Multi-Client Functional Encryption for Separable Functions

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

[1] The University of Edinburgh, Edinburgh, UK
{michele.ciampi,hendrik.waldner}@ed.ac.uk
[2] Concordium Blockchain Research Center, 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) [10,34] 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* sk_f associated with a function f. Using such a functional key 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 except for $f(x)$ is leaked. In the classical notion of FE, the decryption algorithm takes as input a single ciphertext and a functional key for a single-input (one-variable) function. The more general notion of *Multi-Input Functional Encryption (MIFE)* [25] allows the evaluation of an n-input function on n encrypted inputs. In more detail, the decryption algorithm takes 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)* [25]. In the MCFE setting, each ciphertext $\mathsf{Enc}(\mathsf{sk}_i, x_i)$ is encrypted using a different secret key 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. Intuitively, the notion of MCFE, says that the adversary cannot learn more about 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 it 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* [2, 4]. In the labeled case every ciphertext is encrypted under a label ℓ. 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 [2, 18].

In this work we focus on MCFE for a specific functionality class called *separable functions* [32, 33]. 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 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 a file which is encrypted using a FE scheme under the secret key sk_i. Consider now an entity P_{w} that receives all the encrypted files and wants to count the number of times that the word w occurs in all these files. In addition, P_{w} receives a functional key $\mathsf{sk}_{f_{\mathsf{w}}}$ for the separable function $f_{\mathsf{w}} = f^1_{\mathsf{w}}, \ldots, f^n_{\mathsf{w}}$, where each function f^i_{w} simply counts the number of occurrences of the word w in a file. Given all the encrypted files and $\mathsf{sk}_{f_{\mathsf{w}}}$, P_{w} can compute the number of occurrences of w over all the encrypted files. In addition, even if P_{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, it would be possible to do the computation of sums, counting, and averages over multiple

[3] For example in the worst case, where the adversary has all but the key sk_j, it should be able to compute the number of times that the word w appears in the i-th file, but nothing more than that.

(n) encrypted tables held by different authorities. As already mentioned in [33] separable functions have several applications in sensor and peer-to-peer networks, where different functions are computed over the data of the different sensors (or resp. peers) and only the sum of evaluations should be learned by the decryptor, but nothing about the individual results of the sensors (resp. peers).

Decentralized MCFE. Both, the notions of MIFE and MCFE, assume the existence of a central trusted authority that generates and distributes the secret and functional keys. This is undesirable in some scenarios, 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). To remove the need for a trusted authority, Chotard et al. [17] introduced the notion of *decentralized multi-client functional encryption* (DMCFE), where the generation of the secret keys and the functional keys happens in a decentralized way. In this work, we consider DMCFE for the case of separable functions.

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 secret-key FE scheme and outputs a MCFE scheme for separable functions that is *selectively* secure[4] 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[5] (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 any selective secure secret-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 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 any adaptive secure secret-key FE scheme that supports an a priori bounded number of encryption and functional key queries, then there exists an adaptive secure MCFE scheme for separable functions that supports a bounded number of encryption queries and functional key queries.*

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

[5] 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.

We prove our constructions for the so-called *pos*$^+$ security notion [1, 18]. 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 [1, 18] allows the adversary to ask a left-or-right encryption query on as many positions as it 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 [1] which amplifies any pos$^+$ secure MCFE scheme into 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 secret-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 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 multi-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 sk_f for f. Each of the functional keys $\mathsf{sk}_{f,i}$ can be applied on a ciphertext $\mathsf{ct}_{i,\ell}$ (a ciphertext under label ℓ that contains the i-th input of the function) to obtain a decryption share $\varphi_{i,\ell}$. An evaluator that obtains 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_{w}, would receive the decryption shares from $P_1, \ldots, P_n$, and execute the (light) combine algorithm to obtain the final output of the computation. The word count example can also be seen as a special case of a class of problems that can be parallelized using the *MapReduce* paradigm [21]. 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 this parallelization paradigm.

The security definition of this notion is almost identical to the security definition of MCFE. They mainly differ in their correctness definition (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 secret-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 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 secret-key FE scheme from indistinguishability obfuscation or multilinear maps [13]. We can obtain our adaptive secure MCFE scheme (and the decentralized one) from learning with errors [26], one-way functions or low-depth pseudorandom generators [27]. In more detail, as already mentioned in [13], based on the results of Ananth et al. [8] and Brakerski et al. [15], it is possible to generically obtain a function-hiding scheme by relying on any selectively secure and message-private functional encryption scheme.[6] This implies that function-hiding schemes for any number of encryption and key-generation queries can be based on indistinguishability obfuscation [23, 35], differing-input obfuscation [7, 11], and multilinear maps [24]. Besides this, it is possible to construct function-hiding schemes for a polynomially bounded number, denoted by 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) [26], or on pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with q and with an upper bound on the circuit size of the functions) [27], and based on one-way functions (for $q = 1$) [27].

1.2 Overview of our Techniques

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

In the setup procedure of MCFE we execute n times the setup of FE thus obtaining n master secret keys $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$. We define the i'th secret key for MCFE as $\mathsf{sk}_i := \mathsf{msk}_i$ for $i = 1, \ldots, n$, whereas the master secret key of MCFE

[6] In the informal theorems above we actually require the underlying functional encryption scheme to be function-hiding, but since this property comes for free from any selectively secure and message-private functional encryption scheme, we do not state it specifically.

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 FE using the secret key sk_i and the message x_i thus obtaining the ciphertext 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 msk_i (which corresponds to sk_i) of FE the key generation algorithms for 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 ct_i thus obtaining φ_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, hence it hides the value r_i. The fact that the value r_i is protected allows us to argue that φ_i encrypts the partial output $f^i(x_i)$ (that the adversary is not supposed to see). Indeed, φ_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 modify it as follows: Assuming that we know a polynomial 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 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 (i.e., $r_1 + \cdots + r_n = 0$), 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 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. Note that even if new t-values are generated for each encryption we still need to rely on the r-values hidden inside the function. Otherwise an adversary could use the same ciphertext $\mathsf{ct}_{i,j}$ as the input of multiple functions, which would be the same as reusing a one-time pad key.

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. [1][7], 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 4 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 5.

Decentralized Multi-Client Functional Encryption. In a DMCFE scheme, as introduced in [17], the key-generation phase is decentralized in the sense that each secret key owner should be able to compute 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. Additionally, it is assumed that the setup procedure is a protocol between the different parties that allows for the generation of the different secret keys. This results in a completely decentralized setup that does not require a trusted authority. The MCFE scheme presented above seems to be easily translatable into the decentralized setting but there is an issue: 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 which needs to be computed without interaction between the parties. To do that, we adopt again the technique proposed in [1] to distributively generate a secret sharing of 0. The idea of decentralizing a MCFE scheme in this way has first been proposed in [2].

Outsourceable Multi-Client Functional Encryption. We show how to 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

[7] This technique has previously been used in [16] to remove the central authority in the context of multi-authority attribute based encryption and in [30] in the context of privacy-friendly aggregation.

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

Multi-Input/Client Functional Encryption. Since the introduction of multi-input and multi-client functional encryption [25] several contributions have been made to provide constructions in these areas. In this work we follow the notation of [28], which means that we denote a scheme with a single encryption key that can be used to generate ciphertexts for every position as a MIFE scheme and a scheme where every position is associated with its own encryption key as multi-client functional encryption scheme. One of 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 that presents such a "lifting" in the secret-key setting is the work of Brakerski et al. [12]. In this work, the authors manage to transform a single-input selectively secure functional encryption scheme into an adaptive function-hiding multi-input functional encryption scheme which supports a constant number of inputs. In [29] the authors, among other results, improve the result of [12] 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 an 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 [27]. Their construction relies only on the existence of one-way functions. A second construction in this category has been proposed by Goldwasser et al. [26] and it is based on the Learning with Errors (LWE) assumption.

In the case of unbounded message security most of the known constructions are based on less standard assumptions like indistinguishable obfuscation [9,23,35], multilinear maps [24] and differing-input obfuscation [7,11]. All of the mentioned schemes 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. [5]. The construction they present relies on pairings. A follow up work [4] 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 can be instantiated using DDH, Paillier or LWE. It turns out that the construction of Abdalla et al. [4] also fulfills the

	Number of Inputs	**Functions**	**Setting**	**Assumptions**
[12]	Constant	Generic	MIFE	SK Single-input FE
[29]	$\log(\lambda)^{\delta}$ $0 < \delta < 1$	Generic	MIFE	SK Single-input FE
[4]	poly(λ)	Inner-product	(D)MCFE (no labels)	SK Single-input FE for Inner-product
[1]	poly(λ)	Inner-product	(D)MCFE	PK Single-input FE for Inner-product
This work	poly(λ)	Separable functions	(D)MCFE	SK Single-input FE

Table 1: Comparison with the most relevant compilers. λ: the security parameter, SK: secret key, PK: public key.

stronger notion of multi-client functional encryption (without labels) which has been proven in [2]. 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 [17]; its security is proven based on DDH in the random oracle model. Following, Abdalla et al. [1] and Libert and Titiu [31] show how to construct multi-client functional encryption with labels in the standard model. In more detail, Abdalla et al. [1] present a compiler that lifts a single-input public key functional encryption scheme, which can be instantiated using MDDH, DCR or LWE, into a MCFE scheme with labels. Whereas, Libert and Titiu [31] show how to directly construct a MCFE scheme with labels based on LWE. More recently, Abdalla et al. [3] show how to construct a MCFE scheme with labels in the random oracle model based on MDDH, DCR or LWE, which extends the results of Chotard et al. [17]. In 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.

Decentralization. The notion of DMCFE has been introduced in the work of Chotard et al. [17] in the context of inner product functional encryption. In their work, the authors also present a construction based on the symmetric external Diffie-Hellman assumption in the random oracle model that achieves security in the DMCFE setting. Since then, several compilers for inner product functional encryption have been proposed [1,2,18] that turn a MCFE scheme into a DMCFE scheme. In the works [1, 2] the authors present decentralization compilers that purely rely on information theoretic arguments in the standard model and in the work of Chotard et al. [18] the authors present a compiler based on either the CDH assumption in the random oracle model or the DDH assumption in the standard model. The standard notion of DMCFE [17], with and without labels [2], has the main limitation that it is not possible to let parties join or leave adaptively after the setup procedure has been executed. This problems has been first considered in the work of Agrawal et al. [6], where the authors propose the notion of Ad Hoc Multi-Input Functional Encryption. In this setting every user generates its own public and secret key. Functional key shares are generated with respect to the public keys of other parties. Combining all the functional keys of the specified subset of parties yields the full functional key. This notion allows every party to join the system adaptively and to decide during the key generation which parties' data can be used in the decryption. The authors show how to realize this notion by bootstrapping standard MIFE to ad hoc MIFE without relying on additional assumptions. They also present a direct construction of an ad hoc MIFE for the inner product functionality based on the LWE assumption. In both constructions malicious security is achieved in the common reference string (CRS) model. The high level idea of these constructions is to combine standard MIFE and two-round secure multi-party computation. Another work that considers the above mentioned limitation is the work of Chotard et al. [19]. In their work, the authors introduce the notion of dynamic decentralized MCFE, which generalizes the notion of ad-hoc MIFE. The notion of dynamic DMCFE does not require a specified group of users for the generation of a functional key. Additionally, their notion also considers labels, to prevent certain mix and match attacks and leaks less information about the underlying plaintexts. The authors present a dynamic DMCFE scheme for the inner product functionality from standard assumptions in the random oracle model.

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 $\mathrm{negl} : \mathbb{N} \to \mathbb{R}^+$ *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 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 G and $\mathcal{A}$. We define the distinguishing advantage between games G_0 and 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) [14]. They are an adaption of the notion from [10, 34].

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 λ), 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 λ and generates a master secret key* msk.

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

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

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

A scheme 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 function-hiding 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 (Function-Hiding of SK-FE). *Let* FE *be an SK-FE scheme, $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ a collection of function families indexed by λ. For $\mathrm{xx} \in \{\mathrm{sel}, \mathrm{ad}\}$ and $\beta \in \{0, 1\}$, we define the experiment $\mathrm{xx}\text{-}\mathrm{FH}_\beta^{\mathsf{FE}}$ 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* QKeyG*, and every query $(x^0, x^1) \in Q_{\mathsf{LeftRight}}$, we require that:*

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

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

```
$\text{ad-FH}_{\beta}^{\mathsf{FE}}(\lambda, \mathcal{A})$
$\mathsf{msk} \leftarrow \mathsf{Setup}(1^{\lambda})$
$\alpha \leftarrow \mathcal{A}^{\mathsf{QLeftRight}(\cdot,\cdot),\mathsf{QKeyG}(\cdot,\cdot)}(1^{\lambda})$
Output: $\alpha$ if Condition (*) is
        satisfied, or a uniform
        bit otherwise
```

Fig. 1: Function-Hiding Games for SK-FE

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

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

A secret-key functional encryption scheme FE *is* xx-FH *secure, if for any polynomial-time adversary $\mathcal{A}$, there exists a negligible function* negl *such that:* $\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\text{xx-FH}}(\lambda) \leq \text{negl}(\lambda)$*. 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* $q = \text{poly}(\lambda)$*.*

2.2 Multi-Client Functional Encryption

Now, we introduce multi-client functional encryption (MCFE) as in [1,2,25]. 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 λ), where every $f \in \mathcal{F}_{\lambda}$ is a polynomial time function $f\colon \mathcal{X}_{\lambda,1} \times \cdots \times \mathcal{X}_{\lambda,n} \rightarrow \mathcal{Y}_{\lambda}$. 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* $\mathsf{MCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

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

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

$\mathsf{Enc}(\mathsf{sk}_i, x_i, \ell)$**:** *Takes as input a secret key* 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* sk_f *and* n *ciphertexts under the same label* ℓ *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 $\mathsf{Labels} = \{\perp\}$ 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 of Goldwasser et al. [25] (more specifically [28]), whereas we also allow the adversary to determine under which label it 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 [1, 18]. As also noted in [1, 2] 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 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* λ *and* 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* $\text{sel-yy-IND}_\beta^{\mathsf{MCFE}}$ *in Fig. 2 and* $\text{ad-yy-IND}_\beta^{\mathsf{MCFE}}$ *in Fig. 3, where the oracles are defined as:*

Corruption oracle $\mathsf{QCor}(i)$**:** *Outputs the encryption key* 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, ℓ). *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, ℓ). *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* $\mathsf{QKeyG}(f)$*:*

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

– *When* $\text{yy} = \text{pos}^+$*: 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* $\text{yy} = \text{any}$*: there is no restriction in the left-or-right queries of the adversary.*

```
sel-yy-IND$_\beta^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$
$(\mathcal{CS}, \{Q_{i,\ell}\}_{i \in [n], \ell \in QL}, \{Q'_{i,\ell}\}_{i \in [n], \ell \in QL'}) \leftarrow \mathcal{A}(1^\lambda, n)$
$(\{\mathsf{sk}_i\}_{i \in [n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$
$\mathsf{ct}_{i,\ell}^j \leftarrow \mathsf{QLeftRight}(i, x_i^{j,0}, x_i^{j,1}, \ell)$, 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 $\ell \in QL'$.
$\alpha \leftarrow \mathcal{A}^{\mathsf{QKeyG}(\cdot)}(\{\mathsf{sk}_i\}_{i \in \mathcal{CS}}, \{\mathsf{ct}_{i,\ell}^j\}_{i \in [n], \ell \in QL, j \in [|Q_{i,\ell}|]},$
                              $\{\mathsf{ct}'^j_{i,\ell}\}_{i \in [n], \ell \in QL', j \in [|Q'_{i,\ell}|]})$
Output: $\alpha$ if Condition (*) is satisfied, or a uniform bit
        otherwise
```

Fig. 2: Selective Security Games for MCFE

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

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

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:* $\mathsf{Adv}_{\mathsf{MCFE},\mathcal{A}}^{\text{xx-yy-IND}}(\lambda, n) \leq \text{negl}(\lambda)$*.*

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* $q = \text{poly}(\lambda)$*.*

```
$\text{ad-yy-IND}_\beta^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$
$(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, n)$
$\alpha \leftarrow \mathcal{A}^{\mathsf{QCor}(\cdot),\mathsf{QKeyG}(\cdot),\mathsf{QEnc}(\cdot,\cdot,\cdot),\mathsf{QLeftRight}(\cdot,\cdot,\cdot,\cdot)}(1^\lambda)$
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 $\mathcal{A}$ as a 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-FH and ad-FH. 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. By ad we denote the case in which all three, corruption, left-or-right and encryption queries, are adaptive.

2.3 Separable Functions

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 2.5 (Separable Functions [32]). *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*

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

Functionality Class. 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 $\{f^i\}_{i\in[n]}$.

2.4 Security Compiler, Pseudorandom Functions (PRF), Symmetric Encryption and One-Time Pad Extension

The details of the security compiler presented in Abdalla et al. [1] and its adaption to the bounded case as well as the details on pseudorandom functions and regarding symmetric encryption and the one-time pad extension can be found in the full version [20].

3 Multi-Client Functional Encryption for Separable Functions

In this section, we present our compiler, described in Fig. 4, that turns a single-input functional encryption scheme for class $\mathcal{F}_1^{\mathsf{sep}}$ into a multi-client functional encryption scheme MCFE with labels Labels for the class of separable functions $\mathcal{F}_n^{\mathsf{sep}}$, by relying on a 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 $f \in \mathcal{F}_n^{\mathsf{sep}}$.

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 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 sk_i, a message x_i and a label ℓ. 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 sk_i. This padding is different for every label ℓ 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 ℓ is encrypted using msk_i.

The key generation procedure, takes as inputs the master secret key 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 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

```
$\underline{\mathsf{Setup}^{\mathsf{mc}}(1^\lambda, n):}$
$\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$, for all $i \in [n]$
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 $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{msk})$
$\underline{\mathsf{Enc}^{\mathsf{mc}}(\mathsf{sk}_i, x_i, \ell):}$
Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]})$
$t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$
$\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i, \perp, t_{i,\ell}, \ell))$
Return $\mathsf{ct}_{i,\ell}$
```

```
$\underline{\mathsf{KeyGen}^{\mathsf{mc}}(\mathsf{msk}, \{f^i\}_{i\in[n]}):}$
Parse $\mathsf{msk} := (\{\mathsf{msk}_i\}_{i\in[n]},$
        $\{\mathsf{K}_{i,j}\}_{i,j\in[n],i\neq j})$
For all $i \in [n-1]$, $r_i \leftarrow \mathcal{Y}_\lambda$
$r_n := -\sum_{i\in[n-1]} r_i$
$\mathsf{sk}_{f^i_{r_i}} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_i})$,
with $f^i_{r_i}$ as defined in Fig. 5a Fig. 5b.
Return $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$
$\underline{\mathsf{Dec}^{\mathsf{mc}}(\mathsf{sk}_f, \{\mathsf{ct}_{i,\ell}\}_{i\in[n]}):}$
Parse $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$
$\mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{f^i_{r_i}}, \mathsf{ct}_{i,\ell}) = f^i(x_i) + t_{i,\ell} + r_i$
Return $\sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_i$
```

Fig. 4: 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. We note that "$\perp$" denotes a slot of size q.

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

(a) Selective Security

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

(b) Adaptive Security

Fig. 5: Description of the function that is used for the key generation under the different security definitions.[8]

or a different partial functional key has been used 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 and 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.

4 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 $i \in \mathcal{HS}\backslash i^*$ 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[9] and switch from encryptions of $(x_i^0, t_{i,\ell}, \ell)$ to encryptions of $(x_i^1, 0^\lambda, \ell)$.[10] 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 4.1 (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-FH-*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. 4 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:*

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

Proof. The arguments used for the generation of the values $t_{i,\ell}$ are based on the proof in [1] 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-FH security of the underlying single-input functional encryption scheme FE.

[8] We note that the label ℓ of the plaintext is ignored by the functions and therefore not necessary for the correctness of the construction. However, it is needed in the security proof later.

[9] This encoding results in a functional key size that polynomially depends on the number of challenge and encryption queries. The security of our construction can therefore only been shown if the number of challenge and encryption queries is bounded such that the desired programming is possible.

[10] For our compiler to work, it is required that the underlying single-input functional encryption scheme allows for the desired programmability of the functional keys. Therefore every functional encryption scheme which allows for the desired programming can be used in our compiler and not only functional encryption schemes for a general functionality class, as stated in the formal theorem.

Namely, we build a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}^{\text{sel-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq \mathsf{Adv}^{\text{sel-pos}^+\text{-FH}}_{\mathsf{FE},\mathcal{B}}(\lambda)$. 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 $\{\mathsf{ct}^j_{i^*,\ell}\}_{\ell\in QL,j\in[|Q_{i,\ell}|]}$ 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.e. $\{(i, x_i^j, \ell)\}_{i\in[n],\ell\in QL',j\in|Q'_{i,\ell}|}$, to the encryption oracle QEnc, we distinguish between two different cases. In the case that $\mathcal{A}$ asks for an encryption for all positions $i \neq i^*$, $\mathcal{B}$ 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 QEnc for the position i^*, i.e. it queries (i^*, x^j, ℓ), $\mathcal{B}$ 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 $((i^*, x^j, \ell), (i^*, x^j, \ell))$ 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 $\mathcal{A}$.

This results in the fact that $\mathsf{Adv}^{\text{sel-pos}^+\text{-IND}}_{\mathsf{MCFE},\mathcal{A}}(\lambda, n) \leq \mathsf{Adv}^{\text{sel-FH}}_{\mathsf{FE},\mathcal{B}}(\lambda)$.

For the cases with more than one honest position, we use a hybrid argument with the games defined below. More details on the description of the different games can be found in the full version [20]. Note that G_0 corresponds to the game sel-pos$^+$-IND$^{\mathsf{MCFE}}_0(\lambda, n, \mathcal{A})$, and 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 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 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 RF

denotes a random function (see the full version [20] for more details). The transition from G_0 to G_1 is justified by the security of the PRF. Namely, we exhibit a PPT adversary $\mathcal{B}_0$ such that:

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

Game 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^λ 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. 5a for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i,Y_i}}\}_{i\in[n]}$ (see Fig. 6 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 G_1 to G_2 is achieved using a hybrid argument with a sequence of games $\mathsf{G}_{1.k}$, for $k \in [n]$. It holds that $\mathsf{G}_1 = \mathsf{G}_{1.0}$ and $\mathsf{G}_2 = \mathsf{G}_{1.n}$. This results in

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

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

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

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 G_1 to G_2. Namely, we exhibit a PPT adversary $\mathcal{B}_1$ such that:

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

Game 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 G_2 to 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 the full version [20]. Namely, we show that

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

Game 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^λ values. We also replace the functional key $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{\mathcal{Q}_i, Y_i}}\}_{i\in[n]}$ (see Fig. 6 for the function description) with $\mathsf{sk}_f := \{\mathsf{sk}_{f^i_{r_i}}\}_{i\in[n]}$ (see Fig. 5a for the function description). The transition from G_3 to G_4 is almost symmetric to the transition from G_1 to G_2, justified by the function-hiding security of FE applied on every slot $i \in [n]$. Namely, it can be proven that there exists a PPT adversary $\mathcal{B}_2$ such that:

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

Game G_5: This game is identical to sel-pos$^+$-IND$_1^{\mathsf{MCFE}}(\lambda, n, \mathcal{A})$. The transition from G_4 to G_5 is almost symmetric to the transition from G_0 to G_1, justified by the security of the PRF. Namely, it can be proven that there exists a PPT adversary $\mathcal{B}_3$ such that:

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

Putting everything together, we obtain the theorem. □

The detailed proof of the different game transitions can be found in the full version [20].

5 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

```
$f^i_{\mathcal{Q}_i, Y_i}(x, t_{i,\ell}, \ell)$ :
Parse $\mathcal{Q}_i := \{\{Q_{i,\ell}\}_{\ell\in QL}, \{Q'_{i,\ell}\}_{\ell\in QL'}\}$ and
      $Y_i := \{\{y_{i,\ell}^{j,f^i}\}_{\ell\in QL, j\in[|Q_{i,\ell}|]}, \{y'^{j,f^i}_{i,\ell}\}_{\ell\in QL', j\in[|Q'_{i,\ell}|]}\}$
If $(\cdot, x) \in Q_{i,\ell}$
    Output: $y_{i,\ell}^{j,f^i}$
If $x \in Q'_{i,\ell}$
    Output: $y'^{j,f^i}_{i,\ell}$
```

Fig. 6: Description of the function that is used in the reduction for the selective security reduction.

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 message queries in advance.

We overcome the first problem using a proof technique borrowed from [1]. We define an explicitly honest slots (as in [1]) 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. To solve the second issue, we make use of the $\perp$ slot 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 and 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 of the theorem can be found in the full version [20].

Theorem 5.1 (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-FH-*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. 4 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:*

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

6 Decentralized Multi-Client Functional Encryption

6.1 Definition of Decentralized Multi-Client Functional Encryption

Here, we recap the definition of decentralized multi-client functional encryption (DMCFE) as introduced in [17].

Definition 6.1 (Decentralized Multi-Client Functional Encryption). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ be a family (indexed by λ) 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 $\{\perp\}$ be a set of labels. A decentralized multi-client functional encryption scheme (DMCFE) for the function family $\mathcal{F}$ and the label set Labels is a tuple of six algorithms $\mathsf{DMCFE} = (\mathsf{Setup}, \mathsf{KeyGenShare}, \mathsf{KeyGenComb}, \mathsf{Enc}, \mathsf{Dec})$:

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

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

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

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

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

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

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

Definition 6.2 (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 $\mathcal{P}_1, \dots, \mathcal{P}_n$ and the adversary $\mathcal{A}$ can corrupt a subset of them, namely $\mathcal{P}_{j_1}, \dots, \mathcal{P}_{j_n}$ 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 $\{\mathsf{sk}_{i,f}\}_{i\in[n]}$.*

6.2 Construction of Decentralized Multi-Client Functional Encryption

In this section, we describe the necessary modifications to turn the presented MCFE of Fig. 4 into a decentralized MCFE scheme (DMCFE). In the decentralized setting, following Definition 6.1, the algorithm KeyGenShare is decentralized and non-interactive. Therefore we can not directly use KeyGen, as described in Fig. 4, 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 of the scheme described in Fig. 4. 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 [2].

The PRF keys for KeyGenShare and Enc are generated during the setup phase, where the setup is executed between a set of players $\mathcal{P}_1, \dots, \mathcal{P}_n$, (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 [36] 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 Π thus obtaining keys for the functional keys and for the encryption algorithm.

We formally describe the DMCFE scheme in Fig. 7.

```
$\underline{\mathsf{Setup}^{\mathsf{mc}}(1^\lambda, n):}$
For all $i \in [n]$: $\mathcal{P}_i$ executes the following steps:
    $\mathsf{msk}_i \leftarrow \mathsf{Setup}^{\mathsf{si}}(1^\lambda)$
    Run $P_i$ of $\Pi$ to obtain PRF keys
    for all $j \in [n], j > i$:
    $\mathsf{K}_{i,j} = \mathsf{K}_{j,i} \leftarrow \{0,1\}^\lambda$ and
    $\mathsf{K}^{\mathsf{F}}_{i,j} = \mathsf{K}^{\mathsf{F}}_{j,i} \leftarrow \{0,1\}^\lambda$
$\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$
Return $\mathsf{sk}_i$

$\underline{\mathsf{Enc}^{\mathsf{mc}}(\mathsf{sk}_i, x_i, \ell):}$
Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$
$t_{i,\ell} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}_{i,j}}(\ell)$
$\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}^{\mathsf{si}}(\mathsf{msk}_i, (x_i, \perp, t_{i,\ell}, \ell))$
Return $\mathsf{ct}_{i,\ell}$
```

```
$\underline{\mathsf{KeyGenShare}^{\mathsf{mc}}(\mathsf{sk}_i, f^i):}$
Parse $\mathsf{sk}_i := (\mathsf{msk}_i, \{\mathsf{K}_{i,j}, \mathsf{K}^{\mathsf{F}}_{i,j}\}_{j\in[n]})$
$r_{i,f} := \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}_{\mathsf{K}^{\mathsf{F}}_{i,j}}(f)$
$\mathsf{sk}_{i,f} \leftarrow \mathsf{KeyGen}^{\mathsf{si}}(\mathsf{msk}_i, f^i_{r_{i,f}})$,
with $f^i_{r_{i,f}}$ as defined in
Fig. 5a Fig. 5b
Return $\mathsf{sk}_{i,f}$

$\underline{\mathsf{KeyGenComb}^{\mathsf{mc}}(\mathsf{sk}_{1,f}, \ldots, \mathsf{sk}_{n,f}):}$
$\mathsf{sk}_f := \{\mathsf{sk}_{i,f}\}_{i\in[n]}$
Return $\mathsf{sk}_f$

$\underline{\mathsf{Dec}(\mathsf{sk}_f, \{\mathsf{ct}_i\}_{i\in[n]}):}$
Parse $\mathsf{sk}_f := \{\mathsf{sk}_{i,f}\}_{i\in[n]}$
$\mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell}) = f^i(x_i) + t_{i,\ell} + r_{i,f}$
Return $\sum_{i\in[n]} f^i(x_i) + t_{i,\ell} + r_{i,f}$
```

Fig. 7: 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. We note that "$\perp$" denotes a slot of size q.

Following the approach of Section 5 we also obtain a decentralized MCFE scheme DMCFE that is ad-IND secure with a bounded number of message-and-functional key queries.

Correctness. The correctness of DMCFE follows from the correctness of FE, and the completeness of Π. 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 equality follows from the fact that $\sum_{i\in[n]} t_{i,\ell} = 0$ and $\sum_{i\in[n]} r_{i,f} = 0$. This shows the correctness of the construction.

Theorem 6.3 (sel-pos$^+$-IND security). *Let* FE $= (\mathsf{Setup}^{\mathsf{si}}, \mathsf{KeyGen}^{\mathsf{si}}, \mathsf{Enc}^{\mathsf{si}}, \mathsf{Dec}^{\mathsf{si}})$ *be a q-message bounded* sel-FH-*secure single-input functional encryption* *scheme for the functionality class* $\mathcal{F}_1^{\mathsf{sep}}$*,* PRF *an* IND *secure pseudorandom function, and* Π *secure realizes function* $F_{\mathcal{K}}$*, then* DMCFE *described in Fig. 7 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 4.1, with the following two differences:

1. We consider an initial game G_1^* where we switch to the simulator $\mathcal{S}_\Pi$ of Π in order to simulate $\mathcal{P}_{j_1}, \ldots, \mathcal{P}_{j_n}$ s.t. $j_i \in \mathcal{HS}$. The transition from G_1^* to G_1 follows from the security of Π.
2. The game G_1 is slightly modified and separated into two games, G_1' and G_1''. The game G_1' corresponds to G_1 and in game 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 $\mathcal{HS} := \{i_1, \ldots, i_h\}$, with $h \leq n$ as the number of honest users.

The transition from G_1' to G_1'' and from G_1'' to G_2 follows as in the transition from G_0 to G_1 in Theorem 4.1 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. $\square$

Theorem 6.4 (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-FH*-secure single-input functional encryption scheme for the functionality class* $\mathcal{F}_1^{\mathsf{sep}}$*,* PRF *an* IND *secure pseudorandom function and* Π *secure realizes function* $F_{\mathcal{K}}$ *with security against adaptive corruption, then the* DMCFE *scheme described in Fig. 4 is a* q*-message-and-key bounded* ad-FH*-secure for the functionality class* $\mathcal{F}_n^{\mathsf{sep}}$*.*

The security proof proceeds very similar to the one of Theorem 5.1 with the argument described above. Moreover correctness of DMCFE follows from the same arguments as the correctness of DMCFE. A description on how to lift an pos$^+$ secure DMCFE scheme into an any secure DMCFE scheme can be found in the full version [20]

7 Outsourceable Multi-Client Functional Encryption

7.1 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 sent 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 7.1 (Outsourceable Multi-Client Functional Encryption). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a collection of function families (indexed by λ), 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 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 λ and the number of parties n, and generates n secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$ and a master secret key* msk*.*

$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** *Takes as input the master secret key* 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 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 ℓ and outputs a value $y \in \mathcal{Y}_\lambda$.*

We require that the computational complexity of $\mathsf{DecComb}$ *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*

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

The security definition for this new notion is the same as for multi-client functional encryption (Definition 2.4). We remark that in [22] 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 [22] 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 6.1. This adaption is straightforward and we omit it here.

7.2 Construction of Outsourceable Multi-Client Functional Encryption

In our $\mathsf{OMCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{DecComb})$ scheme the algorithms Setup, KeyGen, and Enc are defined as for the MCFE scheme MCFE described in Fig. 4 and the algorithms $\mathsf{PartDec}$ and $\mathsf{DecComb}$ are defined as follows:

We observe that $\mathsf{DecComb}$ satisfies the efficiency requirement stated in Definition 7.1 since it only consists of a single addition of shares.

```
$\mathsf{PartDec}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell})$ :
Return $s_{i,\ell} = \mathsf{Dec}^{\mathsf{si}}(\mathsf{sk}_{i,f}, \mathsf{ct}_{i,\ell})$
$\mathsf{DecComb}(\{s_{i,\ell}\}_{i\in[n]})$ :
Return $\sum_{i\in[n]} s_{i,\ell}$
```

Fig. 8: Description of PartDec and DecComb

Correctness. The correctness of the OMCFE scheme follows from the correctness of FE. We note that the values $s_{i,\ell}$ correspond 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 equality follows from the fact that $\sum_{i\in[n]} t_{i,\ell} = 0$ and $\sum_{i\in[n]} r_i = 0$. This shows the correctness of the construction.

Theorem 7.2. *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-FH-*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 above 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 5.1 can be carried out in the same way for Theorem 7.2 with the only difference that the decryption phase is composed of the algorithms PartDec and DecComb.

Following the approach of Section 5 we also obtain an outsourceable MCFE scheme OMCFE that is ad-pos$^+$-IND-secure with a bounded number of message-and-key queries. In the adaptively secure scheme OMCFE = (Setup, KeyGen, Enc, PartDec, DecComb) the algorithms Setup, KeyGen, Enc correspond to the ones of the MCFE scheme MCFE as described in Fig. 4, whereas PartDec, DecComb are defined as described in Fig. 8.

Theorem 7.3. *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-FH-*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 above 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 7.2. We remark that we achieve sel-pos$^+$-IND and ad-pos$^+$-IND security for the schemes OMCFE and OMCFE respectively.

Acknowledgments. We 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

1. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_19
2. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_5
3. Abdalla, M., Bourse, F., Marival, H., Pointcheval, D., Soleimanian, A., Waldner, H.: Multi-client inner-product functional encryption in the random-oracle model. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 525–545. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_26
4. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_20
5. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_21
6. Agrawal, S., Clear, M., Frieder, O., Garg, S., O'Neill, A., Thaler, J.: Ad hoc multi-input functional encryption. In: Vidick, T. (ed.) ITCS 2020. vol. 151, pp. 40:1–40:41. LIPIcs (Jan 2020). https://doi.org/10.4230/LIPIcs.ITCS.2020.40
7. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013), `http://eprint.iacr.org/2013/689`
8. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_32
9. Badrinarayanan, S., Gupta, D., Jain, A., Sahai, A.: Multi-input functional encryption for unbounded arity functions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 27–51. Springer, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_2
10. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16
11. Boyle, E., Chung, K.M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (Feb 2014). https://doi.org/10.1007/978-3-642-54242-8_3
12. Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 852–880. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_30
13. Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. Journal of Cryptology **31**(2), 434–520 (Apr 2018). https://doi.org/10.1007/s00145-017-9261-0

14. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46497-7_12
15. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. Journal of Cryptology **31**(1), 202–225 (Jan 2018). https://doi.org/10.1007/s00145-017-9255-y
16. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009. pp. 121–130. ACM Press (Nov 2009). https://doi.org/10.1145/1653662.1653678
17. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_24
18. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018), https://eprint.iacr.org/2018/1021
19. Chotard, J., Dufour-Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Dynamic decentralized functional encryption. Cryptology ePrint Archive, Report 2020/197 (2020), https://eprint.iacr.org/2020/197
20. Ciampi, M., Siniscalchi, L., Waldner, H.: Multi-client functional encryption for separable functions. Cryptology ePrint Archive, Report 2020/219 (2020), https://eprint.iacr.org/2020/219
21. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (Jan 2008). https://doi.org/10.1145/1327452.1327492, https://doi.org/10.1145/1327452.1327492
22. Fan, X., Tang, Q.: Making public key functional encryption function private, distributively. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 218–244. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_8
23. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.13
24. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49099-0_18
25. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_32
26. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488678
27. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.)

CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_11
28. Gordon, S.D., Katz, J., Liu, F.H., Shi, E., Zhou, H.S.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774 (2013), http://eprint.iacr.org/2013/774
29. Komargodski, I., Segev, G.: From minicrypt to obfustopia via private-key functional encryption. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 122–151. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_5
30. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (Jul 2011). https://doi.org/10.1007/978-3-642-22263-4_10
31. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_18
32. Mosk-Aoyama, D., Shah, D.: Computing separable functions via gossip. In: Ruppert, E., Malkhi, D. (eds.) 25th ACM PODC. pp. 113–122. ACM (Jul 2006). https://doi.org/10.1145/1146381.1146401
33. Mosk-Aoyama, D., Shah, D.: Fast distributed algorithms for computing separable functions. IEEE Trans. Information Theory **54**(7), 2997–3007 (2008). https://doi.org/10.1109/TIT.2008.924648, https://doi.org/10.1109/TIT.2008.924648
34. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), http://eprint.iacr.org/2010/556
35. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_33
36. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25