_{i,\ell} + r_i$ inside the functional keys and switch from encryptions of $(x_i^0, t_{i,\ell}, \ell)$ to encryptions of $(x_i^1, 0^\lambda, \ell)$. Since, after this step, all the random values are part of the functional key, we can rely on an information theoretic argument and change the values encoded in the functional key from $f^i(x_i^0) + t_{i,\ell} + r_i$ to $f^i(x_i^1) + t_{i,\ell} + r_i$. In the next hybrid, we generate the functional key in the same way as before and change from encryptions of $(x_i^1, 0^\lambda, \ell)$ to encryptions of $(x_i^1, t_{i,\ell}, \ell)$. In the last hybrid, we replace the random functions again with pseudorandom functions and therefore security follows. We present the formal security proof:

**Theorem 5.1 ($q$-message sel-pos$^+$-IND-security of MCFE).** *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a $q$-message bounded* sel-FULL*-secure single-input functional encryption scheme for the functionality class $\mathcal{F}_1^{\mathsf{sep}}$, and* PRF *an* IND *secure pseudorandom function, then the MCFE scheme* $\mathsf{MCFE} = (\mathsf{Setup}^{\mathsf{mc}}, \mathsf{KeyGen}^{\mathsf{mc}}, \mathsf{Enc}^{\mathsf{mc}}, \mathsf{Dec}^{\mathsf{mc}})$ *described in Fig. 9 is a $q$-message bounded* sel-pos$^+$-IND*-secure for the functionality*

*class $\mathcal{F}_n^{\mathsf{sep}}$. Namely, for any PPT adversary $\mathcal{A}$, there exists PPT adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that:*

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

*Proof.* The arguments used for the generation of the values $t_{i,\ell}$ are based on the proof in [ABG19] and we recap those parts here adapted to our construction. For the case with only one honest (non-corrupted) position, we can rely directly on the sel-FULL security of the underlying single-input functional encryption scheme $\mathsf{FE}$. Namely, we build a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\text{sel-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda,n) \leq \mathsf{Adv}^{\text{sel-pos}^+\text{-FULL}}_{\mathsf{FE},\mathcal{B}}(\lambda)$. After $\mathcal{B}$ has received $\{Q_{i,\ell}\}_{i\in[n],\ell\in QL}$, $\{Q'_{i,\ell}\}_{i\in[n],\ell\in QL'}$ and $\mathcal{CS}$ from $\mathcal{A}$, it generates $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n]\setminus i^*$, where $i^*$ denotes the honest slot, and samples $\mathsf{K}_{i,j}$ for all $i,j \in [n]$. Finally $\mathcal{B}$ sets $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ and sends $\{\mathsf{sk}_i\}_{i\in[n]\setminus\{i^*\}}$ to $\mathcal{A}$. It must hold for the queries $\{Q_{i,\ell}\}_{i\in[n],\ell\in QL}$, i.e. $\{(i,x_i^{j,0},x_i^{j,1},\ell)\}_{i\in[n],\ell\in QL,j\in[|Q_{i,\ell}|]}$, of $\mathcal{A}$ that $x_i^{j,0} = x_i^{j,1}$ for all $i \in [n]\setminus\{i^*\}$ and $j \in [|Q_{i,\ell}|]$. This results in the fact that $f^i_{r_i}(x_i^{j,0}) = f^i_{r_i}(x_i^{j,1})$ in every slot $i \in [n]\setminus\{i^*\}$ and for all queries $j \in [|Q_{i,\ell}|]$, which implies that $f^{i^*}_{r_{i^*}}(x_{i^*}^{j,0}) = f^{i^*}_{r_{i^*}}(x_{i^*}^{j,1})$. The left-or-right queries $\{Q_{i,\ell}\}_{i\in[n]\setminus i^*,\ell\in QL}$ can directly be answered by $\mathcal{B}$, it submits $\{((x_{i^*}^{j,0},t_{i^*,\ell},\ell),(x_{i^*}^{j,1},t_{i^*,\ell},\ell))\}_{\ell\in QL,j\in[|Q_{i,\ell}|]}$, with $t_{i^*,\ell} := \mathsf{Gen}(\mathsf{sk}_{i^*},i^*,\ell)$ for all $\ell \in QL$ computed by $\mathcal{B}$, as its own left-or-right queries to the experiment. It receives $\{\mathsf{ct}^j_{k,\ell}\}_{\ell\in QL,j\in[|Q_{i,\ell}|]}$ as an answer and sends $\{\mathsf{ct}^j_{i,\ell}\}_{i\in[n],\ell\in QL,j\in[|Q_{i,\ell}|]}$ as a reply to $\mathcal{A}$. For the submitted queries $\{Q'_{i,\ell}\}_{i\in[n],\ell\in QL'}$ $(i,x_i^j,\ell)$ to the encryption oracle $\mathsf{QEnc}$, we distinguish between two different cases. In the case that $\mathcal{A}$ asks for an encryption for all positions $i \neq k$, $\mathcal{B}$ computes $t_{i,\ell} := \mathsf{Gen}(\mathsf{sk}_i,i,\ell)$ for all $\ell \in QL'$ and $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^j,t_{i,\ell},\ell))$ for all $j \in [|'Q_{i,\ell}|]$ and $\ell \in QL'$. If $\mathcal{A}$ queries $\mathsf{QEnc}$ for the position $k$, i.e. it queries $(k,x^j,\ell)$ ,$\mathcal{B}$ computes $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$ for all $\ell \in QL'$, queries its own left-or-right encryption oracle on $((k,x^j,\ell),(k,x^j,\ell))$ for all $j \in [|Q'_{i,\ell}|]$ and $\ell \in QL'$. Finally, $\mathcal{B}$ sends the answer $\{\mathsf{ct}^j_{i,\ell}\}_{i\in[n],\ell\in QL',j\in[|Q'_{i,\ell}|]}$ to $\mathcal{A}$. Whenever $\mathcal{A}$ asks a key generation query $\mathsf{QKeyG}(\{f^i\}_{i\in[n]})$, $\mathcal{B}$ samples $r_i \leftarrow \mathcal{Y}_\lambda$ for all $i \in [n-1]$, sets $r_n := -\sum_{i\in[n-1]} r_i$ and generates $\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, f^i_{r_i})$ for all $i \in [n]\setminus\{i^*\}$. For the functional key $\mathsf{sk}_{f^{i^*}_{r_{i^*}}}$, $\mathcal{B}$ queries its own key generation oracle on $(f^{i^*}_{r_{i^*}}, f^{i^*}_{r_{i^*}})$. Finally it sends $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$ as a reply to $\mathcal{A}$ and we receive $\mathsf{Adv}^{\text{sel-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda,n) \leq \mathsf{Adv}^{\text{sel-FULL}}_{\mathsf{FE},\mathcal{B}}(\lambda)$.

For the cases with more than one honest position, we use a hybrid argument with the games defined in Fig. 11. Note that $\mathsf{G}_0$ corresponds to the game sel-pos$^+$-IND$^{\mathsf{MCFE}}_0(\lambda,n,\mathcal{A})$, and $\mathsf{G}_5$ corresponds to the game sel-pos$^+$-IND$^{\mathsf{MCFE}}_1(\lambda,n,\mathcal{A})$. This results in:

$$\mathsf{Adv}^{\text{sel-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda,n) = |\mathsf{Win}^{\mathsf{G}_0}_{\mathcal{A}}(\lambda,n) - \mathsf{Win}^{\mathsf{G}_5}_{\mathcal{A}}(\lambda,n)| \ .$$

We describe the different intermediate games in more detail:

**Game $\mathsf{G}_1$:** We replace the PRF evaluation for the computation of the masking values $t_{i,\ell}$ for the left-or-right oracle $\mathsf{QLeftRight}$ and the encryption oracle $\mathsf{QEnc}$ in the non-corrupted positions $i \in [n]\setminus\mathcal{CS}$ with random function evaluations. In more detail, we switch from the PRF generated values $\mathsf{PRF}_{\mathsf{K}_{i_1,i_s}}$ to $\mathsf{RF}_s(\ell)$, for all $s \in \{2,\ldots,h\}$, where the set of honest users is denoted as $\mathcal{HS} := \{i_1,\ldots,i_h\}$, $h \leq n$ denotes the number of honest users, and $\mathsf{RF}$ denotes a random function (see Fig. 12 for more details). The transition from $\mathsf{G}_0$ to $\mathsf{G}_1$ is justified by the security of the $\mathsf{PRF}$. Namely, in Lemma 5.2, we exhibit a PPT adversary $\mathcal{B}_0$ such that:

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

**Game $\mathsf{G}_2$:** We replace the encryptions of $(x_i^{j,0},t_{i,\ell},\ell)$ with the encryptions of $(x_i^{j,1},0^\lambda,\ell)$ for all $(x_i^{j,0},x_i^{j,1}) \in Q_{i,\ell}$, all $\ell \in QL$ and all $i \in [n]$ in the left-or-right oracle and we replace the encryptions of $(x_i^j,t_{i,\ell},\ell)$ with the encryptions of $(x_i^j,0^\lambda,\ell)$ for all $x_i^j \in Q'_{i,\ell}$, all $\ell \in QL'$ and all $i \in [n]$ in the encryption oracle. The values $t_{i,\ell}$ in the left-or-right queries and the encryption queries are replaced with $0^\lambda$ to make the ciphertexts independent from the masking values $t_{i,\ell}$. We also replace the functional key $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$ (see Fig. 10a for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{Q_i,Y_i}}\}_{i\in[n]}$ (see Fig. 13 for the function description).

The hardcoded values $y_{i,\ell}^{j,f^i} \in Y_i$ are generated using the random value $r_i$, the queries $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$. The same holds for the hardcoded values $y_{i,\ell}'^{j,f^i} \in Y_i$. They are generated using the random value $r_i$, the queries $x_i^j \in Q'_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $y_{i,\ell}'^{j,f^i} := f^i(x_i^j) + t_{i,\ell} + r_i$. The transition from $\mathsf{G}_1$ to $\mathsf{G}_2$ is achieved using a hybrid argument with sequence $\mathsf{G}_{1.k}$, for $k \in [n]$. As already described in Fig. 11, it holds that $\mathsf{G}_1 = \mathsf{G}_{1.0}$ and $\mathsf{G}_2 = \mathsf{G}_{1.n}$. This results in

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n)| \leq \sum_{k=1}^{n} |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}}(\lambda, n)| \ ,$$

for any PPT adversary $\mathcal{A}$. The transition from $\mathsf{G}_{1.k-1}$ to $\mathsf{G}_{1.k}$ is justified by the full security of $\mathsf{FE}$. Namely, in Lemma 5.4, we exhibit a PPT adversary $\mathcal{B}_k$ for all $k \in [n]$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}}(\lambda, n)| \leq \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_k}^{\text{sel-FULL}}(\lambda) \ .$$

combining both of the statements and noticing that a PPT adversary $\mathcal{B}_1$ can be obtained by picking $i \in [n]$ and running $\mathcal{B}_i$, we can justify the transition from $\mathsf{G}_1$ to $\mathsf{G}_2$. Namely, in Lemma 5.3, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n)| \leq n \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_1}^{\text{sel-FULL}}(\lambda) \ .$$

**Game $\mathsf{G}_3$:** We change the generation of all the values $y_{i,\ell}^{j,f^i} \in Y_i$, which are computed using the random value $r_i$, the queries $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and the masking values $t_{i,\ell}$. We change the generation from $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$ to $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$. The transition from $\mathsf{G}_2$ to $\mathsf{G}_3$ is justified by an information theoretic argument and happens for all $i \in [n]$. In more detail, we prove the transition by relying on the conditioned perfect security of several instances of the one-time pad as shown in Lemma 3.6. Namely, in Lemma 5.5, we show that

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3}(\lambda, n)| = 0 \ ,$$

for all adversaries $\mathcal{A}$.

**Game $\mathsf{G}_4$:** We replace the encryptions of $(x_i^{j,1}, 0^\lambda, \ell)$ with the encryptions of $(x_i^{j,1}, t_{i,\ell}, \ell)$ for all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$, all $\ell \in QL$ and all $i \in [n]$ in the left-or-right oracle and we replace the encryptions of $(x_i^j, 0^\lambda, \ell)$ with the encryptions of $(x_i^j, t_{i,\ell}, \ell)$ for all $x_i^j \in Q'_{i,\ell}$, all $\ell \in QL'$ and all $i \in [n]$ in the encryption oracle. The masking values $t_{i,\ell}$ are inserted back into the ciphertext and replace the $0^\lambda$ values. We also replace the functional key $\mathsf{sk}_f := \{\mathsf{sk}_{f_{\mathcal{Q}_i,Y_i}^i}\}_{i\in[n]}$ (see Fig. 10a for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f_{r_i}^i}\}_{i\in[n]}$ (see Fig. 13 for the function description). The transition from $\mathsf{G}_3$ to $\mathsf{G}_4$ is almost symmetric to the transition from $\mathsf{G}_1$ to $\mathsf{G}_2$, justified by the full security of $\mathsf{FE}$ applied on every slot $i \in [n]$. Namely, it can be proven as in Lemma 5.5 that there exists a PPT adversary $\mathcal{B}_2$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4}(\lambda, n)| \leq n \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_2}^{\text{sel-FULL}}(\lambda) \ .$$

We defer to the proof of Lemma 5.5 for further details.

**Game $\mathsf{G}_5$:** This game is identical to sel-pos$^+$-IND$_1^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$. The transition from $\mathsf{G}_4$ to $\mathsf{G}_5$ is almost 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 5.2 that there exists a PPT adversary $\mathcal{B}_3$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_5}(\lambda, n)| \leq (h-1) \cdot \mathsf{Adv}_{\mathsf{PRF},\mathcal{B}_3}^{\text{IND}}(\lambda) \ .$$

We defer to the proof of Lemma 5.2 for further details.

Putting everything together, we obtain the theorem. □

| Game | $\mathsf{ct}^j_{i,\ell}$ | $\mathsf{ct}'^j_{i,\ell}$ | $\mathsf{sk}_f$ | justification/ remark |
|---|---|---|---|---|
| $\mathsf{G}_0$ | $t_{i,\ell} := \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,0}, t_{i,\ell}, \ell))$ | $t_{i,\ell} := \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, t_{i,\ell}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br><br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ | |
| $\mathsf{G}_1$ | $t_{i,\ell} := \boxed{\mathsf{Gen}'(\mathsf{sk}_i, i, \ell)}$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,0}, t_{i,\ell}, \ell))$ | $t_{i,\ell} := \boxed{\mathsf{Gen}'(\mathsf{sk}_i, i, \ell)}$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, t_{i,\ell}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br><br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ | PRF |
| $\mathsf{G}_{1.k}$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (\boxed{x_i^{j,1}, 0^\lambda}, \ell))$,<br>for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,0}, t_{i,\ell}, \ell))$,<br>for $i > k$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{0^\lambda}, \ell))$,<br>for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, t_{i,\ell}, \ell))$,<br>for $i > k$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>If $i \le k$<br>For all $\ell \in QL, (x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y^{j,f^i}_{i,\ell} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}$,<br>$\{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q'_{i,\ell}\rvert]}\}$<br>If $i \le k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$<br>If $i > k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ | Full Security of FE |
| $\mathsf{G}_2$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (\boxed{x_i^{j,1}, 0^\lambda}, \ell))$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br><br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{0^\lambda}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>For all $\ell \in QL, (x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y^{j,f^i}_{i,\ell} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}$,<br>$\{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q'_{i,\ell}\rvert]}\}$<br><br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | $\mathsf{G}_2 = \mathsf{G}_{1.n}$ |

Fig. 11a: Description of the games $\mathsf{G}_0$ to $\mathsf{G}_2$ for the proof of selective security.

| Game | $\mathsf{ct}^{j}_{i,\ell}$ | $\mathsf{ct}'^{j}_{i,\ell}$ | $\mathsf{sk}_f$ | justification/ remark |
|---|---|---|---|---|
| $\mathsf{G}_3$ | $\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, 0^\lambda, \ell))$ | $\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, 0^\lambda, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>For all $\ell \in QL, (x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$\boxed{y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i}$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}$,<br>$\{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q'_{i,\ell}\rvert]}\}$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | inf. theoretic |
| $\mathsf{G}_{3.k}$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, \boxed{t_{i,\ell}}, \ell))$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, 0^\lambda, \ell))$, for $i > k$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{t_{i,\ell}}, \ell))$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, 0^\lambda, \ell))$, for $i > k$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>If $i > k$<br>For all $\ell \in QL, (x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$y'^{j,f^i}_{i,\ell} := f^i(x_i^{j}) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}$,<br>$\{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q'_{i,\ell}\rvert]}\}$<br>If $i \le k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$<br>If $i > k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | Full Security of FE |
| $\mathsf{G}_4$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, \boxed{t_{i,\ell}}, \ell))$ | $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{t_{i,\ell}}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\boxed{\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})}$ | $\mathsf{G}_4 = \mathsf{G}_{3.n}$ |
| $\mathsf{G}_5$ | $t_{i,\ell} := \boxed{\mathsf{Gen}(\mathsf{sk}_i, i, \ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, t_{i,\ell}, \ell))$ | $t_{i,\ell} := \boxed{\mathsf{Gen}(\mathsf{sk}_i, i, \ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, t_{i,\ell}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ | PRF |

Fig. 11b: Description of the games $\mathsf{G}_3$ to $\mathsf{G}_5$ for the proof of selective security.

| $\mathsf{Gen}(\mathsf{sk}_i, i, \ell)$ | $\mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ |
|---|---|
| Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ | Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ |
| $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$ | $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$ |
| Return $t_{i,\ell}$ | If $i \in \mathcal{HS} := \{i_1, \ldots, i_h\}$, then: |
| | • If $i = i_1$, $t_{i,\ell} := \sum_{j\in\mathcal{CS}}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) + \sum_{s=2}^{h}\mathsf{RF}_s(\ell)$. |
| | • If $i = i_s$, for $s \in \{2, \ldots, h\}$, $t_{i,\ell} := \sum_{j\in[n]\setminus\{i_1,i_s\}}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) - \mathsf{RF}_s(\ell)$. |
| | Return $t_{i,\ell}$ |

Fig. 12: Generation of the tags for the labels in the selective case

$f^i_{\mathcal{Q}_i,Y_i}(x, t_{i,\ell}, \ell)$ :

Parse $\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$ and

$Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{\ell\in QL, j\in[|Q_{i,\ell}|]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[|Q'_{i,\ell}|]}\}$

If $(\cdot, x) \in Q_{i,\ell}$

Output: $y^{j,f^i}_{i,\ell}$

If $x \in Q'_{i,\ell}$

Output: $y'^{j,f^i}_{i,\ell}$

Fig. 13: Description of the function that is used in the reduction for the selective security reduction.

**Lemma 5.2 (Transition from $\mathsf{G}_0$ to $\mathsf{G}_1$).** *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}'$ such that*

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

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

*Proof.* This proof works mainly as described in [ABG19].

For all the honest positions $i, j \in \mathcal{HS}$, we can replace the PRF $\mathsf{PRF}$ with a random function $\mathsf{RF}$. This is due to the fact that the keys $\mathsf{K}_{i,j}$ (with $i, j \in \mathcal{HS}$) are totally hidden from the adversary $\mathcal{A}$. We can show that it is sufficient for the transition of game $\mathsf{G}_0$ to $\mathsf{G}_1$ to rely on the security of the PRF in $h-1$ chosen slots. In more detail, we write the ordered set $\mathcal{HS} := \{i_1, \ldots, i_h\}$, with $i_1 < i_2 < \cdots < i_h$. For all keys of the form $\mathsf{K}_{i_1,j}$ with $j \in \mathcal{HS} \setminus \{i_1\}$ we rely on the security of the $\mathsf{PRF}$.

The adversary $\mathcal{B}'$ works as follows. After receiving $\mathcal{CS}$ and the challenge messages $\{Q_{i,\ell}\}_{i\in[n],\ell\in QL}$ and $\{Q'_{i,\ell}\}_{i\in[n],\ell\in QL'}$ from $\mathcal{A}$, it samples $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n]$ and sets $\mathsf{msk} := \{\mathsf{msk}_i\}_{i\in[n]}$. In the next step, $\mathcal{B}'$ samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$, for all $i \in [n] \setminus \{i_1\}$ and $j > i$. The secret keys for the corrupted positions $i \in \mathcal{CS}$ are defined as $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$. After $\mathcal{B}'$ has set the secret keys for the corrupted positions, it sends them to $\mathcal{A}$. $\mathcal{B}'$ answers the queries to $\mathsf{QLeftRight}(i, x^{j,0}_i, x^{j,1}_i, \ell)$ and $\mathsf{QEnc}(i, x^j_i, \ell)$ using the defined $\mathsf{sk}_i$ for $i \in \mathcal{CS}$ and uses $\mathsf{msk}$ to simulate the $\mathsf{QKeyG}$ oracle.

The generation of the masking values $t_{i,\ell}$ for the honest position, depends on the queried slot. For the first honest slot $i_1$, $\mathcal{B}'$ computes

$$t_{i,\ell} := \sum_{j \in \mathcal{CS}} (-1)^{j<i_1} \mathsf{PRF}_{K_{i_1,j}}(\ell) + \sum_{s=2}^{h} \mathsf{RF}_s(\ell) \ .$$

This masking value is used to answer the queries $\mathsf{QLeftRight}(i_1, x_{i_1}^{j,0}, x_{i_1}^{j,1}, \ell)$ and $\mathsf{QEnc}(i_1, x_{i_1}^{j}, \ell)$.

For all the other honest slots, $\mathcal{B}'$ computes

$$t_{i,\ell} := \sum_{j \in [n] \setminus \{i_s, i_1\}} (-1)^{j<i_s} \mathsf{PRF}_{K_{i_s,j}}(\ell) + \sum_{s=2}^{h} \mathsf{RF}_s(\ell) \ ,$$

and uses it to answer queries of the form $\mathsf{QLeftRight}(i_t, x_{i_t}^{j,0}, x_{i_t}^{j,1}, \ell)$ and $\mathsf{QEnc}(i_t, x_{i_t}^{j}, \ell)$.

In this experiment, the adversary $\mathcal{B}'$ is interacting with $h-1$ instances of the $\mathrm{IND}_\beta^{\mathsf{PRF}}(\lambda, \mathcal{B}')$ experiment. With many instances we mean that the adversary is querying many different experiments but in all of these experiments the reply is either the random function evaluation or the pseudorandom functions evaluation. It can be shown via a simple hybrid argument that the many instances experiment follows from a single instance experiment. □

**Lemma 5.3 (Transition from $\mathsf{G}_1$ to $\mathsf{G}_2$).** *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}''$ such that*

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n)| \leq n \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}''}^{\text{sel-FULL}}(\lambda) \ .$$

*Proof.* To prove that $\mathsf{G}_1$ is indistinguishable from $\mathsf{G}_2$ we need to apply a hybrid argument over the $n$ slots, using the full security of the single-input functional encryption scheme.

Using the definition of the games in Fig. 11 and the triangle inequality, we can see that

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2}(\lambda, n)| \leq \sum_{k=1}^{n} |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}}(\lambda, n)| \ ,$$

where $\mathsf{G}_1$ corresponds to game $\mathsf{G}_{1.0}$ and whereas $\mathsf{G}_2$ is identical to game $\mathsf{G}_{1.n}$.

Now, we can bound the difference between each consecutive pair of games for every $k \in [n]$.

**Lemma 5.4.** *For every $k \in [n]$, there exists a PPT adversary $\mathcal{B}_k$ against the* sel-FULL *property of the single-input scheme* FE *such that*

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}}(\lambda, n)| \leq \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_k}^{\text{sel-FULL}}(\lambda) \ .$$

*Proof.* We build an adversary $\mathcal{B}_k$ that simulates $\mathsf{G}_{1.k-1+\beta}$ to $\mathcal{A}$ when interacting with the underlying sel-FULL$_\beta^{\mathsf{FE}}$ experiment.

In the beginning of the reduction, $\mathcal{B}_k$ receives $\mathcal{CS}$, $\{Q_{i,\ell}\}_{i\in[n],\ell\in QL}$ and $\{Q'_{i,\ell}\}_{i\in[n],\ell\in QL'}$ from $\mathcal{A}$ and sets $\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$. If $k \in \mathcal{CS}$, the adversary $\mathcal{B}_k$ directly outputs $\alpha \leftarrow \{0,1\}$. This is due to the fact that the games $\mathsf{G}_{1.k-1}$ and $\mathsf{G}_{1.k}$ are identical in this case, which results in an advantage equal to 0 and Lemma 5.4 trivially holds. If $k \notin \mathcal{CS}$, $\mathcal{B}_k$ generates $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n] \setminus \{k\}$, samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$, with $i \in \mathcal{CS}$ and sets $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ for $i \in \mathcal{CS}$.

In the first step, to answer the left-or-right queries, $\mathcal{B}_k$ computes $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for all $i \geq k$ and all $i \in \mathcal{CS}$ for all $\ell \in QL$. To generate the final ciphertexts it does the following: For all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ for all $\ell \in QL$, $\mathcal{B}_k$ computes the ciphertexts $\mathsf{ct}_{i,\ell}^j \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, 0^\lambda, \ell))$ for all $i < k$ and $i \in \mathcal{HS}$ and $\mathsf{ct}_{i,\ell}^j \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,0}, t_{i,\ell}, \ell))$ for all $i > k$ or $i \in \mathcal{CS}$. To compute $\mathsf{ct}_{k,\ell}^j \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_k, x_k^{j,\beta}, t_{k,\ell}^\beta, \ell)$ (with $t_{i,\ell}^0 = t_{i,\ell}$ and $t_{i,\ell}^1 = 0^\lambda$), $\mathcal{B}_k$ submits the set $\{((x_k^{j,0}, t_{k,j}, \ell), (x_k^{j,1}, 0^\lambda, \ell))\}_{\ell\in QL, j\in[|Q_{i,\ell}|]}$ as its own left-or-right queries to the experiment. It receives $\{\mathsf{ct}_{k,\ell}^j\}_{\ell\in QL, j\in[|Q_{i,\ell}|]}$ as an answer to its queries.

The adversary $\mathcal{B}_k$ behaves similar to answer the encryption oracle queries. In more detail, for all $x_i^j \in Q'_{i,\ell}$ for all $\ell \in QL'$, $\mathcal{B}_k$ first computes $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for all $i \geq k$ and $i \in \mathcal{CS}$ and for all $\ell \in QL'$. If $i = k$, $\mathcal{B}_k$ submits the left-or-right query $((x_i^j, t_{i,\ell}, \ell), (x_i^j, 0^\lambda, \ell))$ for all $j \in [|Q_{i,\ell}|]$ and $\ell \in QL'$ to its own left-or-right oracle and receives $\mathsf{ct}'^j_{i,\ell}$ as an answer. For $i < k$ and $i \in \mathcal{HS}$, $\mathcal{B}_k$ uses its master secret key $\mathsf{msk}_i$ and computes $\mathsf{ct}'_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{msk}_i, (x_i^j, t_{i,\ell}, \ell))$ for all $j \in [|Q_{i,\ell}|]$ and $\ell \in QL'$ and for $i > k$ and $i \in \mathcal{CS}$, $\mathcal{B}_k$ uses its master secret key $\mathsf{msk}_i$ and computes $\mathsf{ct}'_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{msk}_i, (x_i^j, 0^\lambda, \ell))$ for all $j \in [|Q_{i,\ell}|]$ and $\ell \in QL'$.

As an answer to the queries asked by $\mathcal{A}$ in the beginning of the game, $\mathcal{B}_k$ sends $(\{\mathsf{sk}_i\}_{i\in\mathcal{CS}}, \{\mathsf{ct}^j_{i,\ell}\}_{i\in[n],\ell\in QL,j\in[|Q_{i,\ell}|]}$, $\{\mathsf{ct}'^j_{i,\ell}\}_{i\in[n],\ell\in QL',j\in[|Q'_{i,\ell}|]})$ to $\mathcal{A}$.

Whenever the adversary $\mathcal{A}$ asks a key generation query $\mathsf{QKeyG}(\{f^i\}_{i\in[n]})$, $\mathcal{B}_k$ samples $r_i \leftarrow \mathcal{Y}_\lambda$ for all $i \in [n]$ and sets $r_n := -\sum_{i\in[n-1]} r_i$. For all the left-or-right oracle queries $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ for all $\ell \in QL$ and $i \in [n]$, $\mathcal{B}_k$ generates $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for all $i \in [k] \setminus \mathcal{CS}$ and computes $y^{j,f^i}_{i,\ell} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$ for all $j \in [|Q_{i,\ell}|]$. For all the encryption queries $x_i^j \in Q'_{i,\ell}$ for all $\ell \in QL'$ and $i \in [n]$, $\mathcal{B}_k$ generates $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for all $i \in [k] \setminus \mathcal{CS}$ and computes $y'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$ for all $j \in [|Q'_{i,\ell}|]$. $\mathcal{B}_k$ sets $Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{\ell\in QL,j\in[|Q_{i,\ell}|]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL',j\in[|Q'_{i,\ell}|]}\}$ for all $i \in [n]$ and computes $\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i,Y_i})$ for the slots $i < k$ with $i \in \mathcal{HS}$ and $\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ for $i > k$ or $i \in \mathcal{CS}$. To generate $\mathsf{sk}_{f^k}$, $\mathcal{B}_k$ queries its own key generation oracle $\mathsf{QKeyG}$ on $(f^k_{r_k}, f^k_{\mathcal{Q}_k,Y_k})$, this query fulfills the functional restriction, i.e. it holds that $f^k_{r_k}(x_k^{j,0}) = f^k_{\mathcal{Q}_k,Y_k}(x_k^{j,0})$ for all $(x_k^{j,0}, \cdot) \in Q_{k,\ell}$ for all $\ell \in QL$. $\mathcal{B}_k$ receives $\mathsf{sk}_{f^k}$ as an answer, sets $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}}\}_{i\in[k-1]\setminus\mathcal{CS}} \cup \{\mathsf{sk}_{f^k}\} \cup \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in(\{k+1,\ldots,n\}\cup\mathcal{CS})}$ and sends $\mathsf{sk}_f$ to $\mathcal{A}$.

This covers the simulation of the game $\mathsf{G}_{1.k-1+\beta}$. Finally, $\mathcal{B}_k$ outputs the same bit $\beta'$ returned by $\mathcal{A}$. Thus, we obtain the lemma. □

The proof of the lemma follows by noticing that the adversary $\mathcal{B}''$ in the lemma statement can be obtained by picking $k \in [n]$ and running $\mathcal{B}_k$. □

**Lemma 5.5 (Transition from $\mathsf{G}_2$ to $\mathsf{G}_3$).** *For any adversary $\mathcal{A}$ it holds that*

$$|\mathsf{Win}^{\mathsf{G}_2}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_3}_{\mathcal{A}}(\lambda, n)| = 0 \ .$$

*Proof.* We build a PPT adversary $\mathcal{B}'''$ that simulates $\mathsf{G}_{2+\beta}$ to $\mathcal{A}$, when interacting with $(|\mathcal{HS}|-1)\cdot|QL|\cdot|Q_f|$ instances of the one-time pad in the CON-PERF-IND$_\beta$ experiment, as proven in Lemma 3.6. With many-instances we mean that encryption oracle of the one-time pad can be queried several times, but always the same position (left or right) is encrypted. In more detail, a new key $k^{f^i}_{i,\ell}$ is chosen for every position $i \in [n]$, for every label $\ell \in QL$ and for every function $\{f^i\}_{i\in[n]} \in Q_f$.

In the beginning of the reduction, $\mathcal{B}'''$ receives $\mathcal{CS}$, $\{Q_{i,\ell}\}_{i\in[n],\ell\in QL}$ and $\{Q'_{i,\ell}\}_{i\in[n],\ell\in QL'}$ from $\mathcal{A}$ and sets $\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$. Then $\mathcal{B}'''$ generates $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n]$, samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i < j \in [n]$, with $i \in \mathcal{CS}$ and sets $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ for $i \in \mathcal{CS}$.

To answer the left-or-right queries, $\mathcal{B}'''$ proceeds different corresponding to the queried position $i$. For all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ for all $\ell \in QL$, $\mathcal{B}'''$ computes the ciphertexts $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, 0^\lambda, \ell))$ if $i \in \mathcal{HS}$ and it computes the ciphertext $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,0}, t_{i,\ell}, \ell))$ with $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for $i \in \mathcal{CS}$.

To answer the encryption oracle queries $x_i^j \in Q'_{i,\ell}$ for all $\ell \in QL'$, $\mathcal{B}'$ first computes $t_{i,\ell} := \mathsf{Gen}'(\mathsf{sk}_i, i, \ell)$ for all $i \in \mathcal{CS}$ and for all $\ell \in QL'$. Then $\mathcal{B}_k$ uses its master secret key $\mathsf{msk}_i$ and computes $\mathsf{ct}'_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{msk}_i, (x_i^j, t_{i,\ell}, \ell))$ for $i \in \mathcal{CS}$ for all $j \in [|Q_{i,\ell}|]$ and $\ell \in QL'$ and $\mathsf{ct}'_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{msk}_i, (x_i^j, 0^\lambda, \ell))$ for $i \in \mathcal{HS}$ for all $j \in [|Q_{i,\ell}|]$ and $\ell \in QL'$

As an answer to the queries asked by $\mathcal{A}$ in the beginning of the game, $\mathcal{B}'''$ sends $(\{\mathsf{sk}_i\}_{i\in\mathcal{CS}}, \{\mathsf{ct}^j_{i,\ell}\}_{i\in[n],\ell\in QL,j\in[|Q_{i,\ell}|]}$, $\{\mathsf{ct}'^j_{i,\ell}\}_{i\in[n],\ell\in QL',j\in[|Q'_{i,\ell}|]})$ to $\mathcal{A}$.

Whenever the adversary $\mathcal{A}$ asks a key generation query $\mathsf{QKeyG}(\{f^i\}_{i\in[n]})$, $\mathcal{B}'''$ queries the underlying one-time pad for all the honest slots except one, i.e. for all $i \in \mathcal{HS} \setminus \{k\}$. In more detail, $\mathcal{B}'''$ queries the

underlying one-time pad with $(f^i(x_i^{j,0}), f^i(x_i^{j,1}))$ for all $i \in \mathcal{HS} \setminus \{k\}$, all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and all $\ell \in QL$. $\mathcal{B}'''$ also queries the underlying one-time pad with $(f^i(x_i^j), f^i(x_i^j))$ for all $i \in \mathcal{HS} \setminus \{k\}$, all $x_i^j \in Q'_{i,\ell}$ and all $\ell \in QL'$. To prove that this query made by $\mathcal{B}'''$ is valid, in the sense of the CON-PERF-IND$_\beta$ game, we need to show that for all $i \in \mathcal{HS} \setminus \{k\}$, all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and all $\ell \in QL$ it holds that $f^i(x_i^{1,1}) - f^i(x_i^{1,0}) = f^i(x_i^{j,1}) - f^i(x_i^{j,0})$. This follows immediately from the fact that a left-or-right query needs to be asked in every position and the fact that $f(x_1^{j,0}, \ldots, x_n^{j,0}) = f(x_1^{j,1}, \ldots, x_n^{j,1})$ for all $i \in [n]$, all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and all $\ell \in QL$, which is equivalent to $\sum_{i\in[n]} f^i(x_i^{j,0}) = \sum_{i\in[n]} f^i(x_i^{j,0})$ in the case of separable functions. In more detail, we consider the case in which a left-or-right query has been asked in every position at least once and another left-or-right query, $(x_{i^*}^{j,0}, x_{i^*}^{j,1})$, is made for the slot $i^*$. For the function evaluation of this query, it must hold that $\sum_{i\in[n]\setminus\{i^*\}} f^i(x_{i^*}^{1,0}) + f^i(x_{i^*}^{j,0}) = \sum_{i\in[n]\setminus\{i^*\}} f^i(x_i^{1,1}) + f^i(x_{i^*}^{j,1})$, which results in $f^i(x_{i^*}^{j,1}) - f^i(x_{i^*}^{j,0}) = \sum_{i\in[n]\setminus\{i^*\}} f^i(x_{i^*}^{1,0}) - \sum_{i\in[n]\setminus\{i^*\}} f^i(x_i^{1,1})$, since this holds for all $j \in [|Q_{i,\ell}|]$ it directly follows that $f^i(x_{i^*}^{1,1}) - f^i(x_{i^*}^{1,0}) = f^i(x_{i^*}^{j,1}) - f^i(x_{i^*}^{j,0})$ for all $(x_{i^*}^{j,0}, x_{i^*}^{j,1}) \in Q_{i^*,\ell}$ for all $\ell \in QL$. After showing that the condition of the CON-PERF-IND$_\beta$ game is fulfilled, we show that $\mathcal{B}'''$ perfectly simulates the key generation. After $\mathcal{B}'''$ received the replies $y_{i,\ell}^{j,f^i}$ of its queries $(f^i(x_i^{j,0}), f^i(x_i^{j,1}))$ for all $i \in \mathcal{HS} \setminus \{k\}$, all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and all $\ell \in QL$, it computes $e_\ell^{j,f} := f(x_1^{j,0}, \ldots, x_n^{j,0}) = f(x_1^{j,1}, \ldots, x_n^{j,1})$ for all $j \in [|Q_{i,\ell}|]$ and sets $y_{k,\ell}^{j,f^i} := e_\ell^{j,f} - (\sum_{i\in[n]\setminus\{k\}} y_{i,\ell}^{j,f^i})$ for all $j \in [|Q_{i,\ell}|]$. $\mathcal{B}'''$ sets $Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[|Q_{i,\ell}|]}, \{y_{i,\ell}^{\prime j,f^i}\}_{\ell\in QL', j\in[|Q'_{i,\ell}|]}\}$. In the final step, $\mathcal{B}'''$ generates $\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i,Y_i})$ for all $i \in \mathcal{HS}$ and $\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, f^i_{r_i})$ for all $i \in \mathcal{CS}$, sets $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}}\}_{i\in\mathcal{HS}} \cup \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in\mathcal{CS}}$ and sends $\mathsf{sk}_f$ to $\mathcal{A}$.

This shows the perfect simulation of $\mathsf{G}_{2+\beta}$. Finally, $\mathcal{B}'''$ outputs the same bit $\beta'$ returned by $\mathcal{A}$. Thus, we obtain the lemma. □

## 6 Adaptive Security

To prove the adaptive security of our construction, we face two main problems that do not occur in the case of selective security: First, we do not know all the honest slots in advance and therefore cannot directly replace the honest pseudorandom function evaluations with random function evaluations. The second problem is that we cannot encode all the function evaluations inside the functional keys since we do not know all the messages that are going to be queried in advance.

We overcome the first problem using a proof technique borrowed from [ABG19]. We define an explicitly honest slots (as in [ABG19]) as slots where the first left-or-right oracle query happens for different messages $x_i^0$ and $x_i^1$, i.e. $x_i^{1,0} \neq x_i^{1,1}$. Notice that if a slot $i$ is disclosed as explicitly honest it cannot be corrupted afterwards anymore and we can replace the pseudorandomness in this slot with real randomness (i.e. by relying on the security of the PRF). To know which slots are going to be explicitly honest, we will guess, at a very high level, the number of corrupted slots and the index of the first and the last slots that will be corrupted. This results only in a polynomial loss in the reduction instead of an exponential loss. Note that by Conditions (*) (see Definition 2.4) it follows that if an explicitly honest slots does not occur that we can relay on the security of the underlying functional encryption scheme. The details for this part of the proof are described below. To solve the second issue, we make use of the $\perp$ position in the different encryptions. In more detail, we create a list that contains all the functions that have already been queried to the key generation oracle. Whenever the adversary queries the left-or-right oracle or the encryption oracle on a new challenge, we place all the function evaluations for every previous queried functions inside the $\perp$ position of the ciphertext. Combining this with the approach from the selective security proof, we ensure that the function evaluation happens correctly no matter if the encryption or left-or-right oracle query happened before or after a functional key query. Since the ciphertext also contains function evaluations, we need to replace them together with function evaluations contained inside the functional key. This happens with the same information theoretic argument as in the selective security case extended to the ciphertexts. The formal proof is described below.

**Theorem 6.1** (*q*-message-and-key ad-pos$^+$-IND-security of MCFE). *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a q-message-and-key bounded* ad-FULL*-secure single-input functional encryption scheme for the functionality class* $\mathcal{F}_1^{\mathsf{sep}}$*, and* PRF *an* IND *secure pseudorandom function, then the MCFE scheme* MCFE *described in Fig. 9 is a q-message-and-key bounded* ad-pos$^+$-IND*-secure functional encryption scheme for the functionality class* $\mathcal{F}_n^{\mathsf{sep}}$*. Namely, for any PPT adversary* $\mathcal{A}$*, there exists PPT adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *such that:*

$$\mathsf{Adv}^{\text{ad-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda) \leq 2(n+1)n(n-1)^2 \cdot \mathsf{Adv}^{\text{IND}}_{\mathsf{PRF},\mathcal{B}}(\lambda) + 4(n+1)n \cdot \mathsf{Adv}^{\text{ad-FULL}}_{\mathsf{FE},\mathcal{B}'}(\lambda) \ ,$$

*Proof.* The arguments used for the generation of the values $t_{i,\ell}$ is based on the proof in [ABG19] and we recap those parts here adapted to our construction.

We denote by $Q_{i,\ell}$ the encryption queries of the form $(x_i^{j,0}, x_i^{j,1}, \ell)$ made for position $i$ and by $Q_{i,f}$ the key queries of the form $(f^i, r_i)$ made for every position $i \in [n]$.

For the case with only one honest (non-corrupted) position, we proceed in the same way as for the selective security case.

As in Abdalla et al. [ABG19], we consider a slot $i$ as explicitly honest if the first left-or-right oracle query $\mathsf{QLeftRight}(i, x_i^{1,0}, x_i^{1,1}, \ell)$ that has been made for this slot contains two challenges such that $x_i^{1,0} \neq x_i^{1,1}$. This can happen under any label $\ell$. For the slots that are not explicitly honest, i.e. it holds for the first query that $x_i^{1,0} = x_i^{1,1}$, it must hold that for all further queries $j$ in this slot that $f^i(x_i^{j,0}) = f^i(x_i^{j,1})$ (a detailed reasoning why this holds can be found in the proof of Lemma 6.6) and therefore the security of such a slot $i$ can be directly reduced to the security of the underlying functional encryption scheme.

For the cases with more than one honest position, we use a hybrid argument with the games defined in Fig. 14. This results in:

$$\mathsf{Adv}^{\text{ad-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) = |\mathsf{Win}^{\mathsf{G}_0^\star}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}_5^\star}_{\mathcal{A}}(\lambda, n)| \ .$$

We describe the different intermediate games in more detail:

**Game** $\mathsf{G}_0^\star$ The game is the same as the ad-pos$^+$-IND$_0$ game, but with the difference that the number of explicitly honest slots is guessed in advance. This happens by choosing a uniformly random $\kappa^\star \leftarrow \{0, \ldots, n\}$. This guess is necessary since the honest slots are not known in advance as in the selective case. The game behaves exactly as the ad-pos$^+$-IND$_0$ game, except that it outputs 0 and ignores $\mathcal{A}'$s output in the case that the guess $\kappa^\star$ was incorrect. Since the guess is correct with probability $\frac{1}{n+1}$, we have

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

**Game** $\mathsf{G}_1^\star$**:** We change the distribution of the $t_{i,\ell}$ values that are needed to answer left-or-right oracle QLeftRight and encryption oracle QEnc queries, for the case $\kappa^\star \geq 2$. For these, the $t_{i,\ell}$ vector gets computed as usual, but a share of a perfect $\kappa^\star$ out of $\kappa^\star$ secret sharing is added. This game is similar to the game $\mathsf{G}_1$ from Fig. 11 for the proof of Theorem 5.1. Similarly to Lemma 5.2, we justify this transition using the security of the PRF with the crucial difference that corruptions happen adaptive here. Therefore, the reduction does not know the set of honest slot in advance. Since guessing the entire set of explicitly honest slot would incur an exponential security loss, we gradually introduce the shares. Starting with 2 out of 2 perfect secret sharing, then 3 out of 3 until we reach the the $\kappa^\star$ out of $\kappa^\star$ secret sharing among all the queried slots. This gradual introduction happens via a hybrid argument that is described in Fig. 17. To go from one hybrid to another, we only require to guess a pair of slot $(i, j)$ (namely the first and the last slot to be reveled ) to use the security of the PRF on the key $\mathsf{K}_{i,j}$. Namely, in Lemma 6.2, we show that there exists a PPT adversary $\mathcal{B}_0$ such that:

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

**Game** $\mathsf{G}_2^\star$**:** We replace the encryptions of $(x_i^{j,0}, \perp, t_{i,\ell}, \ell)$ with the encryptions of $(x_i^{j,1}, Z_i, 0^\lambda, \ell)$ for all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$, all $\ell \in QL$ and all $i \in [n]$ in the left-or-right oracle and we replace the encryptions of $(x_i^j, \perp, t_{i,\ell}, \ell)$ with the encryptions of $(x_i^j, Z_i', 0^\lambda, \ell)$ for all $x_i^j \in Q'_{i,\ell}$ for all $\ell \in QL'$ in the encryption

oracle. The values $t_{i,\ell}$ in the left-or-right queries and the encryption queries are replaced with $0^\lambda$ to make the ciphertexts independent from the masking values $t_{i,\ell}$. The hardcoded values $z_{i,f}^{j} \in Z_i$ are generated using the values $(f^i, r_i) \in Q_{i,f}$, the queries $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $z_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$. The same holds for the hardcoded values $z_{i,\ell}^{\prime j,f^i} \in Z'_i$. They are generated using the values $(f^i, r_i) \in Q_{i,f}$, the queries $x_i^j \in Q'_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $z_{i,\ell}^{\prime j,f^i} := f^i(x_i^j) + t_{i,\ell} + r_i$. We also replace the functional key $\mathsf{sk}_f := \{\mathsf{sk}_{f_{r_i}^i}\}_{i\in[n]}$ (see Fig. 10b for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f_{\mathcal{Q}_i,Y_i}^i}\}_{i\in[n]}$ (see Fig. 16 for the function description). The hardcoded values $y_{i,\ell}^{j,f^i} \in Y_i$ are generated using the random value $r_i$, the queries $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$. The same holds for the hardcoded values $y_{i,\ell}^{\prime j,f^i}$. They are generated using the random value $r_i$, the queries $x_i^j \in Q'_{i,\ell}$ and by computing the masking values $t_{i,\ell}$, i.e. $y_{i,\ell}^{\prime j,f^i} := f^i(x_i^j) + t_{i,\ell} + r_i$. The transition from $\mathsf{G}_1^\star$ to $\mathsf{G}_2^\star$ is achieved using a hybrid argument with sequence $\mathsf{G}_{1.k}^\star$, for $k \in [n]$. As already described in Fig. 14, it holds that $\mathsf{G}_1^\star = \mathsf{G}_{1.0}^\star$ and $\mathsf{G}_2^\star = \mathsf{G}_{1.n}^\star$. This results in

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda, n)| \leq \sum_{k=1}^{n} |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}^\star}(\lambda, n)| \ ,$$

for any PPT adversary $\mathcal{A}$. The transition from $\mathsf{G}_{1.k-1}^\star$ to $\mathsf{G}_{1.k}^\star$ is justified by the full security of $\mathsf{FE}$. Namely, in Lemma 5.4, we exhibit a PPT adversary $\mathcal{B}_k$ for all $k \in [n]$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}^\star}(\lambda, n)| \leq \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_k}^{\text{sel-FULL}}(\lambda) \ .$$

combining both of the statements and noticing that a PPT adversary $\mathcal{B}_1$ can be obtained by picking $i \in [n]$ and running $\mathcal{B}_i$, we can justify the transition from $\mathsf{G}_1^\star$ to $\mathsf{G}_2^\star$. Namely, in Lemma 5.3, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda, n)| \leq n \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_1}^{\text{sel-FULL}}(\lambda) \ .$$

**Game $\mathsf{G}_3^\star$:** We change the generation of all the values $y_{i,\ell}^{j,f^i} \in Y_i$ and $z_{i,\ell}^{j,f^i} \in Z_i$, which are computed using the random values $r_i$, the queries $Q_{i,\ell}$ and $Q_{i,f}$ and the masking values $t_{i,\ell}$. We change the generation from $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$ to $y_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$ and from $z_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$ to $z_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$. The transition from $\mathsf{G}_2^\star$ to $\mathsf{G}_3^\star$ is justified by an information theoretic argument and happens for all $i \in [n]$. In more detail, we prove the transition by relying on the conditioned perfect security of several instances of the one-time pad as shown in Lemma 3.6. Namely, in Lemma 6.6, we show that

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3^\star}(\lambda, n)| = 0 \ ,$$

for all adversaries $\mathcal{A}$.

**Game $\mathsf{G}_4^\star$:** We replace the encryptions of $(x_i^{j,1}, Z_i, 0^\lambda, \ell)$ with the encryptions of $(x_i^{j,1}, \perp, t_{i,\ell}, \ell)$ for all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$, all $\ell \in QL$ and all $i \in [n]$ in the left-or-right oracle and we replace the encryptions of $(x_i^j, Z'_i, 0^\lambda, \ell)$ with the encryptions of $(x_i^j, \perp, t_{i,\ell}, \ell)$ for all $x_i^j \in Q'_{i,\ell}$, all $\ell \in QL'$ and all $i \in [n]$ in the encryption oracle. The masking values $t_{i,\ell}$ are inserted back into the ciphertext and replace the $0^\lambda$ values. We also replace the functional key $\mathsf{sk}_f := \{\mathsf{sk}_{f_{\mathcal{Q}_i,Y_i}^i}\}_{i\in[n]}$ (see Fig. 10b for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f_{r_i}^i}\}_{i\in[n]}$ (see Fig. 16 for the function description). The transition from $\mathsf{G}_3^\star$ to $\mathsf{G}_4^\star$ is almost symmetric to the transition from $\mathsf{G}_1^\star$ to $\mathsf{G}_2^\star$, justified by the full security of $\mathsf{FE}$ applied on every slot $i \in [n]$. Namely, it can be proven as in Lemma 6.4 that there exists a PPT adversary $\mathcal{B}_2$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_3^\star}(\lambda, n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4^\star}(\lambda, n)| \leq n \cdot \mathsf{Adv}_{\mathsf{FE},\mathcal{B}_2}^{\text{ad-FULL}}(\lambda) \ .$$

We defer to the proof of Lemma 6.4 for further details.

**Game $\mathsf{G}_5^\star$:** The transition from $\mathsf{G}_4^\star$ to $\mathsf{G}_5^\star$ is almost 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 5.2 that there exists a PPT adversary $\mathcal{B}_3$ such that:

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_4^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_5^\star}(\lambda,n)| \leq n(n-1)^2 \cdot \mathsf{Adv}^{\text{-IND}}_{\mathsf{PRF},\mathcal{B}_3}(\lambda,n) \ .$$

Since $\mathsf{G}_5^\star$ is exactly as the game ad-pos$^+$-IND$_1^{\mathsf{MCFE}}(\lambda,n,\mathcal{A})$ except the guess $\kappa^\star \leftarrow \{0,\ldots,n\}$, we have

$$\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_5^\star}(\lambda,n) = \frac{1}{(n+1)} \cdot \mathsf{Win}^{\text{xx-yy-IND}_1}(\lambda,n).$$

Putting everything together, we obtain the theorem. □

| Game | $\mathsf{ct}_{i,\ell}^j$ | $\mathsf{ct}_{i,\ell}'^j$ | $\mathsf{sk}_f$ | justification/ remark |
|---|---|---|---|---|
| $\mathsf{G}_0^\star$ | $t_{i,\ell} := \mathsf{Gen}(\mathsf{sk}_i,i,\ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j,0},\bot,t_{i,\ell},\ell))$ | $t_{i,\ell} := \mathsf{Gen}(\mathsf{sk}_i,i,\ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j},\bot,t_{i,\ell},\ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f_{r_i}^i)$ | |
| $\mathsf{G}_1^\star$ | $t_{i,\ell} := \boxed{\mathsf{Gen}''(\mathsf{sk}_i,i,\ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j,0},\bot,t_{i,\ell},\ell))$ | $t_{i,\ell} := \boxed{\mathsf{Gen}''(\mathsf{sk}_i,i,\ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j},\bot,t_{i,\ell},\ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f_{r_i}^i)$ | PRF |
| $\mathsf{G}_{1.k}^\star$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i,i,\ell)$<br>If $i \leq k$<br>Add $(x_i^{j,0},x_i^{j,1})$ to $Q_{i,\ell}$<br>For all $(f^i,r_i) \in Q_{i,f}$<br>$z_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>$Z_i := \{z_{i,\ell}^{j,f^i}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(\boxed{x_i^{j,1}, Z_i, 0^\lambda},\ell))$,<br>for $i \leq k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j,0},\bot,t_{i,\ell},\ell))$,<br>for $i > k$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i,i,\ell)$<br>If $i \leq k$<br>Add $x_i^j$ to $Q'_{i,\ell}$<br>For all $(f^i,r_i) \in Q_{i,f}$<br>$z'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j},\boxed{Z'_i, 0^\lambda},\ell))$,<br>for $i \leq k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i,(x_i^{j},\bot,t_{i,\ell},\ell))$,<br>for $i > k$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>If $i \leq k$<br>Add $(f^i,r_i)$ to $Q_{i,f}$<br>For all $\ell \in QL$,<br>$(x_i^{j,0},x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i,i,\ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i,i,\ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j}) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q'_{i,\ell}\rvert]}\}$<br>If $i \leq k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i,Y_i})$<br>If $i > k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f_{r_i}^i)$ | Full Security of FE |

Fig. 14a: Description of the games $\mathsf{G}_0^\star$ to $\mathsf{G}_{1.k}^\star$ for the proof of adaptive security.

| Game | $\mathsf{ct}^j_{i,\ell}$ | $\mathsf{ct}'^j_{i,\ell}$ | $\mathsf{sk}_f$ | justification/ remark |
|---|---|---|---|---|
| $\mathsf{G}_2^\star$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>Add $(x_i^{j,0}, x_i^{j,1})$ to $Q_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$z_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>$Z_i := \{z_{i,\ell}^{j,f^i}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (\boxed{x_i^{j,1}, Z_i, 0^\lambda}, \ell))$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>Add $x_i^j$ to $Q'_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$z'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^j, \boxed{Z'_i, 0^\lambda}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>Add $(f^i, r_i)$ to $Q_{i,f}$<br>For all $\ell \in QL$,<br>$(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j,0}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q_{i,\ell}\rvert]}\}$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | $\mathsf{G}_2^\star = \mathsf{G}_{1.n}^\star$ |
| $\mathsf{G}_3^\star$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>Add $(x_i^{j,0}, x_i^{j,1})$ to $Q_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$\boxed{z_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i}$<br>$Z_i := \{z_{i,\ell}^{j,f^i}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, Z_i, 0^\lambda, \ell))$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>Add $x_i^j$ to $Q'_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$z'^{j,f^i}_{i,\ell} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^j, Z'_i, 0^\lambda, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda, \forall i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>Add $(f^i, r_i)$ to $Q_{i,f}$<br>For all $\ell \in QL$,<br>$(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$\boxed{y_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i}$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^j) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q_{i,\ell}\rvert]}\}$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | inf. theoretic |

Fig. 14b: Description of the games $\mathsf{G}_2^\star$ to $\mathsf{G}_3^\star$ for the proof of adaptive security.

| Game | $\mathsf{ct}^{j}_{i,\ell}$ | $\mathsf{ct}'^{j}_{i,\ell}$ | $\mathsf{sk}_f$ | justification/ remark |
|---|---|---|---|---|
| $\mathsf{G}^{\star}_{3.k}$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>If $i > k$<br>Add $(x_i^{j,0}, x_i^{j,1})$ to $Q_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$z_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$<br>$Z_i := \{z_{i,\ell}^{j,f^i}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, \boxed{\perp, t_{i,\ell}}, \ell))$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, Z_i, 0^\lambda, \ell))$, for $i > k$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>If $i > k$<br>Add $x_i^{j}$ to $Q'_{i,\ell}$<br>For all $(f^i, r_i) \in Q_{i,f}$<br>$z'^{j,f^i}_{i,\ell} := f^i(x_i^{j}) + t_{i,\ell} + r_i$<br>$Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{\perp, t_{i,\ell}}, \ell))$, for $i \le k$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, Z'_i, 0^\lambda, \ell))$, for $i > k$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>If $i > k$<br>Add $(f^i, r_i)$ to $Q_{i,f}$<br>For all $\ell \in QL$, $(x_i^{j,0}, x_i^{j,1}) \in Q_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j,1}) + t_{i,\ell} + r_i$<br>For all $\ell \in QL', x_i^j \in Q'_{i,\ell}$<br>$t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$y_{i,\ell}^{j,f^i} := f^i(x_i^{j}) + t_{i,\ell} + r_i$<br>$\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$<br>$Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[\lvert Q_{i,\ell}\rvert]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[\lvert Q_{i,\ell}\rvert]}\}$<br>If $i \le k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$<br>If $i > k$:<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i, Y_i})$ | Full Security of FE |
| $\mathsf{G}^{\star}_{4}$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, \boxed{\perp, t_{i,\ell}}, \ell))$ | $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \boxed{\perp, t_{i,\ell}}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\boxed{\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})}$ | $\mathsf{G}^{\star}_{4} = \mathsf{G}^{\star}_{3.n}$ |
| $\mathsf{G}^{\star}_{5}$ | $t_{i,\ell} := \boxed{\mathsf{Gen}(\mathsf{sk}_i, i, \ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j,1}, \perp, t_{i,\ell}, \ell))$ | $t_{i,\ell} := \boxed{\mathsf{Gen}(\mathsf{sk}_i, i, \ell)}$<br>$\mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i^{j}, \perp, t_{i,\ell}, \ell))$ | $r_i \leftarrow \mathcal{Y}_\lambda$, for all $i \in [n-1]$<br>$r_n := -\sum_{i\in[n-1]} r_i$<br>$\mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ | PRF |

Fig. 14c: Description of the games $\mathsf{G}^{\star}_{3.k}$ to $\mathsf{G}^{\star}_{5}$ for the proof of adaptive security.

| $\mathsf{Gen}(\mathsf{sk}_i, i, \ell)$ | $\mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$ |
|---|---|
| Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ | Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ |
| $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$ | $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$ |
| Return $t_{i,\ell}$ | We denote by $\{i_1, \ldots, i_\kappa\}$ the set of explicitly honest slots in the order they are revealed. |
| | If $i \in \{i_1, \ldots, i_\kappa\}$, then: |
| | • If $i = i_1$, $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) + \sum_{s=2}^{\kappa^\star} u_{s,\ell}$. |
| | • If $i = i_s$, for $s \in \{2, \ldots, \kappa^\star\}$, $t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell) - u_{s,\ell}$. |
| | Return $t_{i,\ell}$ |

Fig. 15: Generation of the tags for the labels in the adaptive case

$f^i_{\mathcal{Q}_i, Y_i}(x, Z_i, t_{i,\ell}, \ell)$ :

Parse $\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$,
$Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{j\in[Q_\ell],\ell\in QL}, \{y'^{j,f^i}_{i,\ell}\}_{j\in[Q'_\ell],\ell\in QL'}\}$

If $(\cdot, x) \in Q_{i,\ell}$
  Parse $Z_i := \{z^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$
  If $y^{j,f^i}_{i,\ell}$ is defined
    Return $y^{j,f^i}_{i,\ell}$
  Return $z^{j,f^i}_{i,\ell}$

If $x \in Q'_{i,\ell}$
  Parse $Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$
  If $y'^{j,f^i}_{i,\ell}$ is defined
    Return $y'^{j,f^i}_{i,\ell}$
  Return $z'^{j,f^i}_{i,\ell}$

Fig. 16: Description of the function that is used in the reduction for the adaptive security

$\underline{\mathsf{G}^{\star}_{0.k} \text{ for } k \in \{0,\dots,n\}:}$

$\kappa^{\star} \leftarrow \{0,\dots,n\},$ for all $s \in \{2,\dots,\kappa^{\star}\}$ for all $\ell \in [q], u_{s,\ell} \leftarrow \mathcal{Y}_{\lambda}$

$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}'(1^{\lambda}, n)$

$\alpha \leftarrow \mathcal{A}^{\mathsf{QCor}(\cdot),\mathsf{QKeyG}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot)}(1^{\lambda})$

Output $\alpha$ if Condition (*) is satisfied AND the guesses $\kappa^{\star}$ and $\lambda^{\star}$ are correct, or 0 otherwise.

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

Return $\mathsf{sk}_i$

$\underline{\mathsf{QKeyG}(f):}$

Return $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}'(\mathsf{msk}, f)$

$\underline{\mathsf{QEnc}(i, x_i^j, \ell):}$

Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]}), v_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$.

We denote by $\{i_1,\dots,i_{\kappa}\}$ the set of explicitly honest slots in the order they are revealed ($i_1$ is the first honest slot, $i_2$ the second, ...), and we set $\theta := \min(\kappa^{\star}, k)$.

If $\theta \geq 2$ then do the following:

- If $i = i_1,$ then $t_{i,\ell} := v_{i,\ell} + \sum_{s=2}^{\theta} u_{s,\ell}$
- If $i = i_s,$ for $s \in \{2,\dots,\theta\},$ then $t_{i,\ell} := v_{i,\ell} - u_{s,\ell}$
- If $i = i_s,$ for $s \in \{\theta+1,\dots,\kappa^{\star}\},$ then $t_{i,\ell} := v_{i,\ell}$
- If $i = i_s,$ for $s > \kappa^{\star},$ that means $k > \kappa^{\star}$, the guess was incorrect. Ends the game and outputs 0.

If $\theta < 2$, then $t_{i,\ell} := v_{i,\ell}$.

Return $\mathsf{Enc}'(\mathsf{sk}_i, (x_i^j, \perp, t_{i,\ell}, \ell))$

$\underline{\mathsf{QLeftRight}(i, x_i^{j,0}, x_i^{j,1}, \ell):}$

Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]}), v_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$.

We denote by $\{i_1,\dots,i_{\kappa}\}$ the set of explicitly honest slots in the order they are revealed, and we set $\theta := \min(\kappa^{\star}, k)$.

If $\theta \geq 2$ then do the following:

- If $i = i_1,$ then $t_{i,\ell} := v_{i,\ell} + \sum_{s=2}^{\theta} u_{s,\ell}$
- If $i = i_s,$ for $s \in \{2,\dots,\theta\},$ then $t_{i,\ell} := v_{i,\ell} - u_{s,\ell}$
- If $i = i_s,$ for $s \in \{\theta+1,\dots,\kappa^{\star}\},$ then $t_{i,\ell} := v_{i,\ell}$
- If $i = i_s,$ for $s > \kappa^{\star},$ that means $k > \kappa^{\star}$, the guess was incorrect. Ends the game and outputs 0.

If $\theta < 2$, then $t_{i,\ell} := v_{i,\ell}$.

Return $\mathsf{Enc}'(\mathsf{sk}_i, (x_i^{j,0}, \perp, t_{i,\ell}, \ell))$

Fig. 17: Games for the proof of Lemma 6.2. The guess $\kappa^{\star}$ is correct if it equals the size of the set of explicitly honest slots.

**Lemma 6.2 (Transition from $\mathsf{G}_0^\star$ to $\mathsf{G}_1^\star$).** *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}'$ such that*

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

*Proof.* To prove that $\mathsf{G}_0^\star$ is indistinguishable from $\mathsf{G}_1^\star$ we need to apply a hybrid argument over the explicitly honest users by relying on the security of the PRF.

Using the definition of the games in Figs. 14 and 17 and the triangular inequality, we can see that

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_0^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda,n)| \leq \sum_{k=2}^{n} |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k-1}^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k}^\star}(\lambda,n)| \ ,$$

where $\mathsf{G}_0^\star$ corresponds to game $\mathsf{G}_{0.0}^\star$ (and $\mathsf{G}_{0.1}^\star$) and whereas $\mathsf{G}_1^\star$ is identical to game $\mathsf{G}_{0.n}^\star$. Since $\mathsf{G}_{0.0}^\star = \mathsf{G}_{0.1}^\star$, we do not analyze the transition between these two games.

Now, we can bound the difference between each consecutive pair of games for every $k \in \{2,\ldots,n\}$.

**Lemma 6.3.** *For every $k \in \{2,\ldots,n\}$, there exists a PPT adversary $\mathcal{B}_k$ against the* IND *property of* PRF *such that*

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k-1}^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k}^\star}(\lambda,n)| \leq n(n-1) \cdot \mathsf{Adv}^{\mathrm{IND}}_{\mathsf{PRF},\mathcal{B}_k}(\lambda) \ .$$

*Proof.* This proof works mainly as described in [ABG19].

We build an adversary $\mathcal{B}_k$ that simulates $\mathsf{G}_{0.k-1+\beta}^\star$ for $k \in \{2,\ldots,n\}$ to $\mathcal{A}$ when interacting with the underlying $\mathrm{IND}_\beta^{\mathsf{PRF}}$ experiment.

If $\kappa^\star < 2$, the games $\mathsf{G}_{0.k-1}^\star$ and $\mathsf{G}_{0.k}^\star$ are the same. Therefore, we only consider the case where $\kappa^\star \geq 2$.

The adversary $\mathcal{B}_k$ starts by guessing the first and $k$'th honest slots $i^\star$ and $j^\star$, by sampling random $i^\star, j^\star \leftarrow [n]$, with $i^\star < j^\star$. Whenever $\mathcal{A}$ asks an encryption or left-or-right oracle query, $\mathcal{B}_k$ replies as described in Fig. 17 for every explicitly honest slot in the order they are revealed. Since $\mathcal{B}_k$ has guessed the first and $k$'th explicitly honest slot, it knows how to answer the queries for every explicitly honest slot that is revealed in between. If it turns out that the guess of $\mathcal{B}_k$ is incorrect, the simulation ends and returns 0. If the guess turns out to be correct, we can rely on the security of the PRF on the key $\mathsf{K}_{i^\star,j^\star}$ to a uniformly random value $\mathsf{RF}(\ell)$. Then we argue that $\mathsf{RF}(\ell)$ is identically distributed to $\mathsf{RF}(\ell) + u_{s,\ell}$ and therefore, since the former distribution corresponds to $\mathsf{G}_{0.k-1}^\star$ and the latter game $\mathsf{G}_{0.k}^\star$, the computational indistinguishability between $\mathsf{G}_{0.k}^\star$ and $\mathsf{G}_{0.k-1}^\star$ follows. We analyze the advantage of this transition in more detail. The guessing of the two honest slots happens with probability $\frac{2}{n(n-1)}$, which results in a security loss of $\frac{n(n-1)}{2}$, i.e. $\frac{2}{n(n-1)} \cdot |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k-1}^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k}^\star}(\lambda,n)| \leq \mathsf{Adv}^{\mathrm{IND}}_{\mathsf{PRF},\mathcal{B}_k}(\lambda) \Leftrightarrow |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k-1}^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{0.k}^\star}(\lambda,n)| \leq \frac{n(n-1)}{2} \cdot \mathsf{Adv}^{\mathrm{IND}}_{\mathsf{PRF},\mathcal{B}_k}(\lambda)$. Finally, we switch $\mathsf{RF}(\ell)$ back to $\mathsf{PRF}_{\mathsf{K}_{i^\star,j^\star}}(\ell)$ using the security of the PRF on the key $\mathsf{K}_{i^\star,j^\star}$ a second time. This results in the advantage described in the lemma.

For every corruption query $\mathsf{QCor}(i)$, $\mathcal{B}_k$ just returns $\mathsf{sk}_i$ and for every key generation query $\mathsf{QKeyG}(f)$, $\mathcal{B}_k$ just computes and returns $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}'(\mathsf{msk}, f)$. □

□

**Lemma 6.4 (Transition from $\mathsf{G}_1^\star$ to $\mathsf{G}_2^\star$).** *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}''$ such that*

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda,n)| \leq n \cdot \mathsf{Adv}^{\text{ad-FULL}}_{\mathsf{FE},\mathcal{B}''}(\lambda) \ .$$

*Proof.* To prove that $\mathsf{G}_1^\star$ is indistinguishable from $\mathsf{G}_2^\star$ we need to apply a hybrid argument over the $n$ slots, using the full security of the single-input functional encryption scheme.

Using the definition of the games in Fig. 14 and the triangular inequality, we can see that

$$|\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_1^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_2^\star}(\lambda,n)| \leq \sum_{k=1}^{n} |\mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k-1}^\star}(\lambda,n) - \mathsf{Win}_{\mathcal{A}}^{\mathsf{G}_{1.k}^\star}(\lambda,n)| \ ,$$

where $\mathsf{G}_1^\star$ corresponds to game $\mathsf{G}_{1.0}^\star$ and whereas $\mathsf{G}_2^\star$ is identical to game $\mathsf{G}_{1.n}^\star$.

Now, we can bound the difference between each consecutive pair of games for every $k \in [n]$.

**Lemma 6.5.** *For every $k \in [n]$, there exists a PPT adversary $\mathcal{B}'_k$ against the* ad-FULL *property of the single-input scheme* $\mathsf{FE}$ *such that*

$$|\mathsf{Win}^{\mathsf{G}^\star_{1.k-1}}_{\mathcal{A}}(\lambda,n) - \mathsf{Win}^{\mathsf{G}^\star_{1.k}}_{\mathcal{A}}(\lambda,n)| \leq \mathsf{Adv}^{\text{ad-FULL}}_{\mathsf{FE},\mathcal{B}'_k}(\lambda) \ .$$

*Proof.* We build an adversary $\mathcal{B}'_k$ that simulates $\mathsf{G}^\star_{1.k-1+\beta}$ to $\mathcal{A}$ when interacting with the underlying sel-FULL$^{\mathsf{FE}}_\beta$ experiment.

We denote by $\mathcal{EHS}$ the set of explicitly honest slots and by $\mathcal{RS}$ the set of remaining slots. We also denote the labels queried to left-or-right oracle by $QL$ and to the encryption oracle by $QL'$ upper-bounded by $q$. Before answering any oracle queries, the adversary $\mathcal{B}'_k$ initializes the lists $Q_{i,\ell}$ for all $i \in [n]$ and all $\ell \in QL$, the lists $Q'_{i,\ell}$ for all $i \in [n]$ and all $\ell \in QL'$ and the list $Q_{i,f}$ for all $i \in [n]$. Afterwards, $\mathcal{B}'_k$ generates $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n] \setminus \{k\}$, samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i \in [n], i < j$ and sets $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$.

For every corruption query $\mathsf{QCor}(i)$ with $i \neq k$, $\mathcal{B}'_k$ sends $\mathsf{sk}_i$ to $\mathcal{A}$. If a corruption query is asked for $k$, the adversary $\mathcal{B}'_k$ directly outputs $\alpha \leftarrow \{0,1\}$. This is due to the fact that the games $\mathsf{G}^\star_{1.k-1}$ and $\mathsf{G}^\star_{1.k}$ are identical in this case, which results in an advantage equal to 0 and Lemma 6.5 trivially holds. The same happens in the case that $k$ is not an explicitly honest slot.

Whenever $\mathcal{A}$ asks a left-or-right query $(i, x^{j,0}_i, x^{j,1}_i, \ell)$, $\mathcal{B}'_k$ adds $(x^{j,0}_i, x^{j,1}_i)$ to the list $Q_{i,\ell}$ and computes $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$. To generate the final ciphertext $\mathcal{B}'_k$ proceeds different corresponding to the different positions $i$. For $i < k$ and $i \in \mathcal{EHS}$, $\mathcal{B}'_k$ computes $z^{j,f^i}_{i,\ell} := f^i(x^{j,0}_i) + t_{i,\ell} + r_i$ for all $(f^i, r_i) \in Q_{i,f}$, sets $Z_i := \{z^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$ and computes $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x^{j,1}_i, Z_i, 0^\lambda, \ell))$. For $i = k$, $\mathcal{B}'_k$ computes $z^{j,f^i}_{i,\ell} := f^k(x^{j,0}_k) + t_{k,\ell} + r_k$ for all $(f^k, r_k) \in Q_{k,f}$, sets $Z_k := \{z^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$ and submits $((x^{j,0}_k, \bot, t_{k,\ell}, \ell), (x^{j,1}_k, Z_k, 0^\lambda, \ell))$ to its own left-or-right oracle and receives the ciphertext $\mathsf{ct}^j_{k,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_k, (x^{j,\beta}_k, Z^\beta, t^\beta, \ell))$ (with $Z^0 = \bot$, $Z^1 = Z_k$ and $t^0 = t_{k,\ell}$, $t^1 = 0^\lambda$) as an answer. For $i > k$ and $i \in \mathcal{RS}$, $\mathcal{B}'_k$ computes $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x^{j,1}_i, \bot, t_{i,\ell}, \ell))$. Finally $\mathcal{B}'_k$ sends $\mathsf{ct}^j_{i,\ell}$ as a reply to $\mathcal{A}$.

The adversary $\mathcal{B}'_k$ behaves similar to answer an encryption oracle query. In more detail, whenever $\mathcal{A}$ asks an encryption query $(i, x^j_i, \ell)$, $\mathcal{B}'_k$ adds $x^j_i$ to the list $Q'_{i,\ell}$ and computes $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$. To generate the final ciphertexts $\mathcal{B}'_k$ proceeds different corresponding to the different positions $i$. For $i < k$ and $i \in \mathcal{EHS}$, $\mathcal{B}'_k$ computes $z'^{j,f^i}_{i,\ell} := f^i(x^j_i) + t_{i,\ell} + r_i$ for all $(f^i, r_i) \in Q_{i,f}$, sets $Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$ and computes $\mathsf{ct}'^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x^j_i, Z'_i, 0^\lambda, \ell))$. For $i = k$, $\mathcal{B}'_k$ computes $z'^{j,f^i}_{i,\ell} := f^k(x^j_k) + t_{k,\ell} + r_k$ for all $(f^k, r_k) \in Q_{k,f}$, sets $Z'_k := \{z'^{j,f^i}_{i,\ell}\}_{f^k\in Q_{k,f}}$ and submits $((x^j_k, \bot, t_{k,\ell}, \ell), (x^j_k, Z'_k, 0^\lambda, \ell))$ to its own left-or-right oracle and receives the ciphertext $\mathsf{ct}'^j_{k,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_k, (x^j_k, Z'^\beta, t^\beta, \ell))$ (with $Z'^0 = \bot$, $Z'^1 = Z'_k$ and $t^0 = t_{k,\ell}$, $t^1 = 0^\lambda$) as an answer. For $i > k$ and $i \in \mathcal{RS}$, $\mathcal{B}'_k$ computes $\mathsf{ct}'^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x^j_i, \bot, t_{i,\ell}, \ell))$. Finally $\mathcal{B}'_k$ sends $\mathsf{ct}'^j_{i,\ell}$ as a reply to $\mathcal{A}$.

Whenever the adversary $\mathcal{A}$ asks a key generation query $\mathsf{QKeyG}(\{f^i\}_{i\in[n]})$, $\mathcal{B}'_k$ samples $r_i \leftarrow \mathcal{Y}_\lambda$ for all $i \in [n]$ and sets $r_n := -\sum_{i\in[n-1]} r_i$. Then $\mathcal{B}'_k$ adds $(f^i, r_i)$ to the list $Q_{i,f}$. For all the left-or-right oracle queries $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ for all $\ell \in QL$, $\mathcal{B}'_k$ generates $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$ for all $i \in [k] \setminus \mathcal{RS}$ and computes $y^{j,f^i}_{i,\ell} := f^i(x^{j,0}_i) + t_{i,\ell} + r_i$ for all $j \in [Q_\ell]$. For all the encryption queries $x^j_i \in Q'_{i,\ell}$ for all $\ell \in QL'$, $\mathcal{B}'_k$ generates $t_{i,\ell} := \mathsf{Gen}''(i, \ell)$ for all $i \in [k] \setminus \mathcal{RS}$ and computes $y'^{j,f^i}_{i,\ell} := f^i(x^j_i) + t_{i,\ell} + r_i$ for all $j \in [Q'_\ell]$. $\mathcal{B}'_k$ sets $Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{j\in[Q_\ell],\ell\in QL}, \{y'^{j,f^i}_{i,\ell}\}_{j\in[Q'_\ell],\ell\in QL'}\}$ and computes $\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i,Y_i})$ for the slots $i < k$ with $i \in \mathcal{EHS}$ and $\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$ for $i > k$ or $i \in \mathcal{RS}$. To generate $\mathsf{sk}_{f^k}$, $\mathcal{B}'_k$ queries its own key generation oracle $\mathsf{QKeyG}$ on $(f^k_{r_k}, f^k_{\mathcal{Q}_k,Y_k})$, this query fulfills the functional restriction, i.e. it holds that $f^k_{r_k}(x^{j,0}_k) = f^k_{\mathcal{Q}_k,Y_k}(x^{j,0}_k)$ for all $(x^{j,0}_k, \cdot) \in Q_{k,\ell}$ for all $\ell \in QL$. $\mathcal{B}'_k$ receives $\mathsf{sk}_{f^k}$ as an answer, sets $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}}\}_{i\in[k-1]\setminus\mathcal{RS}} \cup \{\mathsf{sk}_{f^k}\} \cup \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in(\{k+1,\ldots,n\}\cup\mathcal{RS})}$ and sends $\mathsf{sk}_f$ to $\mathcal{A}$.

This covers the simulation of the game $\mathsf{G}^\star_{1.k-1+\beta}$. Finally, $\mathcal{B}'_k$ outputs the same bit $\beta'$ returned by $\mathcal{A}$. Thus, we obtain the lemma. ∎

The proof of the lemma follows by noticing that the adversary $\mathcal{B}''$ in the lemma statement can be obtained by picking $k \in [n]$ and running $\mathcal{B}'_k$. $\square$

**Lemma 6.6 (Transition from $\mathsf{G}^\star_2$ to $\mathsf{G}^\star_3$).** *For any adversary $\mathcal{A}$ it holds that*

$$|\mathsf{Win}^{\mathsf{G}^\star_2}_{\mathcal{A}}(\lambda, n) - \mathsf{Win}^{\mathsf{G}^\star_3}_{\mathcal{A}}(\lambda, n)| = 0 \ .$$

*Proof.* We build a PPT adversary $\mathcal{B}'''$ that simulates $\mathsf{G}^\star_{2+\beta}$ to $\mathcal{A}$, when interacting with $(|\mathcal{EHS}| - 1) \cdot \lambda^\star \cdot |Q_f|$ instances of the one-time pad in the CON-PERF-IND$_\beta$ experiment, as proven in Lemma 3.6. With many-instances we mean that encryption oracle of the one-time pad can be queried several times, but always the same position (left or right) is encrypted. In more detail, a new key $\mathsf{k}^{f^i}_{i,\ell}$ is chosen for every position $i \in [n]$, for every label $\ell \in QL$ and for every function $\{f^i\}_{i\in[n]} \in Q_f$.

We denote by $\mathcal{EHS}$ the set of explicitly honest slots and by $\mathcal{RS}$ the set of remaining slots. We also denote the labels queried to left-or-right oracle by $QL$ and to the encryption oracle by $QL'$ upper-bounded by $q$. Before answering any oracle queries, the adversary $\mathcal{B}'''$ initializes the lists $Q_{i,\ell}$ for all $i \in [n]$ and all $\ell \in QL$, the lists $Q'_{i,\ell}$ for all $i \in [n]$ and all $\ell \in QL'$ and the list $Q_{i,f}$ for all $i \in [n]$. Afterwards, $\mathcal{B}'''$ generates $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$ for all $i \in [n]$, samples $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ for all $i \in [n], i < j$ and sets $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$ for all $i \in [n]$.

For every corruption query $\mathsf{QCor}(i)$ asked by $\mathcal{A}$, $\mathcal{B}'''$ sends $\mathsf{sk}_i$ to $\mathcal{A}$.

Whenever the adversary $\mathcal{A}$ asks a key generation query $\mathsf{QKeyG}(\{f^i\}_{i\in[n]})$, $\mathcal{B}'''$ adds $(f^i, r_i)$ to $Q_{i,f}$. Afterwards, $\mathcal{B}'''$ queries the underlying one-time pad for the explicitly honest slots $i \in \mathcal{EHS}$ unequal to the $\kappa^\star$'th explicitly honest slot. In more detail, $\mathcal{B}'''$ queries the underlying one-time pad with $(f^i(x^{j,0}_i)+t'_{i,\ell}, f^i(x^{j,1}_i)+t'_{i,\ell})$ with $t'_{i,\ell} = \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$, for all $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ and all $\ell \in QL$. $\mathcal{B}'''$ also queries the underlying one-time pad with $(f^i(x^j_i) + t'_{i,\ell}, f^i(x^j_i) + t'_{i,\ell})$ with $t'_{i,\ell} = \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$, for all $x^j_i \in Q'_{i,\ell}$ and all $\ell \in QL'$. To prove that this query made by $\mathcal{B}'''$ is valid, in the sense of the CON-PERF-IND$_\beta$ game, we need to show that for all $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ and for all $\ell \in QL$ it holds that $f^i(x^{1,1}_i) - f^i(x^{1,0}_i) = f^i(x^{j,1}_i) - f^i(x^{j,0}_i)$. This follows immediately from the fact that a left-or-right query needs to be asked in every position, and the fact that $f(x^{j,0}_1, \ldots, x^{j,0}_n) = f(x^{j,1}_1, \ldots, x^{j,1}_n)$ for all $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ and for all $\ell \in QL$, which is equivalent to $\sum_{i\in[n]} f^i(x^{j,0}_i) = \sum_{i\in[n]} f^i(x^{j,0}_i)$ in the case of separable functions. In more detail, we consider the case in which a left-or-right query has been asked in every position at least once and another left-or-right query, $(x^{j,0}_{i^*}, x^{j,1}_{i^*})$, is made for the slot $i^*$. For the function evaluation of this query, it must hold that $\sum_{i\in[n]\setminus\{i^*\}} f^i(x^{1,0}_{i^*}) + f^i(x^{j,0}_{i^*}) = \sum_{i\in[n]\setminus\{i^*\}} f^i(x^{1,1}_i) + f^i(x^{j,1}_{i^*})$, which results in $f^i(x^{j,1}_{i^*}) - f^i(x^{j,0}_{i^*}) = \sum_{i\in[n]\setminus\{i^*\}} f^i(x^{1,0}_{i^*}) - \sum_{i\in[n]\setminus\{i^*\}} f^i(x^{1,1}_i)$, since this holds for all $j \in [Q_\ell]$ it directly follows that $f^i(x^{1,1}_{i^*}) - f^i(x^{1,0}_{i^*}) = f^i(x^{j,1}_{i^*}) - f^i(x^{j,0}_{i^*})$ for all $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ for all $\ell \in QL$. After showing that the condition of the CON-PERF-IND$_\beta$ game is fulfilled, we show that $\mathcal{B}'''$ perfectly simulates the key generation. After $\mathcal{B}'''$ received the replies $y^{j,f^i}_{i,\ell}$ of its queries $(f^i(x^{j,0}_i), f^i(x^{j,1}_i))$ for all $i \in \mathcal{EHS} \setminus \{\kappa^\star\}$, all $(x^{j,0}_i, x^{j,1}_i) \in Q_{i,\ell}$ and all $\ell \in QL$, it computes $e^{j,f}_\ell := f(x^{j,0}_1, \ldots, x^{j,0}_n) = f(x^{j,1}_1, \ldots, x^{j,1}_n)$ for all $j \in [Q_\ell]$ and sets $y^{j,f^{\kappa^\star}}_{\kappa^\star,\ell} := e^{j,f}_\ell - (\sum_{i\in[n]\setminus\{\kappa^\star\}} y^{j,f^i}_{i,\ell} + \sum_{i\in[n]\setminus\{\kappa^\star\}} z^{j,f^i}_{i,\ell})$ for all $j \in [Q_\ell]$. $\mathcal{B}'''$ sets $Y_i := \{\{y^{j,f^i}_{i,\ell}\}_{j\in[Q_\ell],\ell\in QL}, \{y'^{j,f^i}_{i,\ell}\}_{j\in[Q'_\ell],\ell\in QL'}\}$. In the final step, $\mathcal{B}'''$ generates $\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, f^i_{\mathcal{Q}_i,Y_i})$ for all $i \in \mathcal{EHS}$ and $\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_i, f^i_{r_i})$ for all $i \in \mathcal{RS}$, sets $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}}\}_{i\in\mathcal{EHS}} \cup \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in\mathcal{RS}}$ and sends $\mathsf{sk}_f$ to $\mathcal{A}$.

For every left-or-right query $(i, x^{j,0}_i, x^{j,1}_i, \ell)$ asked by $\mathcal{A}$, $\mathcal{B}'''$ adds $(x^{j,0}_i, x^{j,1}_i)$ to the list $Q_{i,\ell}$. To generate the final ciphertexts $\mathcal{B}'''$ proceeds different corresponding to the different positions $i$. For the explicitly honest slots $i \in \mathcal{EHS}$ unequal to the $\kappa^\star$'th explicitly honest slot, $\mathcal{B}'''$ queries the underlying one-time pad. In more detail, $\mathcal{B}'''$ queries the underlying one-time pad with $(f^i(x^{j,0}_i) + t'_{i,\ell}, f^i(x^{j,1}_i) + t'_{i,\ell})$ with $t'_{i,\ell} = \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$, for all $(f^i, \cdot) \in Q_{i,f}$. These queries are valid for the same reason as described in the key generation queries. After $\mathcal{B}'''$ received the replies $z^{j,f^i}_{i,\ell}$ of its queries $(f^i(x^{j,0}_i) + t'_{i,\ell}, f^i(x^{j,1}_i) + t'_{i,\ell})$, for all $(f^i, \cdot) \in Q_{i,\ell}$, it sets $Z_i := \{z^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$ and generates $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x^{j,1}_i, Z_i, 0^\lambda, \ell))$ as a reply for $\mathcal{A}$. For a left-or-right query

for the $\kappa^\star$'th explicitly honest slot, $\mathcal{B}'''$ computes $e'^{j,f}_\ell := f(x_1^{j,0},\ldots,x_n^{j,0}) = f(x_1^{j,1},\ldots,x_n^{j,1})$ for all $f \in Q_f$ and all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i\ell}$[9]. Then, $\mathcal{B}'''$ sets $z^{j,f^{\kappa^\star}}_{\kappa^\star,\ell} := e'^{j,f}_\ell - (\sum_{i\in[n]\setminus\{\kappa^\star\}} y^{j,f^i}_{i,\ell} + \sum_{i\in[n]\setminus\{\kappa^\star\}} z^{j,f^i}_{i,\ell})$ for all $f^{\kappa^\star} \in Q_{\kappa^\star,f}$. Afterwards, $\mathcal{B}'''$ sets $Z_{\kappa^\star} := \{z^{j,f^{\kappa^\star}}_{\kappa^\star,\ell}\}_{(f^{\kappa^\star},\cdot)\in Q_{\kappa^\star,f}}$ and generates $\mathsf{ct}^j_{\kappa^\star,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_{\kappa^\star}, (x^{j,1}_{\kappa^\star}, Z_{\kappa^\star}, 0^\lambda, \ell))$ and sends it as a reply to $\mathcal{A}$. For all the remaining slots $i \in \mathcal{RS}$, $\mathcal{B}'''$ computes $\mathsf{ct}^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x_i^{j,0}, \bot, t_{i,\ell}, \ell))$ with $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$ as a reply for $\mathcal{A}$.

The adversary $\mathcal{B}'''$ behaves similar as described above in order to answer an encryption oracle query. In more detail, whenever $\mathcal{A}$ asks an encryption query $(i, x_i^j, \ell)$, $\mathcal{B}'''$ adds $x_i^j$ to the list $Q'_{i,\ell}$. To generate the final ciphertexts $\mathcal{B}'''$ proceeds different corresponding to the different positions $i$. For the explicitly honest slots $i \in \mathcal{EHS}$ unequal to the $\kappa^\star$'th explicitly honest slot, $\mathcal{B}'''$ queries the underlying one-time pad. In more detail, $\mathcal{B}'''$ queries the underlying one-time pad with $(f^i(x_i^j) + t'_{i,\ell}, f^i(x_i^j) + t'_{i,\ell})$ with $t'_{i,\ell} = \mathsf{Gen}(\mathsf{sk}_i, i, \ell)$, for all $(f^i, \cdot) \in Q_{i,f}$. These queries are valid for the same reason as described in the left-or-right queries and under Condition (*) of the security definition. After $\mathcal{B}'''$ received the replies $z'^{j,f^i}_{i,\ell}$ of its queries $(f^i(x_i^j) + t'_{i,\ell}, f^i(x_i^j) + t'_{i,\ell})$, for all $(f^i,\cdot) \in Q_{i,\ell}$, it sets $Z'_i := \{z'^{j,f^i}_{i,\ell}\}_{(f^i,\cdot)\in Q_{i,f}}$ and generates $\mathsf{ct}'^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x_i^j, Z'_i, 0^\lambda, \ell))$ as a reply for $\mathcal{A}$. For an encryption query for the $\kappa^\star$'th explicitly honest slot, $\mathcal{B}'''$ computes $e''^{j,f}_\ell := f(x_1^{j,0},\ldots,x_n^{j,0}) = f(x_1^{j,1},\ldots,x_n^{j,1})$ for all $f \in Q_f$ and all $(x_i^{j,0}, x_i^{j,1}) \in Q_{i\ell}$. Then, $\mathcal{B}'''$ sets $z'^{j,f^{\kappa^\star}}_{\kappa^\star,\ell} := e''^{j,f}_\ell - (\sum_{i\in[n]\setminus\{\kappa^\star\}} y^{j,f^i}_{i,\ell} + \sum_{i\in[n]\setminus\{\kappa^\star\}} z'^{j,f^i}_{i,\ell})$ for all $f^{\kappa^\star} \in Q_{\kappa^\star,f}$. Afterwards, $\mathcal{B}'''$ sets $Z'_{\kappa^\star} := \{z'^{j,f^{\kappa^\star}}_{\kappa^\star,\ell}\}_{(f^{\kappa^\star},\cdot)\in Q_{\kappa^\star,f}}$ and generates $\mathsf{ct}'^j_{\kappa^\star,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_{\kappa^\star}, (x^j_{\kappa^\star}, Z'_{\kappa^\star}, 0^\lambda, \ell))$ and sends it as a reply to $\mathcal{A}$. For all the remaining slots $i \in \mathcal{RS}$, $\mathcal{B}'''$ computes $\mathsf{ct}'^j_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{sk}_i, (x_i^j, \bot, t_{i,\ell}, \ell))$ with $t_{i,\ell} := \mathsf{Gen}''(\mathsf{sk}_i, i, \ell)$ as a reply for $\mathcal{A}$.

This shows the perfect simulation of $\mathsf{G}^\star_{2+\beta}$. Finally, $\mathcal{B}'''$ outputs the same bit $\beta'$ returned by $\mathcal{A}$. Thus, we obtain the lemma. □

# 7 Decentralized Multi-Client Functional Encryption

## 7.1 Definition of Decentralized Multi-Client Functional Encryption

For a decentralized version (i.e. a decentralized setup and functional key generation), we consider the notion of decentralized multi-client functional encryption (DMCFE) as introduced in [CDG+18a]. The work of Chotard et al. [CDG+18a] and all the other works using the notion of decentralized multi-client functional encryption [CDG+18b, ABKW19, ABG19] still consider a centralized setup procedure but mention that this could be fully decentralized by running the setup procedure using an multi-party computation protocol. Now, we define the notion of DMCFE in a fully decentralized way:

**Definition 7.1 (Decentralized Multi-Client Functional Encryption).** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ be a family (indexed by $\lambda$) of sets $\mathcal{F}_\lambda$ of functions $f\colon \mathcal{X}_{\lambda,1} \times \cdots \times \mathcal{X}_{\lambda,n} \to \mathcal{Y}_\lambda$. Let $\mathsf{Labels} = \{0,1\}^*$ or $\{\bot\}$ 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 $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGenShare}, \mathsf{KeyGenComb}, \mathsf{Enc}, \mathsf{Dec})$:*

$\mathsf{Setup} = (\mathcal{P}_1,\ldots,\mathcal{P}_n)$**:** *Is an interactive protocol between $n$ PPT algorithms $\mathcal{P}_1,\ldots,\mathcal{P}_n$, s.t. for all $i \in [n]$ $\mathcal{P}_i$ on input $1^\lambda$ and interacting with $\mathcal{P}_j$ for all $j \in [n]$ with $i \neq j$ obtains the $i$-th secret key $\mathsf{sk}_i$.*

$\mathsf{KeyGenShare}(\mathsf{sk}_i, f)$**:** *Takes a secret key $\mathsf{sk}_i$ from position $i$ and a function $f \in \mathcal{F}_\lambda$, and outputs a partial functional key $\mathsf{sk}_{i,f}$.*

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

$\mathsf{Enc}(\mathsf{sk}_i, x_i, \ell)$ *is defined as for MCFE in Definition 2.3.*

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{1,\ell},\ldots,\mathsf{ct}_{n,\ell})$ *is defined as for MCFE in Definition 2.3.*

[9] where $x_i^{0,j} = x_i^{j,1} = 0$ for all $i \in \mathcal{RS}$

*A scheme* DMCFE *is correct, if for all* $\lambda, n \in \mathbb{N}$, $\{\mathsf{sk}_i\}_{i\in[n]}$ *are the output of* $\mathsf{Setup} = (\mathcal{P}_1, \ldots, \mathcal{P}_n)$ *executed between* $\mathcal{P}_1, \ldots, \mathcal{P}_n$, $f \in \mathcal{F}_\lambda$, $\ell \in \mathsf{Labels}$, $x_i \in \mathcal{X}_{\lambda,i}$, *when* $\mathsf{sk}_{i,f} \leftarrow \mathsf{KeyGenShare}(\mathsf{sk}_i, f)$ *for* $i \in [n]$, *and* $\mathsf{sk}_f \leftarrow \mathsf{KeyGenComb}(\mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f})$, *we have*

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

**Definition 7.2 (Security of DMCFE).** *The* xx-yy-IND *security notion of DMCFE (*$\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$ *with* $\mathrm{yy} \in \{\mathrm{pos}^+, \mathrm{any}\}$*) is similar to the notion of MCFE (Definition 2.4), except that the* Setup *is executed by* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *and the adversary* $\mathcal{A}$ *can corrupt a subset of them, namely* $\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_n}$ *s.t.* $j_i \in \mathcal{CS}$. *Moreover, there is no* msk *and the key generation oracle is now defined as:*

**Key generation oracle** $\mathsf{QKeyG}(f)$**:** *Computes* $\mathsf{sk}_{i,f} \leftarrow \mathsf{KeyGenShare}(\mathsf{sk}_i, f^i)$ *for all* $i \in [n]$ *and outputs* $\{\mathsf{sk}_{i,f}\}_{i\in[n]}$.

### 7.2 Construction of Decentralized Multi-Client Functional Encryption

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

For all $i \in [n]$: $\mathcal{P}_i$ executes the following steps:

$\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$

Run party $P_i$ of $\Pi$ obtaining PRF keys s.t. for all $j \in [n]$ with $j > i$:

$\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K}^{\mathsf{F}}_{i,j} = \mathsf{K}^{\mathsf{F}}_{j,i} \leftarrow \{0,1\}^\lambda$

Set $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$

Return $\mathsf{sk}_i$

$\underline{\mathsf{KeyGenShare}^{\mathsf{mc}}(\mathsf{sk}_i, f^i):}$

Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$

$r_{i,f} := \sum_{j\neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}^{\mathsf{F}}_{i,j}}(f)$

Generate a functional key for $f^i_{r_{i,f}}$ (defined in Fig. 10a Fig. 10b)

$\mathsf{sk}_{f^i} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_{i,f}})$, where the size of $\mathsf{sk}_{f^i_{r_i}}$ is bounded by $q$.

Return $\mathsf{sk}_{f^i}$

$\underline{\mathsf{KeyGenComb}^{\mathsf{mc}}(\mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f}):}$

$\mathsf{sk}_f := \{\mathsf{sk}_{f^i}\}_{i\in[n]}$

Return $\mathsf{sk}_f$

$\underline{\mathsf{Enc}^{\mathsf{mc}}(\mathsf{sk}_i, x_i, \ell):}$

Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$

$t_{i,\ell} := \sum_{j\neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$

$\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i, \bot, t_{i,\ell}, \ell))$, where the size of $\mathsf{ct}^\ell_i$ is bounded by $q$

Return $\mathsf{ct}_{i,\ell}$

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

Compute $\mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{f^i}, \mathsf{ct}_{i,\ell}) = f^i(x_i) + t_{i,\ell} + r_{i,f}$

Return $f(x_1, \ldots, x_n) = \sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_{i,f}$

Fig. 18: The generic construction of $q$-message bounded sel-DMCFE and $q$-message-and-key bounded ad-DMCFE decentralized multi-client functional encryption from single-input functional encryption.

In this section we describe the necessary modifications to turn the presented MCFE of Fig. 9 into a decentralized MCFE scheme (DMCFE). In the decentralized setting, following Section 7.1, the algorithm KeyGenShare is decentralized and non-interactive. Therefore we can not directly use KeyGen as described in Fig. 9 since the $r_i$-values for a certain function $f$ are required to be chosen in such a way that their sum is equal to 0, which requires a simultaneous generation of the functional keys. A way to work around this problem is to generate the $r_i$-values as a PRF output, in the same way as for the encryption procedure Fig. 9. In more detail, for the position $i$ the $r_{i,f}$-value for the function $f$ is defined as $r_{i,f} := \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}_{\mathsf{K}^{\mathsf{F}}_{i,j}}(f)$. The idea of decentralizing a multi-client functional encryption scheme in this way has already been informally described in [ABKW19].

The PRF keys for KeyGenShare and Enc are generated during the setup phase, where the setup is executed between a set of players $\mathcal{P}_1, \ldots, \mathcal{P}_n$, (i.e., $\mathcal{P}_i$ is the $i$-th client of DMCFE scheme). Let $\Pi = (P_1, \ldots, P_n)$ be a $n$-party MPC protocol [Yao86] that securely computes the function $F_{\mathcal{K}}$ which is defined as follows. $F_{\mathcal{K}}$ on input the indexes $1, \ldots n$ outputs for each index $i$ the keys $\{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j \in [n]}$. s.t. $j \in [n]$ with $j > i$: $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K}^{\mathsf{F}}_{i,j} = \mathsf{K}^{\mathsf{F}}_{j,i} \leftarrow \{0,1\}^\lambda$. In the setup phase $\mathcal{P}_i$ executes the player $P_i$ of $\Pi$ thus obtaining keys for the functional keys and for the encryption algorithm.

We formally describe the DMCFE scheme in Fig. 18. Following the approach of Section 6 we also obtain a decentralized MCFE scheme DMCFE that is ad-IND secure with a bounded number of message-and-functional key queries. The adaptively secure scheme DMCFE is also formally described in Fig. 18.

**Correctness.** The correctness of DMCFE follows from the correctness of FE, and the completeness of $\Pi$. Finally we note that $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell})$ outputs the value $\sum_{i \in [n]} f^i(x_i) + t_{i,\ell} + r_{i,f} = \sum_{i \in [n]} f^i(x_i)$, where the last equality follows from the fact that $\sum_{i \in [n]} t_{i,\ell} = 0$ and $\sum_{i \in [n]} r_{i,f} = 0$.

**Theorem 7.3 (sel-pos$^+$-IND security).** *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a $q$-message bounded* sel-FULL*-secure single-input functional encryption scheme for the functionality class $\mathcal{F}_1^{\mathsf{sep}}$,* PRF *an* IND *secure pseudorandom function, and $\Pi$ secure realizes function $F_{\mathcal{K}}$, then* DMCFE *described in Fig. 18 is $q$-message bounded* sel-pos$^+$-IND*-secure for the functionality class $\mathcal{F}_n^{\mathsf{sep}}$.*

*Proof (Sketch).* The security proof proceeds very similar to the one of Theorem 5.1, with the following two differences:

1. We consider an initial game $\mathsf{G}_1^*$ where we switch to the simulator $\mathcal{S}_\Pi$ of $\Pi$ in order to simulate $\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_n}$ s.t. $j_i \in \mathcal{HS}$. The transition from $\mathsf{G}_1^*$ to $\mathsf{G}_1$ follows from the security of $\Pi$.
2. The game $\mathsf{G}_1$ is slightly modified and separated into two games, $\mathsf{G}_1'$ and $\mathsf{G}_1''$. The game $\mathsf{G}_1'$ corresponds to $\mathsf{G}_1$ and in game $\mathsf{G}_1''$ we switch from the pseudorandom values $\mathsf{PRF}_{\mathsf{K}^{\mathsf{F}}_{i_1,i_s}}(f)$ to random values $\mathsf{RF}_s(f)$, for all $s \in \{2, \ldots, h\}$, where the set of honest users is denoted as $\mathcal{HS} := \{i_1, \ldots, i_h\}$, $h \leq n$ as the number of honest users, and RF a random function.
   is described as $\mathsf{G}_1$ but w.r.t. the keys $\mathsf{K}^{\mathsf{F}}_{i_1,i_s}$, where $i_s \in \mathcal{HS}$.

The transition from $\mathsf{G}_1'$ to $\mathsf{G}_1''$ and from $\mathsf{G}_1''$ to $\mathsf{G}_2$ follows from Lemma 5.2 with the observation that all the keys $\mathsf{K}_{j_i,j_k}, \mathsf{K}^{\mathsf{F}}_{j_i,j_k}$ with $j_i, j_k \in \mathcal{HS}$ are not visible to $\mathcal{A}$ since we are executing $\mathcal{S}_\Pi$ for $\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_n}$ with $j_i \in \mathcal{HS}$ in Setup. □

**Theorem 7.4 (ad-pos$^+$-IND security).** *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a $q$-message-and-key bounded* ad-FULL*-secure single-input functional encryption scheme for the functionality class $\mathcal{F}_1^{\mathsf{sep}}$,* PRF *an* IND *secure pseudorandom function and $\Pi$ secure realizes function $F_{\mathcal{K}}$ with security against adaptive corruption, then the* DMCFE *scheme described in Fig. 9 is a $q$-message-and-key bounded* ad-FULL*-secure for the functionality class $\mathcal{F}_n^{\mathsf{sep}}$.*

The security proof proceeds very similar to the one of Theorem 6.1 with the argument described above. Moreover correctness of DMCFE follows from the same arguments as the correctness of DMCFE.

Finally, we note that the compiler of Section 2.3 can be slightly modified in order to obtain the decentralized schemes DMCFE′ and DMCFE′, which are sel-any-IND and ad-any-IND secure. The algorithms

$\mathsf{KeyGenShare}', \mathsf{KeyGenComb}', \mathsf{Enc}', \mathsf{Dec}'$ work as described in Fig. 4, whereas the algorithm $\mathsf{Setup}'$ is defined as an interactive algorithm in order to preserve the decentralized nature of the schemes $\mathsf{DMCFE}'$ and $\mathsf{DMCFE}'$. We notice that the algorithm $\mathsf{Setup}'$ as described in Fig. 4 can be easily decentralized using the same decentralization techniques as for $\mathsf{DMCFE}$ and $\mathsf{DMCFE}$. In particular, let $\Pi = (P_1, \ldots, P_n)$ be a $n$-party MPC protocol [Yao86] that securely computes the function $F_{\mathsf{key}}$, which is defined as follows: On input a index $i$, $F_{\mathsf{key}}$ outputs the keys $\{\mathsf{k}_{i,j}, \mathsf{k}_{i,j}\}_{j\in[n]}$, where $\mathsf{k}_{i,j}, \mathsf{k}_{i,j} \leftarrow \{0,1\}^\lambda$. Formally, $\mathsf{Setup}'$ is defined as follows:

$\mathsf{Setup}' = (P_1, \ldots, P_n)$ :

$\mathsf{sk}_i \leftarrow \mathsf{Setup}(1^\lambda)$, for all $i \in [n]$

For all $i \in [n]$: $\mathcal{P}_i$ executes the following steps:

Run party $P_i$ of $\Pi$ obtaining PRF keys s.t. for all $j \in [n]$ :

$\mathsf{k}_{i,j}$ and $\mathsf{k}_{j,i} \leftarrow \{0,1\}^\lambda$.

$\mathsf{sk}'_i := (\mathsf{sk}_i, \{\mathsf{k}_{i,j}, \mathsf{k}_{j,i}\}_{j\in[n]})$

Return $\{\mathsf{sk}'_i\}_{i\in[n]}$

**Theorem 7.5.** *Let* $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGenShare}, \mathsf{KeyGenComb}, \mathsf{Enc}, \mathsf{Dec})$ *be an* xx-pos$^+$-IND*-secure (key and) message bounded DMCFE scheme for a family of functions* $\mathcal{F}$, $\mathsf{SE} = (\mathsf{Gen}^{\mathsf{SE}}, \mathsf{Enc}^{\mathsf{SE}}, \mathsf{Dec}^{\mathsf{SE}})$ *an* IND-CPA *secure symmetric key encryption scheme,* $\Pi$ *secure realizes function* $F_{\mathsf{key}}$ *(with security against adaptive corruption), then the DMCFE scheme* $\mathsf{DMCFE}' = (\mathsf{Setup}', \mathsf{KeyGenShare}', \mathsf{KeyGenComb}', \mathsf{Enc}', \mathsf{Dec}')$ *described in Section 7.2 is (key and) message bounded* xx-pos$^+$-IND *secure.*

*Proof (Sketch).* The security proof proceeds very similar to the one of Theorem 5.1, but we consider an initial (different) game $\bar{\mathsf{G}}^\star_0$ where we switch to the simulator $\mathcal{S}_\Pi$ of $\Pi$ in order to simulate $\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_n}$ s.t. $j_i \in \mathcal{HS}$. The transition from $\bar{\mathsf{G}}^\star_0$ to $\mathsf{G}^\star_0$ follows immediately from the security of $\Pi$. □

# 8 Outsourceable Multi-Client Functional Encryption

## 8.1 Definition of Outsourceable Multi-Client Functional Encryption

In addition to the definition of (decentralized) multi-client functional encryption, we present another definition called outsourceable multi-client functional encryption (OMCFE). The notion of OMCFE makes it possible to outsource the decryption procedure of the $n$ different ciphertexts to at most $n$ different entities. This notion is especially useful in the case of a very resource consuming decryption procedure. The different ciphertexts $\mathsf{ct}_{i,\ell}$ can be send together with the corresponding partial functional key $\mathsf{sk}_{i,f}$ to the $i$-th entity. The partial decryption procedure applied on $\mathsf{ct}_{i,\ell}$ using $\mathsf{sk}_{i,f}$ generates a decryption share $s_{i,\ell}$. Finally, the shares $s_{i,\ell}$ for every position $i \in [n]$ can be used to reconstruct the final functional output $f(x_1, \ldots, x_n)$. We capture this notion formally.

**Definition 8.1 (Outsourceable Multi-Client Functional Encryption).** *Let* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a collection of function families (indexed by* $\lambda$*), where every* $f \in \mathcal{F}_\lambda$ *is a polynomial time function* $f\colon \mathcal{X}_{\lambda,1} \times \cdots \times \mathcal{X}_{\lambda,n} \to \mathcal{Y}_\lambda$*. Let* $\mathsf{Labels} = \{0,1\}^*$ *or* $\{\bot\}$ *be a set of labels. A outsourceable multi-client functional encryption scheme (OMCFE) for the function family* $\mathcal{F}_\lambda$ *supporting* $n$ *users, is a tuple of four algorithms* $\mathsf{OMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{DecComb})$*:*

$\mathsf{Setup}(1^\lambda, n)$**:** *Takes as input a unary representation of the security parameter* $\lambda$ *and the number of parties* $n$*, and generates* $n$ *secret keys* $\{\mathsf{sk}_i\}_{i\in[n]}$ *and a master secret key* $\mathsf{msk}$*.*

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** *Takes as input the master secret key* $\mathsf{msk}$ *and a function* $f \in \mathcal{F}_\lambda$*, and outputs* $n$ *functional keys* $\mathsf{sk}_{1,f}, \ldots \mathsf{sk}_{n,f}$*.*

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

$\mathsf{PartDec}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell})$: *Takes as input a functional key* $\mathsf{sk}_{i,f}$ *and a ciphertext* $\mathsf{ct}_{i,\ell}$ *and outputs a decryption share* $s_{i,\ell} \in \mathcal{Y}_\lambda$.

$\mathsf{DecComb}(\{s_{i,\ell}\}_{i\in[n]})$ *Takes as input* $n$ *decryption shares* $\{s_{i,\ell}\}_{i\in[n]}$ *under the same label* $\ell$ *and outputs a value* $y \in \mathcal{Y}_\lambda$.

*We require that the computational complexity of* DecComb *is independent from the computational complexity of the function* $f$*, where* $f \in \mathcal{F}_\lambda$.

*A scheme* OMCFE *is correct, if for all* $\lambda, n \in \mathbb{N}$, $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$, $f \in \mathcal{F}_\lambda$, $x_i \in \mathcal{X}_{\lambda,i}$, *when* $\{\mathsf{sk}_{i,f}\}_{i\in[n]} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, *we have*

$$\Pr[\mathsf{DecComb}(\mathsf{PartDec}(\mathsf{sk}_{1,f}, \mathsf{Enc}(\mathsf{sk}_1, x_1, \ell)), \ldots, \mathsf{PartDec}(\mathsf{sk}_{n,f}, \mathsf{Enc}(\mathsf{sk}_n, x_n, \ell))) = f(x_1, \ldots, x_n)] = 1.$$

The security definition for this new notion is the same as for multi-client functional encryption.

We remark that in [FT18] the authors describe a definition of distributed public key FE that has a similar syntax as our definition of OMCFE. Our main goal is to provide a notion of MCFE with an outsourceable decryption procedure, whereas Fan and Tang [FT18] try to construct a public-key functional encryption scheme that achieves a notion of function-hiding. In particular, our definition does not require any privacy w.r.t. the partial functional key.

Respectively, we can also define a decentralized version of OMCFE by decentralizing the key generation procedure and the setup as in Definition 7.1. This adaption is straightforward and we omit it here.

### 8.2 Construction of Outsourceable Multi-Client Functional Encryption

In our $\mathsf{OMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{DecComb})$ the algorithms Setup, KeyGen, Enc corresponds respectively to the one of the MCFE scheme MCFE described in Figure 9. Instead PartDec, DecComb are defined as follow:

$\underline{\mathsf{PartDec}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell}) :}$

Return $s_{i,\ell} = \mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell})$

$\underline{\mathsf{DecComb}(\{s_{i,\ell}\}_{i\in[n]}) :}$

Return $f(x_1, \ldots, x_n) = \sum_{i\in[n]} s_{i,\ell}$.

We observe that the our DecComb satisfies the efficient requirement stated in Definition 8.1, indeed DecComb of the proposed scheme is composed of only addition operations.

**Correctness.** The correctness of OMCFE follows from the correctness of FE, moreover $s_{i,\ell}$ corresponds to $f^i(x_i) + t_{i,\ell} + r_i$ for $i \in [n]$, which in turns implies that $\mathsf{DecComb}(\{s_{i,\ell}\}_{i\in[n]})$ outputs the value $\sum_{i\in[n]} s_{i,\ell} = \sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_i = \sum_{i\in[n]} f^i(x_i)$, where the last equality follows from the fact that $\sum_{i\in[n]} t_{i,\ell} = 0$ and $\sum_{i\in[n]} r_i = 0$.

**Theorem 8.2.** *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a* $q$*-message bounded* sel-FULL*-secure single-input functional encryption scheme for the functionality class* $\mathcal{F}_1^{\mathsf{sep}}$ *and* PRF *an* IND *secure pseudorandom function, then the* OMCFE *scheme described in Section 8.2 is* $q$*-message bounded* ad-pos$^+$-IND*-secure scheme for the functionality class* $\mathcal{F}_n^{\mathsf{sep}}$.

We notice that the proof of Theorem 6.1 can be carried out in a very similar way for Theorem 8.2. In more detail, we can consider the same hybrid experiments as described for Theorem 6.1 with the only difference

that the decryption phase is composed of the algorithms $\mathsf{PartDec}$ and $\mathsf{DecComb}$. Except for this difference, the proof proceeds as the proof for Theorem 6.1.

Following the approach of Section 6 we also obtain an outsourceable MCFE scheme $\mathsf{OMCFE}$ that is ad-pos$^+$-IND-secure with a bounded number of message-and-key queries. In the adaptively secure scheme $\mathsf{OMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{DecComb})$ the algorithms $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}$ correspond to the ones of the MCFE scheme $\mathsf{MCFE}$ as described in Fig. 9, whereas $\mathsf{PartDec}, \mathsf{DecComb}$ are defined specifically defined for the $\mathsf{OMCFE}$ scheme.

**Theorem 8.3.** *Let* $\mathsf{FE} = (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a q-message-and-key bounded* ad-FULL*-secure single-input functional encryecheme for the functionality class* $\mathcal{F}_1^{\mathsf{sep}}$ *and* $\mathsf{PRF}$ *an* IND *secure pseudorandom function, then the* $\mathsf{OMCFE}$ *scheme described in Section 8.2 is q-message-and-key bounded* ad-pos$^+$-IND*-secure scheme for the functionality class* $\mathcal{F}_n^{\mathsf{sep}}$.

The proof proceeds with the same arguments as the proof of Theorem 8.2.
We remark that we achieve sel-pos$^+$-IND and ad-pos$^+$-IND securityfor the schemes $\mathsf{OMCFE}$ and $\mathsf{OMCFE}$ respectively.

**Acknowledgments.** We would like to thank Michel Abdalla for helpful discussions. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 780108 (FENTEC) and by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement 780477 (PRIVILEDGE).

## References

ABG+13. P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://eprint.iacr.org/2013/689. (Pages 3 and 5.)

ABG19. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT 2019, Part III*, *LNCS* 11923, pages 552–582. Springer, Heidelberg, December 2019. (Pages 2, 4, 5, 6, 7, 8, 10, 11, 17, 21, 24, 25, 32, and 35.)

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. (Pages 1, 5, 6, 7, 8, 10, 35, and 37.)

ABSV15. P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 657–677. Springer, Heidelberg, August 2015. (Page 3.)

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. (Pages 1, 5, and 10.)

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. (Pages 5 and 10.)

BCP14. E. Boyle, K.-M. Chung, and R. Pass. On extractability obfuscation. In *TCC 2014*, *LNCS* 8349, pages 52–73. Springer, Heidelberg, February 2014. (Pages 3 and 5.)

BKS16. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *EUROCRYPT 2016, Part II*, *LNCS* 9666, pages 852–880. Springer, Heidelberg, May 2016. (Page 5.)

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. (Page 3.)

BS15. Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. In *TCC 2015, Part II*, *LNCS* 9015, pages 306–324. Springer, Heidelberg, March 2015. (Page 6.)

BS18. Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31(1):202–225, January 2018. (Page 3.)

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. (Pages 1 and 6.)

CDG+18a. 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. (Pages 2, 4, 6, 10, and 35.)

CDG+18b. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. https://eprint.iacr.org/2018/1021. (Pages 1, 2, 6, 8, 10, and 35.)

CVW+18. Y. Chen, V. Vaikuntanathan, B. Waters, H. Wee, and D. Wichs. Traitor-tracing from LWE made simple and attribute-based. In *TCC 2018, Part II*, *LNCS* 11240, pages 341–369. Springer, Heidelberg, November 2018. (Pages 3 and 5.)

DG08. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008. (Page 3.)

FT18. X. Fan and Q. Tang. Making public key functional encryption function private, distributively. In *PKC 2018, Part II*, *LNCS* 10770, pages 218–244. Springer, Heidelberg, March 2018. (Page 39.)

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. (Pages 1, 5, 7, and 8.)

GGH+13. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Pages 3 and 5.)

GGHZ16. S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In *TCC 2016-A, Part II*, *LNCS* 9563, pages 480–511. Springer, Heidelberg, January 2016. (Pages 3 and 5.)

GKP+13. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *45th ACM STOC*, pages 555–564. ACM Press, June 2013. (Pages 3 and 5.)

GKW18. R. Goyal, V. Koppula, and B. Waters. Collusion resistant traitor tracing from learning with errors. In *50th ACM STOC*, pages 660–670. ACM Press, June 2018. (Page 5.)

GVW12. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO 2012*, *LNCS* 7417, pages 162–179. Springer, Heidelberg, August 2012. (Pages 3 and 5.)

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. (Page 4.)

KS17. I. Komargodski and G. Segev. From minicrypt to obfustopia via private-key functional encryption. In *EUROCRYPT 2017, Part I*, *LNCS* 10210, pages 122–151. Springer, Heidelberg, April / May 2017. (Page 5.)

MAS06. D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *25th ACM PODC*, pages 113–122. ACM, July 2006. (Pages 1 and 12.)

MS08. D. Mosk-Aoyama and D. Shah. Fast distributed algorithms for computing separable functions. *IEEE Trans. Information Theory*, 54(7):2997–3007, 2008. (Page 1.)

O'N10. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556. (Pages 1 and 6.)

Sha01. C. E. Shannon. A mathematical theory of communication. *Mobile Computing and Communications Review*, 5(1):3–55, 2001. (Page 13.)

Wat15. B. Waters. A punctured programming approach to adaptively secure functional encryption. In *CRYPTO 2015, Part II*, *LNCS* 9216, pages 678–697. Springer, Heidelberg, August 2015. (Pages 3 and 5.)

Wic15. D. Wichs. Lecture 1: Perfect secrecy and statistical authentication. In *CS 7880 Graduate Cryptography*, 2015. (Page 13.)

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Pages 37 and 38.)