Multi-Client Functional Encryption for Linear Functions in the Standard Model from LWE

Benoît Libert[1,2] and Radu Ţiţiu[2,3]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[3] Bitdefender, Bucharest, Romania

Abstract. Multi-client functional encryption (MCFE) allows ℓ clients to encrypt ciphertexts $(\mathbf{C}_{t,1}, \mathbf{C}_{t,2}, \ldots, \mathbf{C}_{t,\ell})$ under some label. Each client can encrypt his own data X_i for a label t using a private encryption key ek_i issued by a trusted authority in such a way that, as long as all $\mathbf{C}_{t,i}$ share the same label t, an evaluator endowed with a functional key dk_f can evaluate $f(X_1, X_2, \ldots, X_\ell)$ without learning anything else on the underlying plaintexts X_i. Functional decryption keys can be derived by the central authority using the master secret key. Under the Decision Diffie-Hellman assumption, Chotard *et al.* (Asiacrypt 2018) recently described an adaptively secure MCFE scheme for the evaluation of linear functions over the integers. They also gave a decentralized variant (DMCFE) of their scheme which does not rely on a centralized authority, but rather allows encryptors to issue functional secret keys in a distributed manner. While efficient, their constructions both rely on random oracles in their security analysis. In this paper, we build a standard-model MCFE scheme for the same functionality and prove it fully secure under adaptive corruptions. Our proof relies on the Learning-With-Errors (LWE) assumption and does not require the random oracle model. We also provide a decentralized variant of our scheme, which we prove secure in the static corruption setting (but for adaptively chosen messages) under the LWE assumption.

Keywords. Multi-client functional Encryption, inner product evaluation, LWE, standard model, decentralization.

1 Introduction

Functional encryption (FE) [62,19] is a modern paradigm that overcomes the all-or-nothing nature of ordinary encryption schemes. In FE, the master secret key msk allows deriving a sub-key dk_f associated with a specific function f. If a ciphertext C encrypts a message X under the master public key mpk, when dk_f is used to decrypt C, the decryptor only obtains $f(X)$ and nothing else about X. Functional encryption is an extremely general concept as it subsumes identity-based encryption [17,29], searchable encryption [16], attribute-based encryption [62,44], broadcast encryption [32] and many others.

As formalized by Boneh, Sahai and Waters [19], FE only allows evaluating a function f over data provided by a single sender whereas many natural applications require to compute over data coming from distinct distrustful parties. A straightforward solution to handle multiple senders is to distribute the generation of ciphertexts by means of a multi-party computation (MPC) protocol. Unfortunately, jointly generating a ciphertext incurs potentially costly interactions between the senders who should be online at the same time and have their data ready to be submitted. Ideally, the participants should be able to supply their input without interacting with one another and go off-line immediately after having sent their contribution. This motivates the concepts of multi-input [38,37] and multi-client [43,37] functional encryption, which support the evaluation of multivariate functions over data coming from distinct sources.

1.1 (Decentralized) Multi-Client FE

Multi-client functional encryption. As defined in [43,37], multi-client functional encryption (MCFE) allows computing over input vectors $(X_1, \ldots, X_\ell)$ of which each coordinate X_i may be sent by a different client. Each ciphertext C_i is associated with a client index i and a tag t (also called "label"): on input of a vector of ciphertexts $(C_1 = \mathsf{Encrypt}(1, X_1, t), \ldots, C_\ell = \mathsf{Encrypt}(\ell, X_\ell, t))$, where C_i is generated by client i using a secret encryption key ek_i for each $i \in [\ell]$, anyone holding a functional decryption key dk_f for an ℓ-ary function can compute $f(X_1, \ldots, X_\ell)$ as long as all C_i are labeled with the same tag t (which may be a time-specific information or a dataset name). No further information than $f(X_1, \ldots, X_\ell)$ is revealed about individual inputs X_i and nothing can be inferred by combining ciphertexts generated for different tags. MCFE can thus be seen as a multi-party computation (MPC) where each ciphertext C_i can be generated independently of others and no communication is needed between data providers.

Decentralized multi-client functional encryption. Most FE flavors involve a single central authority that should not only be trusted by all users, but also receives the burden of generating all functional secret keys. In decentralized FE systems [52,24], multiple authorities can operate independently without even being aware of one another.

Like its single-client counterpart, multi-client FE requires a trusted entity, which is assigned the task of generating a master key msk as well as handing out encryption keys ek_i to all clients and functional decryption keys dk_f to all decryptors. In some applications, clients may be reluctant to rely on a single point of trust. This motivates the design of a decentralized version of MCFE, as introduced by Chotard *et al.* [27]. Decentralized multi-client functional encryption (DMCFE) obviates the need for a centralized authority by shifting the task of generating functional secret keys to the clients themselves. In a setup phase, the clients $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ first generate public parameters by running an interactive protocol but no further interaction is needed among clients when it comes to generating functional secret keys later on. When a decryptor wishes to obtain a functional secret key for an ℓ-ary function f, it interacts with each client i independently so as to obtain partial functional decryption keys $\mathsf{dk}_{f,i}$. The decryptor can then fold $\{\mathsf{dk}_{f,i}\}_{i=1}^{\ell}$ into a functional decryption key dk_f for f. By doing so, each client has full control over his individual data and the functions for which secret keys are given out. Importantly, no interaction among senders is required beyond the setup phase, where public parameters are generated.

As a motivating example, Chotard *et al.* [27] consider the use-case of a financial analyst that is interested in mining several companies' private data so as to better understand the dynamics of an economical sector. These companies have some incentives to collaborate, but they do not want their clients' data to be abused (in which case, they would risk heavy fines owing to the EU General Data Protection Regulation). After having interactively set up DMCFE parameters, each company can encrypt its own data with respect to a time-stamp. Then, the analyst can contact each company to obtain partial functional keys and reconstruct a key that only reveals a weighted aggregate of companies' private inputs provided they are labeled with the same time-stamp.

Chotard *et al.* [27] described a DMCFE scheme that allows evaluating linear functions over encrypted data: namely, if $(X_1, \ldots, X_\ell) \in \mathbb{Z}^\ell$ are the individual contributions sent by ℓ senders, a functional secret key dk_f for the integer vector $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}^\ell$ allows computing $\sum_{i=1}^{\ell} y_i \cdot X_i$ from $\{C_i = \mathsf{Encrypt}(i, X_i, t)\}_{i=1}^{\ell}$, where C_i is generated by the i-th sender. In the decentralized setting, each sender can also generate a partial functional secret key $\mathsf{dk}_{f,i}$ for $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}^\ell$ using their secret encryption key ek_i.

1.2 Our Contributions

The MCFE scheme of Chotard *et al.* [27] was proved fully secure (as opposed to selectively secure) in the random oracle model under the standard Decision Diffie-Hellman assumption in groups without a bilinear maps. Its decentralized variant was proved secure under the Symmetric eXternal Diffie-Hellman (SXDH) assumption in groups endowed with an asymmetric bilinear map. While efficient, the schemes of [27] both require the random oracle model. Chotard *et al.* thus left open the problem of designing a (D)MCFE system under well-studied hardness assumptions without using random oracles: even in the centralized setting, the only known MCFE candidates in the standard model [43,37] rely on indistinguishability obfuscation. They also left open the problem of instantiating their schemes under the LWE assumption or any other assumption than DDH.

In this paper, we address both problems. For linear functions over the integers (i.e., the same functionality as [27]), we construct the first MCFE scheme in the standard model and prove it fully secure under the Learning-With-Errors assumption [60] in the adaptive corruption setting (note that only static corruptions were considered in [43, Section 2.3]). This construction turns out to be the first standard-model realization of an MCFE system with labels – albeit for a restricted functionality – that does not require obfuscation. Next, we extend our centralized system to obtain the first labeled DMCFE scheme without random oracles. Like [27], our decentralized solution is only proved secure in the static corruption setting although we can handle adaptive corruptions in its centralized version. Both constructions are proved secure under the LWE assumption with sub-exponential approximation factors. Our security proofs stand in the standard model in the sense of the same security definitions as those considered in [27].

We leave it as an open problem to achieve security under an LWE assumption with polynomial approximation factor. Another natural open question is the feasibility of (D)MCFE beyond linear functions under standard assumptions.

1.3 Challenges and Techniques

We start from the observation that the DDH-based MCFE scheme of Chotard *et al.* [27] can be interpreted as relying on (a variant of) the key-homomorphic pseudorandom function [18] of Naor, Pinkas and Reingold [58]. Namely, the scheme of [27] encrypts $x_i \in \mathbb{Z}_q$ for the tag t by computing $C_i = g^{x_i} \cdot H_{t,1}^{s_i} \cdot H_{t,2}^{t_i}$, where $(s_i, t_i) \in \mathbb{Z}_q^2$ is the i-th sender's secret key and $(H_{t,1}, H_{t,2}) = H(t) \in \mathbb{G}^2$ is derived from a random oracle in a DDH-hard group $\mathbb{G} = \langle g \rangle$.

The security proof of [27] crucially exploits the entropy of the secret key (s_i, t_i) in a hybrid argument over all encryption queries. To preserve this entropy, they need to prevent the encryption oracle from leaking too much about uncorrupted users' secret keys $\{(s_i, t_i)\}_i$. For this purpose, they rely on the DDH assumption to modify the random oracle $H : \{0,1\}^* \to \mathbb{G}^2$ in such a way that, in all encryption queries but one, the hash value $H(t) \in \mathbb{G}^2$ lives in a one-dimensional subspace. In order to transpose this technique in the standard model, we would need a programmable hash function [46] that ranges over a one-dimensional subspace of $\mathbb{G}^2$ on polynomially-many inputs while mapping an extra input outside this subspace with noticeable probability. The results of Hanaoka *et al.* [45] hint that such programmable hash functions are hardly instantiable in prime-order DDH groups. While the multi-linear setting [33] allows bypassing the impossibility results of [45], it is not known to enable standard assumptions.

A natural idea is to replace the random-oracle-based key-homomorphic PRF of [58] by an LWE-based key-homomorphic PRF [18,11]. However, analogously to Chotard *et al.* [27],[4] we aim at an MCFE system that can be proved secure in a game where the adversary is allowed to corrupt senders adaptively. In order to deal with the adaptive corruption of senders, we thus turn to the adaptively secure distributed PRF proposed by Libert, Stehlé and Titiu [55]. The latter can be seen as instantiating the programmable hash function of Freire *et al.* [33] in the context of homomorphic encryption (FHE). Their PRF maps an input x to $\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p$, where[5] $\mathbf{s} \in \mathbb{Z}^n$ is the secret key and $\mathbf{A}(x) \in \mathbb{Z}_q^{n \times m}$ is derived from public matrices using the Gentry-Sahai-Waters FHE [36]. More precisely, the matrix $\mathbf{A}(x)$ is obtained as the product of GSW ciphertexts dictated by the output of an admissible hash function [15] applied to the PRF input. The security proof of [55] uses the property that, with noticeable probability, the input-dependent matrix $\mathbf{A}(x)$ is a GSW encryption of 1 for the challenge input $x^\star$: namely, $\mathbf{A}(x^\star)$ is a matrix of of the form $\mathbf{A}(x^\star) = \mathbf{A} \cdot \mathbf{R}^\star + \mathbf{G}$, where $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is

[4] While their decentralized scheme is only proved secure under static corruptions, its centralized version is proved secure under adaptive corruptions.

[5] Introduced in [12], the notation $\lfloor x \rfloor_p$ stands for the rounded value $\lfloor (p/q) \cdot x \rfloor \in \mathbb{Z}_p$, where $x \in \mathbb{Z}_q$, and $p < q$.

the gadget matrix of Micciancio and Peikert [57] and $\mathbf{R}^\star \in \mathbb{Z}^{m \times m}$ is a small-norm matrix. At the same time, all evaluation queries are associated with a matrix $\mathbf{A}(x)$ consisting of a GSW encryption of 0 (i.e., a matrix $\mathbf{A}(x) = \mathbf{A} \cdot \mathbf{R}$, for a small-norm $\mathbf{R} \in \mathbb{Z}^{m \times m}$). Then, the proof of [55] appeals to the lossy mode of LWE [39] and replaces the uniform matrix $\mathbf{A}^\top \in \mathbb{Z}_q^{m \times n}$ by a lossy matrix of the form $\hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}$, where $\mathbf{E} \in \mathbb{Z}^{m \times n}$ is a short integer matrix with Gaussian entries, $\mathbf{C} \in \mathbb{Z}_q^{n_1 \times n}$ is random, and $\hat{\mathbf{A}} \in \mathbb{Z}_q^{n_1 \times m}$ has rank $n_1 \ll n$. In all evaluation queries, the smallness of $\mathbf{s} \in \mathbb{Z}^n$ then ensures that the values $\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p$ always reveal the same information about $\mathbf{s}$, which amounts to the product $\mathbf{C} \cdot \mathbf{s} \in \mathbb{Z}_q^{n_1}$. Since $\mathbf{A}(x^\star)$ depends on $\mathbf{G}$ for the challenge input $x^\star$, the function $\lfloor \mathbf{A}(x^\star)^\top \cdot \mathbf{s} \rfloor_p$ is in fact an injective function of $\mathbf{s}$, meaning that it has high min-entropy.

Our MCFE scheme relies on the lossy mode of LWE in a similar way to [55], except that we add a Gaussian noise instead of using the Learning-With-Rounding technique [12]. The i-th sender uses his secret key $\mathbf{s}_i \in \mathbb{Z}^n$ to encrypt a short integer vector as $\boldsymbol{x}_i \in \mathbb{Z}^{n_0}$ as $\mathbf{C}_i = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(t)^\top \cdot \mathbf{s}_i + \mathsf{noise} \in \mathbb{Z}_q^m$, where $\mathbf{A}(t) \in \mathbb{Z}_q^{n \times m}$ is a tag-dependent matrix derived as a product of GSW ciphertexts indexed by the bits of t and $\mathbf{G}_0 \in \mathbb{Z}_q^{n_0 \times m}$ is a gadget matrix for which the lattice $\Lambda^\perp(\mathbf{G}_0)$ has a short public basis. A functional secret key for the vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top$ consists of $\mathsf{dk}_{\boldsymbol{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ and allows computing $\mathbf{G}_0^\top \cdot (\sum_{i=1} y_i \cdot \boldsymbol{x}_i) + \mathsf{small} \in \mathbb{Z}_q^m$ from $\sum_{i=1}^{\ell} y_i \cdot \mathbf{C}_i \in \mathbb{Z}_q^m$ and eventually recovering the linear function $\sum_{i=1} y_i \cdot \boldsymbol{x}_i \in \mathbb{Z}^{n_0}$ of $\mathbf{X} = [\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_\ell] \in \mathbb{Z}_q^{n_0 \times \ell}$.

At this point, adapting the security proof of [55] is non-trivial. We cannot rely on the DPRF of [55] in a modular way as it would require a DPRF where partial evaluations are themselves pseudorandom so long as the adversary does not obtain the underlying secret key shares: in our setting, a challenge ciphertext contains a bunch of partial evaluations (one for each message slot) rather than a threshold recombination of such evaluations. We emphasize that, in the LWE-based DPRF of [55], partial evaluations are not proven pseudorandom: [55] only proves – via a deterministic randomness extraction argument – the pseudorandomness of the final PRF value obtained by combining partial evaluations. They cannot apply (and neither can we) a randomness extractor to individual partial DPRF evaluations as it would destroy their key homomorphic property. Instead of relying on the pseudorandomness of partial evaluations, we actually prove a milder indistinguishability property which suffices for our purposes.

The first step is to make sure that all encryption queries will involve a lossy matrix $\mathbf{A}(t)^\top = \mathbf{R}_t \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}_t$, for small-norm $\mathbf{R}_t \in \mathbb{Z}^{m \times m}$ and $\mathbf{E}_t \in \mathbb{Z}^{m \times n}$, so that honest senders' ciphertexts are of the form $\mathbf{C}_i = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{R}_t \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} \cdot \mathbf{s}_i + \mathsf{noise}$ and thus leak nothing about $\mathbf{s}_i \in \mathbb{Z}^n$ beyond $\mathbf{C} \cdot \mathbf{s}_i \in \mathbb{Z}_q^{n_1}$. The difficulty arises in the challenge queries $(i, t^\star, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star)$, where $\mathbf{A}(t^\star) \in \mathbb{Z}_q^{n \times m}$ is not a lossy matrix and we must find a way to replace $\mathbf{C}_i^\star = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{0,i}^\star + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}_i + \mathsf{noise}$ by $\mathbf{C}_i^\star = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{1,i}^\star + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}_i + \mathsf{noise}$ without the adversary noticing. In [55], the proof relies on a deterministic randomness extraction[6] argument to extract statistically uniform bits from $\lfloor \mathbf{A}(x^\star)^\top \cdot \mathbf{s} \rfloor_p$, which has high min-entropy when $\mathbf{A}(x^\star)$ is of the form $\mathbf{A} \cdot \mathbf{R}^\star + \mathbf{G}$. Here, we do not see how to apply deterministic extractors in the proof while preserving the functionality of the MCFE scheme.

Our solution is to program the public parameters in such a way that, with noticeable probability, the challenge ciphertexts are generated for a matrix $\mathbf{A}(t^\star) \in \mathbb{Z}_q^{n \times m}$ of the form

$$\mathbf{A}(t^\star)^\top = \mathbf{R}^\star \cdot \mathbf{A}^\top + \mathbf{G}_0^\top \cdot \mathbf{V} = \mathbf{R}^\star \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{G}_0^\top \cdot \mathbf{V} + \mathsf{noise}, \tag{1}$$

for a statistically random matrix $\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n}$ included in the public parameters. In the proof, the simulator generates a statistically uniform matrix $\mathbf{U} = \left[\begin{smallmatrix}\mathbf{V}\\ \mathbf{C}\end{smallmatrix}\right]$, where $\mathbf{C} \in \mathbb{Z}_q^{n_1 \times n}$ is used to build the lossy matrix $\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}$, together with a trapdoor $\mathbf{T}_\mathbf{U}$ for $\Lambda^\perp(\mathbf{U})$. (The idea of embedding a trapdoor in the LWE secret of a lossy matrix is borrowed from [54]). Using $\mathbf{T}_\mathbf{U}$, the simulator can sample a short matrix $\mathbf{T} \in \mathbb{Z}^{n \times n_0}$ satisfying $\mathbf{U} \cdot \mathbf{T} = \left[\begin{smallmatrix}\mathbf{I}_{n_0}\\ \mathbf{0}\end{smallmatrix}\right] \bmod q$, allowing it to define an alternative secret key $\mathbf{s}_i' = \mathbf{s}_i + \mathbf{T} \cdot (\boldsymbol{x}_{0,i}^\star - \boldsymbol{x}_{1,i}^\star) \in \mathbb{Z}^n$. As long as $\mathbf{s}_i$ is sampled from a Gaussian distribution with sufficiently large standard deviation, $\mathbf{s}_i'$ and $\mathbf{s}_i$ are negligibly far apart in terms of statistical distance (note that, as in [67,13], the simulator can guess $\boldsymbol{x}_{0,i}^\star - \boldsymbol{x}_{1,i}^\star$ upfront without affecting the polynomial running time of the reduction since we are in the middle of a purely statistical argument). The alternative secret keys $\{\mathbf{s}_i'\}_{i=1}^{\ell}$ further satisfy $\sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i' = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i$ for all legal functional key queries $\boldsymbol{y} = (y_1, \ldots, y_\ell)$ made by the adversary. The definition of $\mathbf{s}_i'$ finally ensures that $\mathbf{C} \cdot \mathbf{s}_i' = \mathbf{C} \cdot \mathbf{s}_i \bmod q$, meaning that $\mathbf{s}_i'$ is compatible with all encryption queries for which $\mathbf{A}(t)$ is lossy. From (1), the condition $\mathbf{V} \cdot \mathbf{T} = \mathbf{I}_{n_0} \bmod q$ then implies that the challenge ciphertext can be interpreted as an encryption of $\boldsymbol{x}_{1,i}^\star$ since $\mathbf{C}_i^\star = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{1,i}^\star + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}_i' + \mathsf{noise}$ is statistically close to $\mathbf{C}_i^\star = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{0,i}^\star + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}_i + \mathsf{noise}$.

We insist that our construction and proof are not merely obtained by plugging the DPRF of [55] into the high-level design principle of [27]. In particular, we do not rely on the pseudorandomness of partial PRF evaluations, but rather prove a milder indistinguishability property in some transition in our sequence of games. To do this, we need to modify the proof of [55], by introducing a matrix $\mathbf{V}$ and embedding a trapdoor in the matrix $\mathbf{U}$ obtained by stacking up $\mathbf{V}$ and the secret matrix $\mathbf{C}$ of the lossy mode of LWE.

In order to build a DMCFE system, we proceed analogously to [27] and combine two instances of our centralized MCFE scheme. The first one is only used to generate partial functional secret keys whereas the second one is used exactly as in the centralized system. As in [27], we first have the senders run an interactive protocol allowing them to jointly generate public parameters for the two MCFE instances. At the end of this protocol (which may involve costly MPC operations, but is only executed once), each sender holds an encryption key

[6] The standard Leftover Hash Lemma cannot be applied since the source $\lfloor \mathbf{A}(x^\star)^\top \cdot \mathbf{s} \rfloor_p$ is not guaranteed to be independent of the seed. A deterministic extractor based on k-wise independent functions [31] is thus needed in [55].

$\mathsf{ek}_i = (\mathbf{s}_i, \mathbf{t}_i)$ consisting of encryption keys for the two underlying instances. In order to have the i-th sender $\mathcal{S}_i$ generate a partial functional secret key $\mathsf{dk}_{f,i}$ for a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top$, we exploit the fact that our centralized scheme allows encrypting vectors. Namely, the decryptor obtains from $\mathcal{S}_i$ an MCFE encryption of the vector $y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ under the encryption key $\mathbf{t}_i$ of the first instance.

1.4 Related Work

Functional encryption was implicitly introduced by Sahai and Waters in [62], where they also constructed a scheme for threshold functions. Constructions of FE for point functions (known as identity-based encryption) [17,29] existed already, but were not viewed through the lens of FE until later. Subsequent works saw constructions for several more advanced functionalities such as inner product functions [50,7], Boolean formulas [44,51,59,65,53], membership checking [20] and even finite state automaton [66]. Recently, the landscape of functional encryption improved considerably. Gorbunov *et al.* [42] and Garg *et al.* [34] provided the first constructions of attribute-based encryption for all circuits; Goldwasser *et al.* [41] constructed succinct simulation-secure single-key FE scheme for all circuits and also obtained FE for Turing machines [40]. In a breakthrough result, Garg *et al.* [34] designed indistinguishability-secure multi-key FE schemes for all circuits. However, while the constructions of [42,41] rely on standard assumptions, the assumptions underlying the other constructions [34,40] are still ill-understood and have not undergone much cryptanalytic effort.

FE for simple circuits. Abdalla, Bourse, De Caro and Pointcheval [3] considered the question of building FE for linear functions (a functionality dubbed IPFE for "inner product functional encryption"). Here, a ciphertext C encrypts a vector $\boldsymbol{y} \in \mathcal{D}^\ell$ over some ring $\mathcal{D}$, a secret key for the vector $\boldsymbol{x} \in \mathcal{D}^\ell$ allows computing $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and nothing else about $\boldsymbol{y}$. Abdalla *et al.* [3] described two constructions under the Decision Diffie-Hellman (DDH) and Learning-With-Errors (LWE) assumptions, respectively. On the downside, Abdalla *et al.* [3] only proved their schemes to be secure against selective adversaries. Namely, in the security game, the adversary chooses two vectors $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathcal{D}^\ell$ and expects to receive an encryption of one of these in the challenge phase. Selective security forces the adversary to declare $\boldsymbol{x}_0, \boldsymbol{x}_1$ before seeing the public key and before obtaining any private key. Agrawal, Libert and Stehlé subsequently upgraded the constructions of [3] so as to prove security against adaptive adversaries, which may choose $\boldsymbol{x}_0, \boldsymbol{x}_1$ after having seen the public key and obtained a number of private keys. Agrawal *et al.* [8] described several IPFE schemes under well-established assumptions which include the standard Decision Diffie-Hellman (DDH) assumption, the Decision Composite Residuosity (DCR) assumption and the LWE assumption. Under the DCR and LWE assumptions, the schemes of [8] can evaluate both inner products over the integers and modulo a prime or composite number. The IPFE constructions of [3,8] served as building blocks for FE schemes handling general functionalities [9] in the bounded collusion setting [61,42]. Quite recently, the IPFE functionality [3,8] was extended into FE schemes supporting the evaluation of quadratic functions over encrypted data [56,10]. The schemes of [56,10] are only proved secure against selective adversaries and they can only compute functions which have their output confined in a small interval. For the time being, the only known FE schemes that support the evaluation of more general functions than quadratic polynomials either require fancy tools like obfuscation [34] , or are restricted to bounded collusions [42,9].

Multi-Input and Multi-Client Functional Encryption. Goldwasser *et al.* [38,37] introduced the concept of multi-input functional encryption (MIFE). MIFE and MCFE are both more interesting in the secret-key setting than in the public-key setting, where much more information inevitably leaks about the data (see, e.g., [38,5,27]). Similarly to MCFE, MIFE operates over input vectors $(X_1, \ldots, X_\ell)$ comprised of messages sent by distinct parties, but without assigning a tag to ciphertexts: each user i can encrypt X_i as $C_i = \mathsf{Encrypt}(X_i)$ in such a way that anyone equipped with a functional secret key dk_f for an ℓ-argument function f can compute $f(X_1, \ldots, X_n)$ given multiple ciphertexts $\{C_i = \mathsf{Encrypt}(X_i)\}_{i=1}^{\ell}$. Brakerski *et al.* [22] gave a transformation for constructing adaptively secure general-purpose MIFE schemes for a constant n from any general-purpose private-key single-input scheme. Like MCFE, MIFE for general functionalities necessarily rely on indistinguishability obfuscation or multilinear maps, so that instantiations under standard assumptions are currently lacking. Under the SXDH assumption, Abdalla *et al.* [5] managed to construct a MIFE scheme for the inner product functionality. In their scheme, each input slot encrypts a vector $\boldsymbol{x}_i \in \mathbb{Z}_p^m$ while each functional secret key $\mathsf{sk}_{\boldsymbol{y}}$ corresponds to a vector $\boldsymbol{y} \in \mathbb{Z}_p^{\ell \cdot m}$, where ℓ is the total number of slots. On input of encrypted data $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell)$ such that $\boldsymbol{x}_i$ is encrypted by sender i in the i-th slot, their multi-input inner product functionality computes $\langle \boldsymbol{X}, \boldsymbol{y} \rangle$ using $\mathsf{sk}_{\boldsymbol{y}}$. Function-hiding MIFE schemes were described in [30,4]. Abdalla *et al.* [4] notably gave a generic single-input to multi-input transformation, which yields MIFE constructions for the inner product functionality under the DDH, LWE and DCR assumptions.

Besides syntactical differences, MCFE departs from MIFE in the amount of information leaked about plaintexts. The MIFE model [38,37] allows any slot of any ciphertext to be combined with any other slot of any other ciphertext. As soon as senders encrypt more than one ciphertext per slot, a given functional secret key can thus compute a much larger number of values. As discussed in [27], this feature incurs a much more important information leakage, especially when many functional secret keys are given out. In contrast, the multi-client setting only allows functional secret keys to operate over ciphertexts that share the same tag. As long as tags are single-use (e.g., a timestamp), this allows clients to retain a more accurate control over the information leaked about their data.

The first MCFE realization was proposed in [43,37] and relies on the DDH assumption and on indistinguishability obfuscation to handle general circuits. The notion of aggregator-oblivious encryption (AOE) [64,23,48,14] allows an untrusted aggregator to compute sums of encrypted values without learning anything else about individual inputs. As such, AOE can be seen as a form of MCFE with single-key security (namely, the only key revealed to the aggregator is for the vector $(1,1,\ldots,1)^\top$) for the evaluation of inner products. So far, all non-interactive AOE constructions [64,48,14] rely on the random oracle model.

The first efficient MCFE scheme with multi-key security was described by Chotard *et al.* [27] who also introduced the concept of decentralized MCFE. Their schemes both rely on DDH-like assumptions in the random oracle model. At the time of writing, we are not aware of any (D)MCFE construction based on a well-studied assumption in the standard model.

Decentralized Functional Encryption. The first examples of decentralized FE schemes were given in the context of attribute-based encryption (ABE) [25,26]. Lewko and Waters [52] gave the first ciphertext-policy ABE where users' attributes may be certified by completely independent authorities. Boneh and Zhandry [21] suggested distributed broadcast encryption systems, which dispense with the need for an authority handing out keys to registered users. Chandran *et al.* [24] considered decentralized general-purpose FE using obfuscation. The decentralization of multi-client FE was first considered by Chotard *et al.* [27] in a model where all clients run an interactive protocol to generate public parameters, but eliminate any interaction beyond the setup phase.

Abdalla *et al.* [2] described generic transformations providing DMCFE schemes from any MCFE system satisfying extra properties. While applying their compilers to [4] yields DMCFE schemes in the standard model, the resulting ciphertexts are not labeled. Without labels, the functionality leaks much more information about encrypted messages for a given functional key since there is no restriction on the way slots from different ciphertexts can be combined together (any slot from any ciphertext can be combined with any other slot from any other ciphertext). In this paper, our goal is to support labels, which is significantly more challenging and was only achieved in the random oracle model so far.

Chotard *et al.* [28] gave a technique to remove the restriction that forces the adversary to make challenge queries for all uncorrupted ciphertext slots. Their technique upgrades any MCFE scheme satisfying our definition (which is the definition introduced in [27] and called "pos-IND" security in [2]) so as to prove security under a stronger definition where the adversary can obtain incomplete ciphertexts. Their technique builds on a "secret-sharing layer" (SSL) primitive which is only known to exist assuming pairings and random oracles as their SSL scheme [28, Section 4.2] is implicitly based on the Boneh-Franklin IBE [17]. Abdalla *et al.* [2] suggested a different technique to handle incomplete ciphertexts without using pairings, but they either require random oracles or they do not support labels (except in a model with static corruptions and selective security).

Chotard *et al.* [28] also showed how to transform the ROM-based scheme from [27] in such a way that users are allowed multiple encryption queries for each slot-label pair. Their technique is not generic and only works for their DDH-based construction (as they mention in Section 6.2). Finally, [2,28] both give generic compilers from MCFE to DMCFE. Abdalla *et al.* [2] obtain DMCFE under adaptive corruptions, but they need to start from an MCFE which computes inner products modulo an integer L (instead of inner products over $\mathbb{Z}$). Hence, their compiler does not imply DMCFE from LWE in the standard model. As it turns out, neither [2,28] implies MCFE with labels in the standard model from LWE (nor any standard assumption), even for the security definition of [27]. In a concurrent and independent work [1], Abdalla *et al.* provide a solution to this problem via a generic construction of labeled MCFE from single-input IPFE schemes evaluating modular inner products. While their construction satisfies a stronger security notion than ours (which allows multiple encryption queries for the same slot-label pair), their scheme of [1, Section 3] requires longer ciphertext than ours as each slot takes a full IPFE ciphertext of linear size in ℓ if ℓ is the number of slots.

In their construction and in ours, handling incomplete ciphertexts expands partial ciphertexts by a factor $O(\ell)$. In our most efficient schemes, we still need to assume that the adversary obtains challenge ciphertexts for all clients as in [27]. In the full version of the paper, we show that a variant of the compiler of Abdalla *et al.* [2] allows proving security in the standard model, even when the adversary is allowed to obtain incomplete challenge ciphertexts. Our compiler relies on pseudorandom functions satisfying a specific security definition in the multi-instance setting. The concurrent work of Abdalla *et al.* [1] achieves a similar result using any PRF satisfying a standard security definition.

2 Background

2.1 Lattices

For any $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. For a vector $\mathbf{x} \in \mathbb{R}^n$ denote $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots x_n^2}$ and $\|\mathbf{x}\|_\infty = \max_i |x_i|$. If $\mathbf{M}$ is a matrix over $\mathbb{R}$, then $\|\mathbf{M}\| := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|}$ and $\|\mathbf{M}\|_\infty := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{M}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty}$. For a finite set S, we let $U(S)$ denote the uniform distribution over S. If X and Y are distributions over the same domain, then $\Delta(X, Y)$ denotes their statistical distance. Let $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ be a symmetric positive-definite matrix, and $\mathbf{c} \in \mathbb{R}^n$. We define the Gaussian function on $\mathbb{R}^n$ by $\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-\pi(\mathbf{x} - \mathbf{c})^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mathbf{c}))$ and if $\mathbf{\Sigma} = \sigma^2 \cdot \mathbf{I}_n$ and $\mathbf{c} = \mathbf{0}$ we denote it by ρ_σ. For an n dimensional lattice $\Lambda \subset \mathbb{R}^n$ and for any lattice vector $\mathbf{x} \in \Lambda$ the discrete gaussian is defined $\rho_{\Lambda,\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\mathbf{\Sigma},\mathbf{c}}}{\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda)}$. For an n-dimensional lattice Λ, we define $\eta_\varepsilon(\Lambda)$ as the smallest $r > 0$ such that $\rho_{1/r}(\widehat{\Lambda} \setminus \mathbf{0}) \leq \varepsilon$ with $\widehat{\Lambda}$ denoting the dual of Λ, for any $\varepsilon \in (0, 1)$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define $\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q\}$ and $\Lambda(\mathbf{A}) = \mathbf{A}^\top \cdot \mathbb{Z}^n + q\mathbb{Z}^m$. For an arbitrary vector $\mathbf{u} \in \mathbb{Z}_q^n$, we also define the shifted lattice $\Lambda^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q\}$.

Definition 2.1 (LWE). *Let $m \geq n \geq 1$, $q \geq 2$ and $\alpha \in (0, 1)$ be functions of a security parameter λ. The LWE problem consists in distinguishing between the distributions $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$ and $U(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$, where $\mathbf{A} \sim U(\mathbb{Z}_q^{m \times n})$, $\mathbf{s} \sim U(\mathbb{Z}_q^n)$ and $\mathbf{e} \sim D_{\mathbb{Z}^m, \alpha q}$. For an algorithm $\mathcal{A} : \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \to \{0, 1\}$, we define:*

$$\mathbf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathcal{A}) = |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1| ,$$

where the probabilities are over $\mathbf{A} \sim U(\mathbb{Z}_q^{m \times n})$*,* $\mathbf{s} \sim U(\mathbb{Z}_q^n)$*,* $\mathbf{u} \sim U(\mathbb{Z}_q^m)$ *and* $\mathbf{e} \sim D_{\mathbb{Z}^m,\alpha q}$ *and the internal randomness of* $\mathcal{A}$*. We say that* $\mathsf{LWE}_{q,m,n,\alpha}$ *is hard if, for any* ppt *algorithm* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathcal{A})$ *is negligible.*

Micciancio and Peikert [57] described a trapdoor mechanism for LWE. Their technique uses a "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$, with $w = n \log q$, for which anyone can publicly sample short vectors $\mathbf{x} \in \mathbb{Z}^w$ such that $\mathbf{G} \cdot \mathbf{x} = \mathbf{0}$.

Lemma 2.2 ([57, Section 5]). *Let* $m \geq 3n \log q$*. There exists a* ppt *algorithm* GenTrap *that outputs a statistically uniform matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$*, together with a trapdoor* $\mathbf{T_A} \in \mathbb{Z}^{m \times m}$ *for* $\Lambda^{\perp}(\mathbf{A})$*, such that* $\max_j \|\tilde{\mathbf{t}}_j\| \leq O(\sqrt{n \log q})$*, where* $\tilde{\mathbf{t}}_j$ *are the corresponding Gram-Schmidt vectors.*

It is known [57] that, for any $\mathbf{u} \in \mathbb{Z}_q^n$, a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ allows sampling from $D_{\Lambda^{\boldsymbol{u}}(\mathbf{A}), s \cdot \omega\left(\sqrt{\log m}\right)}$ for $s = O(\sqrt{n \log q})$. Since

$$\eta_{2^{-m}}\left(\Lambda^{\perp}(\mathbf{A})\right) \leq \max_j \|\tilde{\boldsymbol{t}}_j\| \cdot \omega(\sqrt{\log m}) \leq s \cdot \omega(\sqrt{\log m})$$

for large enough $s = O(\sqrt{n \log q})$, the magnitude of a vector $\mathbf{x}$ sampled from $D_{\Lambda^{\boldsymbol{u}}(\mathbf{A}), s \cdot \omega\left(\sqrt{\log m}\right)}$, is bounded by $\|\mathbf{x}\| \leq s\sqrt{m} \cdot \omega(\sqrt{\log m})$.

Remark 2.3. For $m \geq 3n \log q$, we can thus sample a statistically uniform matrix $\mathbf{A}$ from $\mathbb{Z}_q^{n \times m}$ together with a trapdoor, which allows finding small solutions of $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q$, with $\|\mathbf{x}\| \leq s\sqrt{m} \cdot \omega(\sqrt{\log m}) = O(\sqrt{mn \log q}) \cdot \omega(\sqrt{\log m})$.

We sometimes rely on the so-called "noise flooding" technique via the next lemma.

Lemma 2.4 ([39, Lemma 3]). *Let* $\boldsymbol{y} \in \mathbb{Z}^m$*. The statistical distance between* $D_{\mathbb{Z}^m,\sigma}$ *and* $\boldsymbol{y} + D_{\mathbb{Z}^m,\sigma}$ *is at most* $\Delta\left(D_{\mathbb{Z}^m,\sigma}, \boldsymbol{y} + D_{\mathbb{Z}^m,\sigma}\right) \leq m \cdot \frac{\|\boldsymbol{y}\|_\infty}{\sigma}$.

Lemma 2.5 ([35, Theorem 4.1]). *There is a* ppt *algorithm that, given a basis* $\mathbf{B}$ *of an* n*-dimensional lattice* $\Lambda = \mathcal{L}(\mathbf{B})$*, a parameter* $s > \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$*, and a center* $\mathbf{c} \in \mathbb{R}^n$*, outputs a sample from a distribution statistically close to* $D_{\Lambda,s,\mathbf{c}}$*.*

2.2 Admissible Hash Functions

Admissible hash functions were introduced by Boneh and Boyen [15] as a combinatorial tool for partitioning-based security proofs for which Freire *et al.* [33] gave a simplified definition. Jager [47] considered the following generalization in order to simplify the analysis of reductions under decisional assumption.

Definition 2.6 ([47]). *Let* $\ell(\lambda), L(\lambda) \in \mathbb{N}$ *be functions of a security parameter* $\lambda \in \mathbb{N}$*. Let* $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$ *be an efficiently computable function. For every* $K \in \{0,1,\perp\}^L$*, let the partitioning function* $P_K : \{0,1\}^\ell \to \{0,1\}$ *such that*

$$P_K(X) := \begin{cases} 0 & \textit{if} \quad \forall i \in [L] \quad (\mathsf{AHF}(X)_i = K_i) \ \vee \ (K_i = \perp) \\ 1 & \textit{otherwise} \end{cases}$$

We say that AHF *is a* **balanced admissible hash function** *if there exists an efficient algorithm* $\mathsf{AdmSmp}(1^\lambda, Q, \delta)$ *that takes as input* $Q \in \mathsf{poly}(\lambda)$ *and a non-negligible* $\delta(\lambda) \in (0,1]$ *and outputs a key* $K \in \{0,1,\perp\}^L$ *such that, for all* $X^{(1)}, \ldots, X^{(Q)}, X^\star \in \{0,1\}^\ell$ *such that* $X^\star \notin \{X^{(1)}, \ldots, X^{(Q)}\}$*, we have*

$$\gamma_{\max}(\lambda) \geq \Pr_K \left[P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0 \right] \geq \gamma_{\min}(\lambda),$$

where $\gamma_{\max}(\lambda)$ *and* $\gamma_{\min}(\lambda)$ *are functions such that*

$$\tau(\lambda) = \gamma_{\min}(\lambda) \cdot \delta(\lambda) - \frac{\gamma_{\max}(\lambda) - \gamma_{\min}(\lambda)}{2}$$

is a non-negligible function of λ*.*

Intuitively, the condition that $\tau(\lambda)$ be non-negligible requires $\gamma_{\min}(\lambda)$ to be noticeable and the difference of $\gamma_{\max}(\lambda) - \gamma_{\min}(\lambda)$ to be small.

It is known [47] that balanced admissible hash functions exist for $\ell, L = \Theta(\lambda)$.

Theorem 2.7 ([47, Theorem 1]). *Let* $(C_\ell)_{\ell \in \mathbb{N}}$ *be a family of codes* $C_\ell : \{0,1\}^\ell \to \{0,1\}^L$ *with minimal distance* $c \cdot L$ *for some constant* $c \in (0, 1/2)$*. Then,* $(C_\ell)_{\ell \in \mathbb{N}}$ *is a family of balanced admissible hash functions. Furthermore,* $\mathsf{AdmSmp}(1^\lambda, Q, \delta)$ *outputs a key* $K \in \{0,1,\perp\}^L$ *for which* $\eta = \lfloor \frac{\ln(2Q + Q/\delta)}{-\ln((1-c))} \rfloor$ *components are not* $\perp$ *and* $\gamma_{\max} = 2^{-\eta}$*,* $\gamma_{\min} = \big(1 - Q(1-c)\big)^\eta \cdot 2^{-\eta}$*, so that* $\tau = (2\delta - (2\delta + 1) \cdot Q \cdot (1-c)^\eta)/2^{\eta+1}$ *is a non-negligible function of* λ*.*

Lemma 2.8 ([49, Lemma 8],[6, Lemma 28]). *Let* $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q, \delta)$*, an input space* $\mathcal{X}$ *and the mapping* γ *that maps a* $(Q+1)$*-uple* $(X^\star, X_1, \ldots, X_Q)$ *in* $\mathcal{X}^{Q+1}$ *to a probability value in* $[0,1]$*, given by:*

$$\gamma(X^\star, X_1, \ldots, X_Q) := \Pr_K \left[P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0 \right].$$

We consider the following experiment where we first execute the PRF security game, in which the adversary eventually outputs a guess $\hat{b} \in \{0,1\}$ *of the challenger's bit* $b \in \{0,1\}$ *and wins with advantage* ε*. We denote by* $X^\star \in \mathcal{X}$ *the challenge input and* $X_1, \ldots, X_Q \in \mathcal{X}$ *the evaluation queries. At the end of the game, we flip a fair random coin* $b'' \hookleftarrow U(\{0,1\})$*. If the condition* $P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \wedge P_K(X^\star) = 0$ *is satisfied we define* $b' = \hat{b}$*. Otherwise, we define* $b' = b''$*. Then, we have* $|\Pr[b' = b] - 1/2| \geq \gamma_{\min} \cdot \varepsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2}$*, where* $\gamma_{\min}$ *and* $\gamma_{\max}$ *are the maximum and minimum of* $\gamma(\mathbb{X})$ *for any* $\mathbb{X} \in \mathcal{X}^{Q+1}$*.*

2.3 Randomness Extraction

The Leftover Hash Lemma was used by Agrawal *et al.* [6] to re-randomize matrices over $\mathbb{Z}_q$ by multiplying them with small-norm matrices.

Lemma 2.9 ([6]). *Let integers* m, n *such that* $m > 2n \cdot \log q$*, for some prime* $q > 2$*. Let* $\mathbf{A}, \mathbf{U} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$ *and* $\mathbf{R} \hookleftarrow U(\{-1,1\}^{m \times m})$*. The distributions* $(\mathbf{A}, \mathbf{AR})$ *and* $(\mathbf{A}, \mathbf{U})$ *are within* $2^{-\Omega(n)}$ *statistical distance.*

2.4 Multi-Client Functional Encryption

We recall the syntax of multi-client functional encryption as introduced in [43].

Definition 2.10. *A* **multi-client functional encryption** *(MCFE) scheme for a message space $\mathcal{M}$ and tag space $\mathcal{T}$ is a tuple* $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{DKeygen}, \mathsf{Decrypt})$ *of efficient algorithm with the following specifications:*

$\textsf{\textbf{Setup}}(\mathsf{cp}, 1^\ell)$: *Takes in global parameters* cp *and a pre-determined number of users* 1^ℓ*, where* cp *specifies a security parameter* 1^λ*. It outputs a set of public parameters* mpk*, a master secret key* msk*, and a set of encryption keys* $\{\mathsf{ek}_i\}_{i=1}^\ell$*. We assume that* mpk *is included in all encryption keys* ek_i*.*

$\textsf{\textbf{Encrypt}}(\mathsf{ek}_i, x_i, t)$: *Takes as input the encryption key* ek_i *of user* $i \in [\ell]$*, a message* x_i *and a tag* $t \in \mathcal{T}$*. It output a ciphertext* $C_{t,i}$*.*

$\textsf{\textbf{DKeygen}}(\mathsf{msk}, f)$: *Takes as input the master secret key* msk *and an* ℓ*-argument function* $f : \mathcal{M}^\ell \to \mathcal{R}$*. It outputs a functional decryption key* dk_f*.*

$\textsf{\textbf{Decrypt}}(\mathsf{dk}_f, t, \mathbf{C})$: *Takes as input a functional decryption key* dk_f*, a tag* t*, and an* ℓ*-vector of ciphertexts* $\mathbf{C} = (C_{t,1}, \ldots, C_{t,\ell})$*. It outputs a function evaluation* $f(\boldsymbol{x}) \in \mathcal{R}$ *or an error message* $\perp$*.*

Correctness. For any set of public parameters cp, any $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^\ell) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$, any vector $\boldsymbol{x} \in \mathcal{M}^n$ any tag $t \in \mathcal{T}$ and any function $f : \mathcal{M}^\ell \to \mathcal{R}$, if $C_{t,i} \leftarrow \mathsf{Encrypt}(\mathsf{ek}_i, x_i, t)$ for all $i \in [\ell]$ and $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$, we have $\mathsf{Decrypt}\big(\mathsf{dk}_f, t, \mathbf{C}_t = (C_{t,1}, \ldots, C_{t,\ell})\big) = f(\boldsymbol{x})$ with overwhelming probability.

We now recall the security definition given in [43] for an adaptively secure MCFE, and then we will give the definition that we use in this work. These two definitions are in fact equivalent.

Definition 2.11 (IND-sec)**.** *For an MCFE scheme with* ℓ *senders, consider the following game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*. The game involves a set* $\mathcal{HS}$ *of honest senders (initialized to* $\mathcal{HS} := [\ell]$*) and a set* $\mathcal{CS}$ *(initialized to* $\mathcal{CS} := \emptyset$*) of corrupted senders.*

Initialization: *The challenger* $\mathcal{C}$ *chooses* cp *and runs* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek_i}\}_{i=1}^\ell) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$*. Then, it chooses a random bit* $b \leftarrow \{0, 1\}$ *and gives the master public key* mpk *to the adversary*

Encryption queries: *The adversary* $\mathcal{A}$ *can adaptively make encryption queries* $\mathsf{QEncrypt}(i, x^0, x^1, t)$*, to which the challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x^b, t)$*. For any given pair* (i, t)*, only one query is allowed and subsequent queries involving the same* (i, t) *are ignored.*

Functional decryption key queries: *The adversary can adaptively obtain functional decryption keys by making queries of the form* $\mathsf{QDKeygen}(f)$*. The challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$*.*

Corruption queries: *For any user* $i \in \mathcal{HS}$*, the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$*, to which the challenger replies with* ek_i *and updates* $\mathcal{HS}$ *and* $\mathcal{CS}$ *by setting* $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ *and* $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$*.*

Finalize: *The adversary makes its guess* $b' \in \{0, 1\}$*;* $\mathcal{A}$ *wins the game if* $\beta = b$*, where* β *is defined to be* $\beta := b'$ *except in the following situations.*

1. *An encryption query* $\mathsf{QEncrypt}(i, x^0, x^1, t)$ *has been made for an index* $i \in \mathcal{CS}$ *with* $x^0 \neq x^1$*.*
2. *For some label* t*, an encryption query* $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *has been asked for* $i \in \mathcal{HS}$*, but encryption queries* $\mathsf{QEncrypt}(j, x_j^0, x_j^1, t)$ *have not been asked for all* $j \in \mathcal{HS}$*.*
3. *For a label* t *and some function* f *queried to* $\mathsf{QDKeygen}$*, there exists a pair of vectors* $(\boldsymbol{x}^0, \boldsymbol{x}^1)$ *such that* $f(\boldsymbol{x}^0) \neq f(\boldsymbol{x}^1)$*, where*
   - $x_i^0 = x_i^1$ *for all* $i \in \mathcal{CS}$*;*
   - $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *have been asked for all* $i \in \mathcal{HS}$*.*

In any of the above cases, $\mathcal{A}$*'s output is replaced by a random* $\beta \leftarrow U(\{0, 1\})$*.*

An MCFE scheme provides IND *security if, for any efficient adversary* $\mathcal{A}$*, we have* $\mathbf{Adv}^{\mathsf{IND}}(\mathcal{A}) := |\Pr[\beta = 1 \mid b = 1] - \Pr[\beta = 1 \mid b = 0]| \in \mathsf{negl}(\lambda)$*.*

In the following, it will be convenient to work with the following security definition, which is equivalent to Definition 2.11.

Definition 2.12 **(1-challenge** IND**-sec).** *For an MCFE scheme with* ℓ *senders, we consider the following game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*. The game involves a set* $\mathcal{HS}$ *(initialized to* $\mathcal{HS} := [\ell]$*), of honest senders and a set* $\mathcal{CS}$ *(initialized to* $\mathcal{CS} := \emptyset$*), of corrupted senders.*

Initialization: *The challenger* $\mathcal{C}$ *generates* cp *and runs* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^{\ell})$*. Then, it chooses a random bit* $b \leftarrow \{0, 1\}$ *and gives the master public key* mpk *to the adversary* $\mathcal{A}$*.*

Encryption queries: *The adversary can adaptively make encryption queries* $\mathsf{QEncrypt}(i, x, t)$*, to which the challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x, t)$*. Any further query involving the same pair* (i, t) *is ignored.*

Challenge queries: *The adversary adaptively makes challenge queries of the form* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$*. The challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^{\star b}, t^{\star})$*. Only one tag* $t^{\star}$ *can be involved in a challenge query. If* $t^{\star}$ *denotes the tag of the first query, the challenger only replies to subsequent challenge queries for the same label* $t^{\star}$*. Moreover, only one query* $(i, t^{\star})$ *is allowed for each* $i \in [\ell]$ *and subsequent queries involving the same* $i \in [\ell]$ *are ignored.*

Functional decryption key queries: *The adversary can adaptively obtain functional decryption keys via queries* $\mathsf{QDKeygen}(f)$*. At each query, the challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$*.*

Corruption queries: *For any user* $i \in \mathcal{HS}$*, the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$*, to which the challenger replies with* ek_i *and updates* $\mathcal{HS}$ *and* $\mathcal{CS}$ *by setting* $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ *and* $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$*.*

Finalize: *The adversary outputs a bit* $b' \in \{0, 1\}$*. The adversary* $\mathcal{A}$ *wins if* $\beta = b$*, where* β *is defined as* $\beta := b'$*, unless of the situations below occurred.*

1. *A challenge query* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$ *has been made for an index* $i \in \mathcal{CS}$ *with* $x_i^{\star 0} \neq x_i^{\star 1}$*.*
2. *An encryption query* $\mathsf{QEncrypt}(i, x, t^{\star})$ *has been made for the challenge tag* $t^{\star}$ *for some index* $i \in [\ell]$*.*

3. *For the challenge tag $t^\star$, a challenge query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ has been asked for some $i \in \mathcal{HS}$, but challenge queries $\mathsf{CQEncrypt}(j, x_j^{\star 0}, x_j^{\star 1}, t^\star)$ have not been asked for all $j \in \mathcal{HS}$.*
4. *For the challenge tag $t^\star$ and some function f queried to $\mathsf{QDKeygen}$, there exists a pair of vectors $(\boldsymbol{x}^{\star 0}, \boldsymbol{x}^{\star 1})$ such that $f(\boldsymbol{x}^{\star 0}) \neq f(\boldsymbol{x}^{\star 1})$, where*
   - $x_i^{\star 0} = x_i^{\star 1}$ *for all* $i \in \mathcal{CS}$*;*
   - $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ *have been asked for all* $i \in \mathcal{HS}$*.*

If any of these events occurred, $\mathcal{A}$'s output is overwritten by $\beta \leftarrow U(\{0,1\})$.

We say that an MCFE scheme provides 1Ch-IND *security if, for any efficient adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\mathsf{1Ch\text{-}IND}}(\mathcal{A}) := \left|\Pr[\beta = b] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda)$.*

In the full version of the paper, we show that 1Ch-IND security implies IND security. We also note that condition 2 of "Finalize" could be:

2'. Both $\mathsf{QEncrypt}(i, x, t^\star)$ and $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ have been made for an index i and the challenge label $t^\star$, such that $x_i^{0\star} \neq x_i^{1\star}$

This allows the adversary to make both an encryption query $\mathsf{QEncrypt}(i, x, t^\star)$ and a challenge query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ where $x_i^{\star 0} = x_i^{\star 1}$. In the full version of the paper, we show that replacing condition 2 by condition $2'$ does not make the adversary any stronger.

Our first construction is proven secure under Definition 2.12. Abdalla *et al.* [2] and Chotard *et al.* [28] independently showed constructions that can be proven secure in the sense of a stronger definition which eliminates restriction 3 from the "Finalize" stage. In the full version of the paper, we show that a variant of the compiler of [2, Section 4.2] is secure in the standard model. Recently, Abdalla *et al.* [1] independently obtained a similar result. While their PRF-based compiler [1] can rely on any PRF, we obtain a tighter reduction using a specific PRF described in [55]. Chotard *et al.* [28] additionally show how to enable repetitions by allowing multiple encryption queries for the same pair (i, t). However, they need random oracles for this purpose.

2.5 Decentralized Multi-Client Functional Encryption

We use the same syntax as Chotard *et al.* [27] with the difference that we explicitly assume common public parameters cp. As in [27], we assume that each function f can be injectively encoded as a tag t_f (called "label" in [27]) taken as input by the partial functional key generation algorithm.

Definition 2.13. *For a message space $\mathcal{M}$ and tag space $\mathcal{T}$, a* **decentralized multi-client functional encryption** *(DMCFE) scheme between ℓ senders $\{\mathcal{S}_i\}_{i=1}^{\ell}$ and a functional decryptor $\mathcal{FD}$ is specified by the following components.*

$\mathsf{\textbf{Setup}}(\mathsf{cp}, 1^\ell)$: *This is an interactive protocol between the senders $\{\mathcal{S}_i\}_{i=1}^{\ell}$, which allows them to generate their own secret keys sk_i and encryption keys ek_i, for $i \in [\ell]$, as well as a set of public parameters mpk.*

Encrypt(ek_i, x_i, t) : *Takes as input the encryption key* ek_i *of user* $i \in [\ell]$*, a message* x_i *and a tag* $t \in \mathcal{T}$*. It output a ciphertext* $C_{t,i}$*.*

DKeygenShare(sk_i, t_f) : *Takes as input a user's secret key* sk_i *and the label* t_f *of a function* $f : \mathcal{M}^\ell \to \mathcal{R}$*. It outputs a partial functional decryption key* $\mathsf{dk}_{f,i}$ *for the function described by* t_f*.*

DKeygenComb$(\{\mathsf{dk}_{f,i}\}_i, t_f)$: *Takes as input a set of partial functional decryption keys* $\{\mathsf{dk}_{f,i}\}_i$ *and the label* t_f *of a function* $f : \mathcal{M}^\ell \to \mathcal{R}$*. It outputs a full functional decryption key* dk_f *for the function* f *described by* t_f

Decrypt$(\mathsf{dk}_f, t, \mathbf{C})$: *Takes as input a functional decryption key* dk_f*, a tag* t*, and an* ℓ*-vector of ciphertexts* $\mathbf{C} = (C_{t,1}, \ldots, C_{t,\ell})$*. It outputs a function evaluation* $f(\boldsymbol{x}) \in \mathcal{R}$ *or a message* $\perp$ *indicating a decryption failure.*

For simplicity, we assume that mpk is included in all secret keys and encryption keys, as well as in (partial) functional decryption keys. We also assume that a description of f is included in (partial) functional decryption keys.

Correctness. For any $\lambda \in \mathbb{N}$, any $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i=1}^{\ell}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$, any $\boldsymbol{x} \in \mathcal{M}^n$, any tag $t \in \mathcal{T}$ and any function $f : \mathcal{M}^\ell \to \mathcal{R}$, if $C_{t,i} \leftarrow \mathsf{Encrypt}(\mathsf{ek}_i, x_i, t)$ for all $i \in [\ell]$ and $\mathsf{dk}_f \leftarrow \mathsf{DKeyComb}(\{\mathsf{DKeyGenShare}(\mathsf{sk}_i, t_f)\}_i, t_f)$, with overwhelming probability, we have $\mathsf{Decrypt}\big(\mathsf{dk}_f, t, \mathbf{C}_t = (C_{t,1}, \ldots, C_{t,\ell})\big) = f(\boldsymbol{x})$.

Definition 2.14 **($\mathsf{IND\text{-}sec}$ for DMCFE).** *For a DMCFE scheme with* ℓ *senders, we consider the following game between an adversary and a challenger. It involves a set* $\mathcal{HS}$ *of honest senders (initialized to* $\mathcal{HS} := [\ell]$*) and a set* $\mathcal{CS}$ *(initialized to* $\mathcal{CS} := \emptyset$*) of the corrupted senders.*

Initialization: *The challenger* $\mathcal{C}$ *generates* cp *and runs* $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i=1}^{\ell}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$*. Then, it flips a fair coin* $b \leftarrow \{0, 1\}$ *and gives the master public key* mpk *to the adversary* $\mathcal{A}$*.*

Encryption queries: *The adversary* $\mathcal{A}$ *can adaptively make encryption queries* $\mathsf{QEncrypt}(i, x^0, x^1, t)$*, to which the challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x^b, t)$*. For any given pair* (i, t)*, only one query is allowed and subsequent queries involving the same* (i, t) *are ignored.*

Functional decryption key queries: *Via queries* $\mathsf{QDKeygen}(i, f)$*,* $\mathcal{A}$ *can adaptively obtain partial functional decryption keys on behalf of uncorrupted senders. At each query, the challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygenShare}(\mathsf{sk}_i, t_f)$ *if* $i \in \mathcal{HS}$ *(if* $i \in \mathcal{CS}$*, the oracle returns* $\perp$*).*

Corruption queries: *For any user* $i \in \mathcal{HS}$*, the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$ *and the challenger replies by returning* $(\mathsf{sk}_i, \mathsf{ek}_i)$*. It also updates the sets* $\mathcal{HS}$ *and* $\mathcal{CS}$ *by setting* $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ *and* $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$*.*

Finalize: *The adversary outputs a bit* $b' \in \{0, 1\}$*. The adversary* $\mathcal{A}$ *wins if* $\beta = b$*, where* β *is defined as* $\beta := b'$*, unless of the situations below occurred.*

1. *An encryption query* $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *has been made for an index* $i \in \mathcal{CS}$ *with* $x_i^0 \neq x_i^1$*.*
2. *For some label* t*, an encryption query* $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *has been asked for* $i \in \mathcal{HS}$*, but encryption queries* $\mathsf{QEncrypt}(j, x_j^0, x_j^1, t)$ *have not been asked for all* $j \in \mathcal{HS}$*.*

3. *For a tag t and some function f queried to* $\mathsf{QDKeygen}(i,.)$ *for all* $i \in \mathcal{HS}$*, there exists a pair of vectors* $(\boldsymbol{x}^0, \boldsymbol{x}^1)$ *such that* $f(\boldsymbol{x}^0) \neq f(\boldsymbol{x}^1)$*, where*
   - $x_i^0 = x_i^1$ *for all* $i \in \mathcal{CS}$*;*
   - $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *have been asked for all* $i \in \mathcal{HS}$*.*

If any of these events occurred, $\mathcal{A}$*'s output is overwritten by* $\beta \leftarrow U(\{0,1\})$*.*

We say that a DMCFE scheme provides IND *security if, for any efficient adversary* $\mathcal{A}$*, we have* $\mathbf{Adv}^{\mathsf{IND}}(\mathcal{A}) := |\Pr[\beta = 1 \mid b = 1] - \Pr[\beta = 1 \mid b = 0]| \in \mathsf{negl}(\lambda)$*.*

The above definition captures adaptive corruptions in that the $\mathsf{QCorrupt}(\cdot)$ oracle may be invoked at any time during the game. In the static corruption setting, all queries to $\mathsf{QCorrupt}(\cdot)$ should be made at once before the initialization phase. In this case, the sets $\mathcal{HS}$ and $\mathcal{CS}$ are thus determined before the generation of $(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i=1}^{\ell}, \{\mathsf{ek}_i\}_{i=1}^{\ell})$. We denote by $\mathsf{sta\text{-}IND\text{-}sec}$ the latter security game.

Our scheme of Section 4 will be proven secure under static corruptions. We insist that only corruptions are static: the encryption oracle can be queried on adaptively chosen messages (x^0, x^1), which is stronger than the selective security game, where the challenge messages have to be declared upfront.

3 Our MCFE Scheme for Linear Functions

The scheme encrypts $\boldsymbol{x}_i \in \mathbb{Z}^{n_0}$ as a vector $\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i$, where $\mathbf{G}_0$ is a gadget matrix; $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ is an admissible hash of the tag t; and $\mathbf{e}_i$ is a Gaussian noise. This is done in a way that a functional secret key $\mathbf{s}_y = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ allows computing $\sum_{i=1}^{\ell} y_i \cdot \boldsymbol{x}_i$ from $\{\mathbf{C}_{t,i}\}_{i=1}^{\ell}$ by using the public trapdoor of the lattice $\Lambda^\perp(\mathbf{G}_0)$.

We derive $\mathbf{A}(\tau)$ from a set of $2L$ public matrices $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$ and an additional matrix $\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n}$. Like [55], our proof interprets each $\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n \times m}$ as a GSW ciphertext $\mathbf{A}_{i,b} = \mathbf{A} \cdot \mathbf{R}_{i,b} + \mu_{i,b} \cdot \mathbf{G}$, where $\mathbf{R}_{i,b} \in \{-1,1\}^{m \times m}$, $\mu_{i,b} \in \{0,1\}$ and $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the gadget matrix of [57]. Then, we homomorphically compute $\mathbf{A}(\tau)$ as an FHE ciphertext $\mathbf{A} \cdot \mathbf{R}'_\tau + (\prod_{i=1}^{L} \mu_{i,\tau[i]}) \cdot \mathbf{G}$, for some small-norm $\mathbf{R}'_\tau \in \mathbb{Z}^{m \times m}$, which is in turn multiplied by $\mathbf{G}^{-1}(\mathbf{V}^\top \cdot \mathbf{G}_0)$ in such a way that $\mathbf{A}(\tau) = \mathbf{A} \cdot \mathbf{R}_\tau + (\prod_{i=1}^{L} \mu_{i,\tau[i]}) \cdot (\mathbf{V}^\top \cdot \mathbf{G}_0)$. Via a careful choice of $\{\mu_{i,b}\}_{i \in [L], b \in \{0,1\}}$, the properties of admissible hash functions imply that $\prod_{i=1}^{L} \mu_{i,x[i]}$ vanishes in all encryption queries but evaluates to 1 on the challenge tag $\tau^\star$. In order to prevent the encryption oracle from leaking too much about $\mathbf{s}_i \in \mathbb{Z}^n$, we proceed as in [55] and replace the random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ by a lossy matrix $\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}$, where $\hat{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n_1 \times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n_1 \times n})$ and for a small-norm $\mathbf{E} \in \mathbb{Z}^{m \times n}$.

Our construction and proof depart from [55] in that we use an additional multiplication by $\mathbf{G}^{-1}(\mathbf{V}^\top \cdot \mathbf{G}_0)$ in order to introduce a matrix $\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n}$ in the expression of $\mathbf{A}(\tau^\star)$. In addition, unlike [55], we do not rely on a randomness extraction argument to exploit the entropy of $\mathbf{A}(\tau^\star)^\top \cdot \mathbf{s}_i + \mathbf{e}_i$ in the challenge phase. Instead, we use a trapdoor for the matrix $\mathbf{U} = [\begin{smallmatrix} \mathbf{V} \\ \mathbf{C} \end{smallmatrix}]$ to "equivocate" the challenge ciphertexts and explain them as an encryption of $\boldsymbol{x}_{1,i}^\star$ instead of $\boldsymbol{x}_{0,i}^\star$.

Another difference with [55] is that the product $\mathbf{A}(\tau)$ of GSW ciphertexts $\{\mathbf{A}_{i,\tau[i]}\}_{i=1}^{L}$ is evaluated in a sequential manner[7] (as in the "right-spine" PRF construction of [11]) in order for the noise matrix $\mathbf{R}_\tau$ to retain small entries.

3.1 Description

In the following description, we assume public parameters

$$\mathsf{cp} := \Big(\ \lambda,\ \ell_{\max},\ X,\ Y,\ n_0,\ n_1,\ n,\ m,\ \alpha,\ \alpha_1,\ \sigma,\ \ell_t,\ L,\ q,\ \mathsf{AHF}\Big),$$

consisting of a security parameter λ and the following quantities:

- $(X, Y, \ell_{\max}, n_0, n_1, n, m)$, which are all in $\mathsf{poly}(\lambda)$
  $X = 1$, $n_1 = \lambda^d$, $q = 2^{\lambda^{d-1}}$, $\alpha = 2^{-\sqrt{\lambda}}$, $\alpha_1 = 2^{-\lambda^{d-1}+d\log\lambda}$, $n_0 = o(\lambda^{d-2})$, $n = O(\lambda^{2d-1})$, $\sigma = 2^{\lambda^{d-1}-2\lambda}$ and $n_0 \cdot \ell_{max} = O(\lambda^{d-2})$ where d is a constant; for instance $d = 3$ works asymptotically.
- The description of a tag space $\mathcal{T} = \{0,1\}^{\ell_t}$, for some $\ell_t \in \mathsf{poly}(\lambda)$, such that tags may be arbitrary strings (e.g., time period numbers or dataset names).
- The description of a balanced admissible hash function $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$, for a suitable $L \in \Theta(\lambda)$.
- The message space will be $\mathcal{M} = [-X, X]^{n_0}$, for some $n_0 \in \mathsf{poly}(\lambda)$.
- Integers $n, n_0, n_1, m \in \mathsf{poly}(\lambda)$ satisfying the conditions $m > 2n \cdot \lceil \log q \rceil$ and $n > 3 \cdot (n_0 + n_1) \cdot \lceil \log q \rceil$.
- A real $\alpha > 0$ and a Gaussian parameter $\sigma > 0$, which specifies an interval $[-\beta, \beta] = [-\sigma\sqrt{n}, \sigma\sqrt{n}]$ where the coordinates of users' secret keys will live (with probability exponentially close to 1).

Letting $\ell \in \mathsf{poly}(\lambda)$, with $\ell \leq \ell_{max}$, be the number of users, our function space is the set of all functions $f_{\boldsymbol{y}} : \mathbb{Z}^{n_0 \times \ell} \to \mathbb{Z}^{n_0}$ indexed by an integer vector $\boldsymbol{y} \in \mathbb{Z}^\ell$ of infinity norm $\|\boldsymbol{y}\|_\infty < Y$.

We define $\mathbf{G}_0 \in \mathbb{Z}_q^{n_0 \times m}$ to be the gadget matrix

$$\mathbf{G}_0 = [\mathbf{I}_{n_0} \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil}) \mid \mathbf{0}^{n_0} \mid \ldots \mid \mathbf{0}^{n_0}]\ \in \mathbb{Z}_q^{n_0 \times m}$$

where the product $\mathbf{I}_{n_0} \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil})$ is padded with $m - n_0 \cdot \lceil \log q \rceil$ zero columns. We similarly denote by $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ the gadget matrix of rank n:

$$\mathbf{G} = [\mathbf{I}_n \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil}) \mid \mathbf{0}^n \mid \ldots \mid \mathbf{0}^n]\ \in \mathbb{Z}_q^{n \times m}.$$

Our MCFE construction goes as follows.

Setup$(\mathsf{cp}, 1^\ell)$**:** On input of cp and a number of users ℓ, do the following.

[7] In [55], the multiplication of ciphertexts $\{\mathbf{A}_{i,\tau[i]}\}_{i=1}^{L}$ was computed in a parallel fashion $\mathbf{A}_0 \cdot \prod_{i=1}^{L} \mathbf{G}^{-1}(\mathbf{A}_{i,\tau[i]})$ because their initial proof required the matrices $\{\mathbf{A}_{i,b}\}_{i,b}$ to be generated in such a way that $\mathbf{G}^{-1}(\mathbf{A}_{i,b})$ was invertible over $\mathbb{Z}_q$.

1. Choose random matrices $\mathbf{A}_{i,b} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$, for each $i \in [L]$, $b \in \{0,1\}$.
2. Choose a uniformly random matrix $\mathbf{V} \hookleftarrow U(\mathbb{Z}_q^{n_0 \times n})$.
3. For each $i \in [\ell]$, sample $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ and define $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$.

Output the master secret key $\mathsf{msk} := \{\mathsf{ek}_i\}_{i=1}^{\ell}$ and the public parameters

$$\mathsf{mpk} := \Big(\mathsf{cp},\ \mathbf{V},\ \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\ \in \mathbb{Z}_q^{n \times m}\ \}_{i=1}^{L}\Big).$$

DKeygen$(\mathsf{msk}, f_{\boldsymbol{y}})$: Given the master secret key $\mathsf{msk} := \{\mathsf{ek}_i\}_{i=1}^{\ell}$ and a linear function $f_{\boldsymbol{y}} : \mathbb{Z}^{n_0 \times \ell} \to \mathbb{Z}^{n_0}$ defined by an integer vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in \mathbb{Z}^\ell$ which maps an input $\mathbf{X} = [\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_\ell] \in \mathbb{Z}^{n_0 \times \ell}$ to $f_{\boldsymbol{y}}(\mathbf{X}) = \mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$, parse each ek_i as a vector $\mathbf{s}_i \in \mathbb{Z}^n$. Then, compute and output the functional secret key $\mathsf{dk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{s}_{\boldsymbol{y}})$, where $\mathbf{s}_{\boldsymbol{y}} = \sum_{i=1}^{\ell} \mathbf{s}_i \cdot y_i \in \mathbb{Z}^n$.

Encrypt$(\mathsf{ek}_i, \boldsymbol{x}_i, t)$: Given $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$, $\boldsymbol{x}_i \in [-X, X]^{n_0}$, and $t \in \{0,1\}^{\ell_t}$,

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau[1] \ldots \tau[L]$.
2. Define $\mathbf{W} = \mathbf{G}_0^\top \cdot \mathbf{V} \in \mathbb{Z}_q^{m \times n}$ and compute

$$\mathbf{A}(\tau) = \mathbf{A}_{L,\tau[L]} \cdot \mathbf{G}^{-1}\Big(\mathbf{A}_{L-1,\tau[L-1]} \cdot \mathbf{G}^{-1}\big(\ldots \mathbf{A}_{2,\tau[2]} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{1,\tau[1]}\big)\big)\Big) \cdot \mathbf{G}^{-1}(\mathbf{W}^\top)\ \in \mathbb{Z}_q^{n \times m}. \qquad (2)$$

3. Sample a noise vector $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$. Then, compute and output

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i\ \in \mathbb{Z}_q^m.$$

Decrypt$(\mathsf{dk}_{\boldsymbol{y}}, t, \mathbf{C}_t)$: On input of a functional secret key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{s}_{\boldsymbol{y}})$ for a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in [-Y, Y]^\ell$, a tag $t \in \{0,1\}^{\ell_t}$, and an ℓ-vector of ciphertexts $\mathbf{C}_t = (\mathbf{C}_{t,1}, \ldots, \mathbf{C}_{t,\ell}) \in (\mathbb{Z}_q^m)^\ell$, conduct the following steps.

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau[1] \ldots \tau[L]$.
2. Compute $\mathbf{A}(\tau) \in \mathbb{Z}_q^{n \times m}$ as per (2).
3. Compute $\mathbf{f}_{t,\boldsymbol{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{C}_{t,i} - \mathbf{A}(\tau)^\top \cdot \mathbf{s}_{\boldsymbol{y}} \mod q$.
4. Interpret $\mathbf{f}_{t,\boldsymbol{y}} \in \mathbb{Z}_q^m$ as a vector of the form $\mathbf{f}_{t,\boldsymbol{y}} = \mathbf{G}_0^\top \cdot \boldsymbol{z} + \tilde{\mathbf{e}} \mod q$, for some error vector $\tilde{\mathbf{e}} \in [-B, B]^m$. Using the public trapdoor of $\Lambda^\perp(\mathbf{G}_0)$, compute and output the underlying vector $\boldsymbol{z} \in [-\ell \cdot X \cdot Y, \ell \cdot X \cdot Y]^{n_0}$.

The following lemma is proved in the full version of the paper.

Lemma 3.1 (Correctness). *Assume that $\alpha q = \omega(\sqrt{\log \ell})$, $Y \cdot \ell \cdot \alpha q \cdot \log q < q/2$ and $\ell \cdot X \cdot Y < q/2$. Then, for any $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\lambda)$, any message $\mathbf{X} = [\boldsymbol{x}_1 | \cdots | \boldsymbol{x}_\ell] \in [-X, X]^{n_0 \times \ell}$, any $\boldsymbol{y} \in [-Y, Y]^\ell$, any tag $t \in \{0,1\}^{\ell_t}$, algorithm $\mathsf{Decrypt}(\mathsf{dk}_{\boldsymbol{y}}, t, \mathbf{C}_t)$ outputs $\boldsymbol{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$ with probability exponentially close to 1, where $\mathbf{C}_{t,i} \leftarrow \mathsf{Encrypt}(\mathsf{ek_i}, \boldsymbol{x_i}, \mathsf{t})$ and $\mathsf{dk}_y \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f_{\boldsymbol{y}})$.*

3.2 Security

We now prove the security of the scheme in the sense of Definition 2.12 (and thus Definition 2.11 modulo some loss of tightness in the reduction).

For the current parameters $n_1 = \lambda^d$, $q = 2^{\lambda^{d-1}}$, and $\alpha_1 = 2^{-\lambda^{d-1}+d\log\lambda}$, $\alpha_1 q = \Omega(\sqrt{n_1})$, we know from [60] that $\mathsf{LWE}_{q,n_1,\alpha_1}$ is at least as hard as GapSVP_γ, with $\gamma = \tilde{O}(n_1/\alpha_1) = \tilde{O}(2^{\lambda^{d-1}})$. The best known algorithms [63] for solving GapSVP_γ run in $2^{\tilde{O}(\frac{n_1}{\log\gamma})}$, which for our parameters is $2^{\tilde{O}(\lambda)}$.

Theorem 3.2. *The above MCFE schemes provides adaptive security under the* $\mathsf{LWE}_{q,m,n_1,\alpha_1}$ *assumption.*

Proof. The proof considers a sequence of games. In each game, we denote by W_i the event that $b' = b$. For each i, the adversary's advantage function in Game_i is $\mathbf{Adv}_i(\mathcal{A}) := |\Pr[b' = b] - 1/2| = \frac{1}{2} \cdot |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|$.

Game_0: This is the real security game. We denote by $t^\star$ the tag of the challenge phase while $t^{(1)}, \ldots, t^{(Q)}$ are the tags involved in encryption queries. Namely, for each $j \in [Q]$, $t^{(j)}$ stands for the j-th distinct tag involved in an encryption query. Since up to ℓ encryption queries $(i, \boldsymbol{x}_i, t)$ are allowed for each tag t, the adversary can make a total of $\ell \cdot Q$ encryption queries. The game begins with the challenger initially choosing encryption keys $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ by sampling $\mathsf{ek}_i = \mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ for each $i \in [\ell]$. In addition, the challenger flips a fair coin $b \hookleftarrow U(\{0,1\})$ which will determine the response to challenge queries. At each corruption query $i \in [\ell]$, the adversary obtains ek_i and the challenger updates a set $\mathcal{CS} := \mathcal{CS} \cup \{i\}$, which is initially empty. At each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the challenger samples $\mathbf{e}_i^{(j)} \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and returns

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i^{(j)} + \mathbf{A}(\tau^{(j)})^\top \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)} \ \in \mathbb{Z}_q^m,$$

where $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$. In the challenge phase, the adversary $\mathcal{A}$ chooses a fresh tag $t^\star$ and two vectors of messages $\mathbf{X}_0^\star = [\boldsymbol{x}_{0,1}^\star \mid \ldots \mid \boldsymbol{x}_{0,\ell}^\star] \in [-X, X]^{n_0 \times \ell}$ and $\mathbf{X}_1^\star = [\boldsymbol{x}_{1,1}^\star \mid \ldots \mid \boldsymbol{x}_{1,\ell}^\star] \in [-X, X]^{n_0 \times \ell}$ subject to the constraint that, for any private key query $\boldsymbol{y} \in [-Y, Y]^\ell$ made by $\mathcal{A}$, we must have $\mathbf{X}_0^\star \cdot \boldsymbol{y} = \mathbf{X}_1^\star \cdot \boldsymbol{y}$ over $\mathbb{Z}$. In addition, the invariant that $\boldsymbol{x}_{0,i}^\star = \boldsymbol{x}_{1,i}^\star$ for any $i \in \mathcal{CS}$ must be satisfied at any time during the game. In response to a challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$, the challenger generates a challenge ciphertext $\mathbf{C}_{t^\star,i}$, where

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{b,i}^\star + \mathbf{A}(\tau^\star)^\top \cdot \mathbf{s}_i + \mathbf{e}_i^\star, \tag{3}$$

where $\tau^\star = \mathsf{AHF}(t^\star)$ and $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ for all $i \in [\ell]$.

When $\mathcal{A}$ halts, it outputs $\hat{b} \in \{0,1\}$ and the challenger defines $b' := \hat{b}$. We have $\mathbf{Adv}(\mathcal{A}) := |\Pr[W_0] - 1/2|$, where W_0 is event that $b' = b$.

Game_1: This game is identical to Game_0 except for the following changes. First, the challenger runs $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q, \delta)$ to generate a key $K \in \{0, 1, \perp\}^L$

for a balanced admissible hash function $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$. When the adversary halts and outputs $\hat{b} \in \{0,1\}$, the challenger checks if the conditions

$$P_K(t^{(1)}) = \cdots = P_K(t^{(Q)}) = 1 \;\wedge\; P_K(t^\star) = 0 \tag{4}$$

are satisfied. If conditions (4) do not hold, the challenger ignores $\mathcal{A}$'s output $\hat{b} \in \{0,1\}$ and overwrites it with a random bit $b'' \hookleftarrow \{0,1\}$ to define $b' = b''$. If conditions (4) are satisfied, the challenger sets $b' = \hat{b}$. By Lemma 2.8,

$$|\Pr[W_1] - 1/2| \geq \gamma_{\min} \cdot \mathbf{Adv}(\mathcal{A}) - \frac{1}{2} \cdot (\gamma_{\max} - \gamma_{\min}) = \tau,$$

where $\tau(\lambda)$ is a noticeable function.

Game$_2$: In this game, we modify the generation of mpk in the following way. Initially, the challenger samples a uniformly random matrix $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$. Next, for each $i \in [L]$, it samples $\mathbf{R}_{i,0}, \mathbf{R}_{i,1} \hookleftarrow U(\{-1,1\})^{m \times m}$ and defines $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^L$ as follows for all $i \in [L]$ and $j \in \{0,1\}$:

$$\mathbf{A}_{i,j} := \begin{cases} \mathbf{A} \cdot \mathbf{R}_{i,j} & \text{if} \quad (j \neq K_i) \;\wedge\; (K_i \neq \perp) \\ \mathbf{A} \cdot \mathbf{R}_{i,j} + \mathbf{G} & \text{if} \quad (j = K_i) \;\vee\; (K_i = \perp) \end{cases} \tag{5}$$

Since $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ was chosen uniformly, the Leftover Hash Lemma ensures that $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^L$ are statistically independent and uniformly distributed over $\mathbb{Z}_q^{n \times m}$. It follows that $|\Pr[W_2] - \Pr[W_1]| \leq L \cdot 2^{-\lambda}$.

We note that, at each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the admissible hash function maps $t^{(j)}$ to $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$, which is itself mapped to a GSW encryption

$$\mathbf{A}(\tau^{(j)}) = \mathbf{A} \cdot \mathbf{R}_{\tau^{(j)}} + (\prod_{i=1}^{L} \mu_i) \cdot \mathbf{W}^\top, \tag{6}$$

of a product $\prod_{i=1}^L \mu_i$, for some small norm matrix $\mathbf{R}_{\tau^{(j)}} \in \mathbb{Z}^{m \times m}$, where

$$\mu_i := \begin{cases} 0 & \text{if} \quad (\mathsf{AHF}(t^{(j)}))_i \neq K_i) \;\wedge\; (K_i \neq \perp) \\ 1 & \text{if} \quad (\mathsf{AHF}(t^{(j)}))_i = K_i) \;\vee\; (K_i = \perp) \end{cases}$$

If conditions (4) are satisfied, at each encryption query $(i, x_i^{(j)}, t^{(j)})$, the admissible hash function ensures that $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$ satisfies

$$\mathbf{A}(\tau^{(j)}) = \mathbf{A} \cdot \mathbf{R}_{\tau^{(j)}} \qquad\qquad \forall j \in [Q], \tag{7}$$

for some small norm $\mathbf{R}_{\tau^{(j)}} \in \mathbb{Z}^{m \times m}$. Moreover, the challenge tag $t^\star$ is mapped to an L-bit string $\tau^\star = \mathsf{AHF}(t^\star)$ such that

$$\mathbf{A}(\tau^\star) = \mathbf{A} \cdot \mathbf{R}_{\tau^\star} + \mathbf{W}^\top = \mathbf{A} \cdot \mathbf{R}_{\tau^\star} + \mathbf{V}^\top \cdot \mathbf{G}_0 \tag{8}$$

Game_3**:** In this game, we modify the distribution of mpk and replace the uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ by a lossy matrix such that

$$\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E} \ \in \mathbb{Z}_q^{m \times n}, \tag{9}$$

where $\hat{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n_1 \times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n_1 \times n})$ and $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m \times n}, \alpha_1 q}$, for $n_1 \ll n$. The matrix (9) is thus "close" to a matrix $\hat{\mathbf{A}}^\top \cdot \mathbf{C}$ of much lower rank than n. Under the LWE assumption in dimension n_1 with error rate α_1, this change should not significantly affect $\mathcal{A}$'s behavior and a straightforward reduction $\mathcal{B}$ shows that $|\Pr[W_3] - \Pr[W_2]| \leq n \cdot \mathbf{Adv}_{\mathcal{B}}^{\mathsf{LWE}_{q,m,n_1,\alpha_1}}(\lambda)$, where the factor n comes from the use of an LWE assumption with n secrets.

Game_4**:** In this game, we modify the encryption oracle. At each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the challenger generates the ciphertext by computing:

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i^{(j)} + \mathbf{R}_{\tau^{(j)}}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)} \ \in \mathbb{Z}_q^m, \tag{10}$$

and for each challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$ the challenger replies with:

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{b,i}^\star + \Big(\mathbf{R}_{\tau^\star}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{G}_0^\top \cdot \mathbf{V}\Big) \cdot \mathbf{s}_i + \mathbf{e}_i^\star \ \in \mathbb{Z}_q^m \tag{11}$$

where $\mathbf{e}_i^{(j)} \hookleftarrow D_{\mathbb{Z}^m, \alpha q}$ and $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m, \alpha q}$. The only difference between Game_3 and Game_4 is thus that the terms $\mathbf{R}_{\tau^{(j)}}^\top \cdot \mathbf{E} \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)}$ and $\mathbf{R}_{\tau^\star}^\top \cdot \mathbf{E} \cdot \mathbf{s}_i + \mathbf{e}_i^\star$ are replaced by $\mathbf{e}_i^{(j)}$ and $\mathbf{e}_i^\star$ respectively, at each encryption or challenge query. However, the smudging lemma (Lemma 2.4) ensures that the two distributions are statistically close as long as α is sufficiently large with respect to α_1 and σ. Concretely, Lemma 3.3 implies $|\Pr[W_4] - \Pr[W_3]| \leq \ell \cdot (Q+1) \cdot 2^{-\Omega(\lambda)}$.

Game_5**:** This game is like Game_4 but we modify the challenge oracle. Instead of encrypting $\mathbf{X}_b^\star = [\boldsymbol{x}_{b,1}^\star \mid \ldots \mid \boldsymbol{x}_{b,\ell}^\star]$ as in (11), the challenger encrypts a linear combination of $\mathbf{X}_0^\star$ and $\mathbf{X}_1^\star$. It initially chooses a uniformly random $\gamma \hookleftarrow U(\mathbb{Z}_q)$ and, at each challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$, computes $\mathbf{C}_{t^\star,i}$ as

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \big((1-\gamma) \cdot \boldsymbol{x}_{b,i}^\star + \gamma \cdot \boldsymbol{x}_{1-b,i}^\star\big) + \Big(\mathbf{R}_{\tau^\star}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{G}_0^\top \cdot \mathbf{V}\Big) \cdot \mathbf{s}_i + \mathbf{e}_i^\star,$$

with $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m, \alpha q}$, for all $i \in [\ell]$. Lemma 3.4 shows that Game_4 and Game_5 are negligibly far part as $|\Pr[W_5] - \Pr[W_4]| \leq 2^{-\Omega(\lambda)}$.

In Game_5, we clearly have $\Pr[W_5] = 1/2$ since the challenge ciphertexts $(\mathbf{C}_{t,1}^\star, \ldots, \mathbf{C}_{t,\ell}^\star)$ reveal no information about $b \in \{0,1\}$. □

Lemma 3.3. *Let* $\mathbf{R}_\tau \in \mathbb{Z}^{m \times m}$ *be as in equation (6). Let* $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m \times n}, \alpha_1 q}$ *and* $\mathbf{s} \hookleftarrow D_{\mathbb{Z}^n, \sigma}$*. If* $\alpha_1 q = \omega(\sqrt{\log n})$*,* $\sigma = \omega(\sqrt{\log n})$ *and* $\alpha \geq 2^\lambda \cdot L \cdot m^4 \cdot n^{3/2} \cdot \alpha_1 \cdot \sigma$*, we have the statistical distance upper bound* $\Delta\left(D_{\mathbb{Z}^m, \alpha q},\ \mathbf{R}_\tau^\top \cdot \mathbf{E} \cdot \mathbf{s} + D_{\mathbb{Z}^m, \alpha q}\right) \leq 2^{-\lambda}$*.* (The proof is given in the full version of the paper.)

Lemma 3.4. *We have* $|\Pr[W_5] - \Pr[W_4]| \le 2^{-\Omega(\lambda)}$.

Proof. To prove the result, we resort to a technique of guessing in advance the difference $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$, which was previously used in [67,13] and can be seen as complexity leveraging with respect to a statistical argument. We consider the following variants of Game_4 and Game_5, respectively.

We define Game'_4 and Game'_5 simultaneously by using an index $k \in \{4, 5\}$:

Game'_k: This game is like Game_k with one difference in the setup phase. To generate mpk, the challenger $\mathcal{B}$ generates a statistically uniform $\mathbf{U} \in \mathbb{Z}_q^{(n_0+n_1)\times n}$ with a trapdoor $\mathbf{T}_U$ for the lattice $\Lambda^\perp(\mathbf{U})$. Then, $\mathcal{B}$ parses $\mathbf{U}$ as

$$\mathbf{U} = \begin{bmatrix} \mathbf{V} \\ \hline \mathbf{C} \end{bmatrix} \in \mathbb{Z}_q^{(n_0+n_1)\times n},$$

where $\mathbf{V} \in \mathbb{Z}_q^{n_0\times n}$ and $\mathbf{C} \in \mathbb{Z}_q^{n_1\times n}$ are statistically independent and uniform over $\mathbb{Z}_q$. Next, it computes

$$\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E} \in \mathbb{Z}_q^{m\times n},$$

where $\hat{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n_1\times m})$ and $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m\times n},\alpha_1 q}$. The obtained matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is then used to generate $\{\mathbf{A}_{i,j}\}_{i\in[L],j\in\{0,1\}}$ as per (5). The upper part $\mathbf{V} \in \mathbb{Z}_q^{n_0\times n}$ of $\mathbf{U}$ is included in mpk, the distribution of which is statistically close to that of Game_k: we indeed have $|\Pr[W'_k] - \Pr[W_k]| \le 2^{-\Omega(\lambda)}$.

We do the same as above and define Game$''_4$ and Game$''_5$ simultaneously by using an index $k \in \{4, 5\}$:

Game''_k: This game is identical to Game'_k with the following difference. At the outset of the game, the challenger randomly chooses $\mathbf{\Delta X} \hookleftarrow U([-2X, 2X]^{n_0\times\ell})$ as a guess for the difference $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$ between the challenge messages $\mathbf{X}^\star_0, \mathbf{X}^\star_1$. In the challenge phase, the challenger checks if $\mathbf{\Delta X} = \mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$. If not, it aborts and replaces $\mathcal{A}$'s output $\hat{b}$ with a random bit $b'' \hookleftarrow U(\{0, 1\})$. If the guess for $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$ was successful (we call Guess this event), the challenger proceeds exactly as it did in Game'_k.

Since the choice of $\mathbf{\Delta X} \hookleftarrow U([-2X, 2X]^{n_0\times\ell})$ is completely independent of $\mathcal{A}$'s view, we clearly have $\Pr[\mathsf{Guess}] = 1/(4X)^{n_0\ell}$. Since Game''_4 is identical to Game'_4 when Guess occurs, this implies $\mathbf{Adv}_{4'}(\mathcal{A}) = (4X)^{n_0\ell} \cdot \mathbf{Adv}_{4''}(\mathcal{A})$. Indeed,

$$\begin{aligned}
\mathbf{Adv}_{4''}(\mathcal{A}) := \frac{1}{2} \cdot |\Pr[b' = 1 \mid b = 1, \mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] + \frac{1}{2} \cdot \Pr[\neg\mathsf{Guess}] \\
- \Pr[b' = 1 \mid b = 0, \mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] - \frac{1}{2} \cdot \Pr[\neg\mathsf{Guess}]| \\
= \frac{1}{2} \cdot \Pr[\mathsf{Guess}] \cdot |\Pr[b' = 1 \mid b = 1, \mathsf{Guess}] - \Pr[b' = 1 \mid b = 0, \mathsf{Guess}]| \\
= \Pr[\mathsf{Guess}] \cdot \mathbf{Adv}_{4'}(\mathcal{A}) = \frac{1}{(4X)^{n_0\ell}} \cdot \mathbf{Adv}_{4'}(\mathcal{A})
\end{aligned}$$

and we can similarly show that $\mathbf{Adv}_{5'}(\mathcal{A}) = (4X)^{n_0\ell} \cdot \mathbf{Adv}_{5''}(\mathcal{A})$.

Game_5''': This game is identical to Game_4'' except that encryption keys $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ are replaced by alternative encryption keys $\{\mathsf{ek}_i'\}_{i=1}^{\ell}$, which are generated as follows. After having sampled $\mathsf{ek}_i = \mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ for all $i \in [\ell]$, the challenger $\mathcal{B}$ chooses $\gamma \hookleftarrow U(\mathbb{Z}_q)$ and uses the trapdoor $\mathbf{T_U}$ for $\Lambda^{\perp}(\mathbf{U})$ to sample a small-norm matrix $\mathbf{T} \in \mathbb{Z}^{n \times n_0}$ satisfying

$$\mathbf{U} \cdot \mathbf{T} = \begin{bmatrix} \gamma \cdot \mathbf{I}_{n_0} \\ \mathbf{0}^{n_1 \times n_0} \end{bmatrix} \bmod q, \tag{12}$$

so that $\mathbf{V} \cdot \mathbf{T} = \gamma \cdot \mathbf{I}_{n_0} \bmod q$ and $\mathbf{C} \cdot \mathbf{T} = \mathbf{0}^{n_1 \times n_0} \bmod q$. For each $i \in [\ell]$, $\mathcal{B}$ then defines the alternative key $\mathsf{ek}_i' = \mathbf{s}_i'$ of user i to be

$$\mathbf{s}_i' = \mathbf{s}_i + \mathbf{T} \cdot \boldsymbol{\Delta x}_i \ \in \mathbb{Z}^n \qquad \forall i \in [\ell], \tag{13}$$

where $\boldsymbol{\Delta x}_i$ is the i-th column of $\boldsymbol{\Delta}\mathbf{X}$ (i.e., the guess for $\boldsymbol{x}_{1-b,i}^{\star} - \boldsymbol{x}_{b,i}^{\star}$). These modified encryption keys $\{\mathsf{ek}_i' = \mathbf{s}_i'\}_{i=1}^{\ell}$ are used to answer all encryption queries and to generate the challenge ciphertext. At each corruption query i, the adversary is also given ek_i' instead of ek_i.

We first claim that, conditionally on Guess, Game_5''' is statistically close to Game_4''. To see this, we first argue that trading $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ for $\{\mathsf{ek}_i'\}_{i=1}^{\ell}$ has no incidence on queries made by a legitimate adversary:

- We have $\mathbf{C} \cdot \mathbf{s}_i' = \mathbf{C} \cdot \mathbf{s}_i \bmod q$, so that encryption queries obtain the same responses no matter which key set is used among $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ and $\{\mathsf{ek}_i'\}_{i=1}^{\ell}$.
- We have $\sum_{i=1}^{\ell} \mathbf{s}_i' \cdot \boldsymbol{y}_i = \sum_{i=1}^{\ell} \mathbf{s}_i \cdot \boldsymbol{y}_i$ so long as the adversary only obtains private keys for vectors $\boldsymbol{y} \in \mathbb{Z}^{\ell}$ such that $(\mathbf{X}_0^{\star} - \mathbf{X}_1^{\star}) \cdot \boldsymbol{y} = \mathbf{0}$ (over $\mathbb{Z}$).
- For any corrupted user $i \in \mathcal{CS}$, it should be the case that $\boldsymbol{x}_{0,i}^{\star} = \boldsymbol{x}_{1,i}^{\star}$, meaning that $\mathbf{s}_i' = \mathbf{s}_i$ as long as Guess occurs.

This implies that Game_5''' is identical to Game_4'', except that users' secret keys are defined via (13) and thus have a slightly different distribution. Lemma 3.5 shows that the statistical distance between the distributions of $\{\mathbf{s}_i'\}_{i=1}^{\ell}$ and $\{\mathbf{s}_i\}_{i=1}^{\ell}$ is at most $2^{-\lambda} \cdot (4X)^{-n_0\ell}$. This implies that Game_4'' and Game_5''' are statistically close assuming that Guess occurs. When Guess does not occur, both games output a random $b' \hookleftarrow U(\{0,1\})$, so that $|\Pr[W_5'''] - \Pr[W_4'']| \leq 2^{-\lambda} \cdot (4X)^{-n_0\ell}$.

We finally claim that, from the adversary's view Game_5''' is identical to Game_5''. Indeed, our choice of $\mathbf{T}$ ensures that $\mathbf{V} \cdot \mathbf{T} = \gamma \cdot \mathbf{I}_{n_0} \bmod q$, so that we have $\mathbf{V} \cdot \mathbf{s}_i' = \mathbf{V} \cdot \mathbf{s}_i + \gamma \cdot (\boldsymbol{x}_{1-b,i}^{\star} - \boldsymbol{x}_{b,i}^{\star}) \bmod q$. This implies

$$\begin{aligned} \mathbf{C}_{t^{\star},i} &= \mathbf{G}_0^{\top} \cdot \boldsymbol{x}_{b,i}^{\star} + (\mathbf{R}_{\tau^{\star}}^{\top} \cdot \hat{\mathbf{A}}^{\top} \cdot \mathbf{C} + \mathbf{G}_0^{\top} \cdot \mathbf{V}) \cdot \mathbf{s}_i' + \mathbf{e}_i{}^{\star} \\ &= \mathbf{G}_0^{\top} \cdot \big((1-\gamma) \cdot \boldsymbol{x}_{b,i}^{\star} + \gamma \cdot \boldsymbol{x}_{1-b,i}^{\star}\big) + (\mathbf{R}_{\tau^{\star}}^{\top} \cdot \hat{\mathbf{A}}^{\top} \cdot \mathbf{C} + \mathbf{G}_0^{\top} \cdot \mathbf{V}) \cdot \mathbf{s}_i + \mathbf{e}_i{}^{\star} \end{aligned}$$

which is exactly the distribution from Game$_5''$.

Putting the above altogether, we find $|\Pr[W_4''] - \Pr[W_5'']| \leq 2^{-\Omega(\lambda)} \cdot (4X)^{-n_0\ell}$, which in turn implies $|\Pr[W_4] - \Pr[W_5]| \leq 2^{-\Omega(\lambda)}$, as claimed.

□

Lemma 3.5. *If $\sigma \geq 2^{\lambda} \cdot n_0 \cdot (4X)^{n_0\ell+1} \cdot \omega(n^2\sqrt{\log n})$, then we have the inequality $\Delta\left(D_{\mathbb{Z}^n,\sigma}, \mathbf{T} \cdot \Delta \boldsymbol{x_i} + D_{\mathbb{Z}^n,\sigma}\right) \leq 2^{-\lambda} \cdot (4X)^{-n_0\ell}$.* (The proof is in the full version.)

4 A DMCFE Scheme for Linear Functions

As in [27], our DMCFE scheme combines two instances of the underlying centralized scheme of Section 3. While the second instance is used exactly in the same way as in the centralized construction, the first instance is used for the sole purpose of generating partial functional secret keys without having the senders communicate with one another. As in [27], the senders have to initially run an interactive protocol in order to jointly generate public parameters for the two schemes. Note that this protocol is the only step that requires interaction among senders and it is only executed once. This interactive step ends with each sender holding an encryption key $\mathsf{ek}_i = (\mathbf{s}_i, \mathbf{t}_i)$ comprised of encryption keys for the two MCFE instances. The distributed protocol also ensures that a functional secret key $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i$ for the all-one vector $(1, 1, \ldots, 1)^\top \in \mathbb{Z}^\ell$ be made publicly available for the first MCFE instance. Later on, when a decryptor wishes to obtain a partial functional secret key $\mathsf{dk}_{f,i}$ for a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top$ from the i-th sender $\mathcal{S}_i$, the latter can generate an MCFE encryption of the vector $y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ under his secret key $\mathbf{t}_i$. Having obtained partial functional secret keys $\mathsf{dk}_{f,i}$ from all senders $\{\mathcal{S}_i\}_{i=1}^{\ell}$, the decryptor can then use the functional secret key $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i$ to compute $\mathbf{s}_y = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$.

4.1 Description

We assume global public parameters

$$\mathsf{cp} := \Big(\lambda,\ \ell_{\max},\ X,\ \bar{X},\ Y,\ \bar{Y},\ n_0,\ n_1,\ \bar{n}_1,\ n,\ \bar{n},,\ m,\ \bar{m},\ \alpha,$$
$$\alpha_1,\ \bar{\alpha}_1,\ \sigma,\ \bar{\sigma},\ \ell_t,\ \ell_f,\ L,\ q,\ \bar{q},\ \mathsf{AHF}_t,\ \mathsf{AHF}_f \Big),$$

which specify a security parameter λ and the following quantities

- Let $\ell_{max} = \lambda^k$, $n_1 = \lambda^d$, $\bar{d} = 3d + k - 1$, $q = 2^{\lambda^{d-1}+\lambda}$, $\bar{q} = 2^{\lambda^{\bar{d}-1}+\lambda}$, $\bar{n}_1 = \lambda^{\bar{d}}$, $\alpha_1 = 2^{-\lambda^{d-1}+d\log\lambda}$, $\bar{\alpha}_1 = 2^{-\lambda^{\bar{d}-1}+\bar{d}\log\lambda}$, $\alpha = 2^{-\sqrt{\lambda}}$, $n_0 \cdot \ell_{max} = O(\lambda^{d-2})$, $n_0 = O(\lambda^{d-2})$, $n = O(\lambda^{2d-1})$, $\bar{n} = O(\lambda^{4d+k-2})$, $X = 1$, $\bar{Y} = 1$, $\sigma = 2^{\lambda^{d-1}-2\lambda}$, $\bar{\sigma} = 2^{\lambda^{\bar{d}-1}-2\lambda}$, $\bar{X} = 2\ell \cdot Y \cdot \sigma\sqrt{n}$ and the rest of the parameters $Y, m, \bar{m}$ are all in $\mathsf{poly}(\lambda)$
- A tag length $\ell_t \in \Theta(\lambda)$ and a length $\ell_f \in \Theta(\lambda)$ of function labels.
- Dimensions $n, m, n_0, n_1, \bar{n}, \bar{m} \in \mathsf{poly}(\lambda)$ such that $n > 3 \cdot (n_0 + n_1) \cdot \lceil \log q \rceil$, $m > 2 \cdot n \cdot \lceil \log q \rceil$, $\bar{n} > 3 \cdot (n + \bar{n}_1) \cdot \lceil \log \bar{q} \rceil$ and $\bar{m} > 2 \cdot \bar{n} \cdot \lceil \log \bar{q} \rceil$.
- The description of balanced admissible hash functions $\mathsf{AHF}_t : \{0,1\}^{\ell_t} \to \{0,1\}^L$ and $\mathsf{AHF}_f : \{0,1\}^{\ell_f} \to \{0,1\}^L$, for a suitable $L \in \Theta(\lambda)$.
- A real $\alpha > 0$ and a Gaussian parameter $\sigma > 0$, which will specify an interval $[-\beta, \beta] = [-\sigma\sqrt{n}, \sigma\sqrt{n}]$ where the coordinates of the secret will live (with probability exponentially close to 1).

We define $\bar{\mathbf{G}} \in \mathbb{Z}_{\bar{q}}^{n \times \bar{m}}$ to be the gadget matrix

$$\bar{\mathbf{G}} = [\mathbf{I}_n \otimes (1, 2, 4, \ldots, 2^{\lceil \log \bar{q} \rceil}) \mid \mathbf{0}^n \mid \ldots \mid \mathbf{0}^n] \ \in \mathbb{Z}_{\bar{q}}^{n \times \bar{m}}$$

where $\mathbf{I}_n \otimes (1, 2, 4, \ldots, 2^{\lceil \log \bar{q} \rceil})$ is padded with $\bar{m} - n \cdot \lceil \log \bar{q} \rceil$ zero columns.

Setup$(\mathsf{cp}, 1^\ell)$**:** On input of a number of users $\ell < \ell_{\max}$, the senders $\{\mathcal{S}_i\}_{i=1}^{\ell}$ run an interactive protocol at the end of which the following quantities are made publicly available.

- Random matrices $\mathbf{A}_{i,b} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$, for each $i \in [L]$, $b \in \{0, 1\}$.
- Random matrices $\mathbf{B}_{i,b} \hookleftarrow U(\mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}})$, for each $i \in [L]$, $b \in \{0, 1\}$.
- Random matrices $\mathbf{V} \hookleftarrow U(\mathbb{Z}_q^{n_0 \times n})$, $\bar{\mathbf{V}} \hookleftarrow U(\mathbb{Z}_{\bar{q}}^{n \times \bar{n}})$.
- The sum $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i \ \in \mathbb{Z}^{\bar{n}}$ of Gaussian vectors $\mathbf{t}_i \hookleftarrow D_{\mathbb{Z}^{\bar{n}}, \bar{\sigma}}$ for $i \in [\ell]$.

In addition, for each $i \in [\ell]$, the i-th sender $\mathcal{S}_i$ privately obtains the following:

- The i-th term $\mathbf{t}_i \in \mathbb{Z}^{\bar{n}}$ of the sum $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i$.
- A Gaussian vector $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n, \sigma}$, which is used to define $\mathcal{S}_i$'s encryption key $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$ and the corresponding secret key $\mathsf{sk}_i = (\mathbf{s}_i, \mathbf{t}_i) \in \mathbb{Z}^n \times \mathbb{Z}^{\bar{n}}$.

The master public key is defined to be

$$\mathsf{mpk} := \Big(\mathsf{cp},\ \mathbf{V},\ \bar{\mathbf{V}}, \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1} \ \in \mathbb{Z}_q^{n \times m}\ \}_{i=1}^{L},$$
$$\{\mathbf{B}_{i,0}, \mathbf{B}_{i,1} \ \in \mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}}\ \}_{i=1}^{L},\ \mathbf{t}\Big),$$

while $\mathcal{S}_i$ obtains $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$ and $\mathsf{sk}_i = (\mathbf{s}_i, \mathbf{t}_i) \in \mathbb{Z}^n \times \mathbb{Z}^{\bar{n}}$ for each $i \in [\ell]$.

DKeygenShare(sk_i, t_f) : Given the secret key $\mathsf{sk}_i = (\mathbf{s}_i, \mathbf{t}_i) \in \mathbb{Z}^n \times \mathbb{Z}^{\bar{n}}$ and the label t_f of a linear function $f_{\boldsymbol{y}} : \mathbb{Z}^{n_0 \times \ell} \to \mathbb{Z}^{n_0}$ described by a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in [-Y, Y]^\ell$, conduct the following steps.

1. Compute $\tau_f = \tau_f[1] \ldots \tau_f[L] = \mathsf{AHF}_f(t_f) \in \{0, 1\}^L$ as well as
$$\mathbf{B}(\tau_f) = \mathbf{B}_{L,\tau_f[L]} \cdot \bar{\mathbf{G}}^{-1}\Big(\mathbf{B}_{L-1,\tau_f[L-1]} \cdot \bar{\mathbf{G}}^{-1}\big(\ldots \mathbf{B}_{2,\tau_f[2]} \cdot \bar{\mathbf{G}}^{-1}\big(\mathbf{B}_{1,\tau_f[1]}\big)\big)\Big)$$
$$\cdot\, \bar{\mathbf{G}}^{-1}(\bar{\mathbf{W}}^\top) \ \in \mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}}, \qquad (14)$$
where $\bar{\mathbf{W}} = \bar{\mathbf{G}}^\top \cdot \bar{\mathbf{V}} \in \mathbb{Z}_{\bar{q}}^{\bar{m} \times \bar{n}}$.
2. Sample a noise vector $\mathbf{e}_{f,i} \hookleftarrow D_{\mathbb{Z}^{\bar{m}}, \alpha \bar{q}}$. Then, compute
$$\mathsf{dk}_{f,i} = \bar{\mathbf{G}}^\top \cdot (y_i \cdot \boldsymbol{s}_i) + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i} \ \in \mathbb{Z}_{\bar{q}}^{\bar{m}}. \qquad (15)$$

Output the partial functional decryption key $\mathsf{dk}_{f,i} \in \mathbb{Z}_{\bar{q}}^{\bar{m}}$.

DKeygenComb$(\{\mathsf{dk}_{f,i}\}_i, t_f)$: Given the label of a function described by a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in [-Y, Y]^\ell$ and ℓ partial functional keys $\{\mathsf{dk}_{f,i}\}_{i=1}^{\ell}$ where $\mathsf{dk}_{f,i} \in \mathbb{Z}_{\bar{q}}^{\bar{m}}$ for each $i \in [\ell]$, conduct the following steps.

1. Compute $\tau_f = \mathsf{AHF}_f(t_f) \in \{0, 1\}^L$ and parse it as $\tau_f = \tau_f[1] \ldots \tau_f[L]$.
2. Compute $\mathbf{B}(\tau_f) \in \mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}}$ as per (14).
3. Compute $\mathbf{d}_{t_f} = \sum_{i=1}^{\ell} \mathsf{dk}_{f,i} - \mathbf{B}(\tau_f)^\top \cdot \mathbf{t} \mod \bar{q}$, where $\mathbf{t} \in \mathbb{Z}^{\bar{n}}$ is taken from mpk.

4. Interpret $\mathbf{d}_{t_f} \in \mathbb{Z}_{\bar{q}}^{\bar{m}}$ as a vector of the form $\mathbf{d}_{t_f} = \bar{\mathbf{G}}^\top \cdot \mathbf{s}_{\mathbf{y}} + \tilde{\mathbf{e}}_f \mod \bar{q}$, for some error vector $\tilde{\mathbf{e}}_f \in [-\bar{B}, \bar{B}]^{\bar{m}}$. Using the public trapdoor of $\Lambda^\perp(\bar{\mathbf{G}})$, compute the underlying $\mathbf{s}_{\mathbf{y}} \in [-\ell \cdot \beta \cdot Y, \ell \cdot \beta \cdot Y]^n$.

Output the functional secret key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{s}_y)$.

$\textsf{\textbf{Encrypt}}(\mathsf{ek}_i, \boldsymbol{x}_i, t):$ Given $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$, $\boldsymbol{x}_i \in [-X, X]^{n_0}$, and $t \in \{0,1\}^{\ell_t}$,

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau[1] \ldots \tau[L]$.
2. Letting $\mathbf{W} = \mathbf{G}_0^\top \cdot \mathbf{V} \in \mathbb{Z}_q^{m \times n}$, compute

$$\mathbf{A}(\tau) = \mathbf{A}_{L,\tau[L]} \cdot \mathbf{G}^{-1}\Big(\mathbf{A}_{L-1,\tau[L-1]} \cdot \mathbf{G}^{-1}\big(\ldots \mathbf{A}_{2,\tau[2]} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{1,\tau[1]}\big)\big)\Big) \cdot \mathbf{G}^{-1}(\mathbf{W}^\top) \;\in \mathbb{Z}_q^{n \times m}. \tag{16}$$

3. Sample a noise vector $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$. Then, compute and output

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i \;\in \mathbb{Z}_q^m.$$

$\textsf{\textbf{Decrypt}}(\mathsf{dk}_{\boldsymbol{y}}, t, \mathbf{C}_t):$ On input of a functional secret key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{s}_{\boldsymbol{y}})$ for a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in [-Y, Y]^\ell$, a tag $t \in \{0,1\}^{\ell_t}$, and an ℓ-vector of ciphertexts $\mathbf{C}_t = (\mathbf{C}_{t,1}, \ldots, \mathbf{C}_{t,\ell}) \in (\mathbb{Z}_q^m)^\ell$, conduct the following steps.

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau[1] \ldots \tau[L]$.
2. Compute $\mathbf{A}(\tau) \in \mathbb{Z}_q^{n \times m}$ as per (16).
3. Compute $\mathbf{f}_{t,\boldsymbol{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{C}_{t,i} - \mathbf{A}(\tau)^\top \cdot \mathbf{s}_{\boldsymbol{y}} \mod q$.
4. Interpret $\mathbf{f}_{t,\boldsymbol{y}} \in \mathbb{Z}_q^m$ as a vector of the form $\mathbf{f}_{t,\boldsymbol{y}} = \mathbf{G}_0^\top \cdot \boldsymbol{z} + \tilde{\mathbf{e}} \mod q$, for some error vector $\tilde{\mathbf{e}} \in [-B, B]^m$. Using the public trapdoor of $\Lambda^\perp(\mathbf{G}_0)$, compute and output the underlying vector $\boldsymbol{z} \in [-\ell \cdot X \cdot Y, \ell \cdot X \cdot Y]^{n_0}$.

The scheme's correctness is implied by that of the two underlying centralized schemes. In turn, these are correct by Lemma 3.1 and the choice of parameters.

4.2 Security

The proof of Theorem 4.1 is given in the full version of the paper. In order to reduce the security of the centralized scheme to that of its decentralized variant, the proof first moves to a game where the partial functional key generation oracle of Definition 2.14 can be simulated using the functional key generation oracle of Definition 2.11. To this end, it relies on the security of the first MCFE instance. The next step is to move to a game where encryption queries $(i, \boldsymbol{x}_{i,0}, \boldsymbol{x}_{i,1}, t)$ are answered by returning encryptions of $\boldsymbol{x}_{i,1}$ instead of $\boldsymbol{x}_{i,0}$. To this end, we rely on the security of the second MCFE instance, which is possible since the partial key generation oracle can be simulated using the centralized key generation oracle. The final transition restores the partial key generation oracle of Definition 2.14 to its original output distribution. To this end, we invoke again the security of the first MCFE instance and reverse the transition of the first step.

Theorem 4.1. *The above DMCFE scheme provides* sta-IND-sec *security under the* LWE *assumption.*

Acknowledgements

Part of this work was funded by the French ANR ALAMBIC project (ANR-16-CE39-0006) and by BPI-France in the context of the national project RISQ (P141580). This work was also supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701).

References

1. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner product functional encryption. In *Asiacrypt*, 2019.
2. M. Abdalla, F. Benhamouda, M. Kolhweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC*, 2019.
3. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015.
4. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Crypto*, 2018.
5. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *Eurocrypt*, 2017.
6. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Eurocrypt*, 2010.
7. S. Agrawal, S. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.
8. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products from standard assumptions. In *Crypto*, 2016.
9. S. Agrawal and A. Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, 2017.
10. C. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Crypto*, 2017.
11. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudo-random functions. In *Crypto*, 2014.
12. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Eurocrypt*, 2012.
13. F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC*, 2017.
14. F. Benhamouda, M. Joye, and B. Libert. A framework for privacy-preserving aggregation of time-series data. *ACM Transactions on Information and System Security (ACM-TISSEC)*, 18(3), 2016.
15. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Crypto*, 2004.
16. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Eurocrypt*, 2004.
17. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Crypto*, 2001.
18. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key-homomorphic PRFs and their applications. In *Crypto*, 2013.
19. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

20. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.
21. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Crypto*, 2014.
22. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *Eurocrypt*, 2016.
23. T. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *FC*, 2012.
24. N. Chandran, V. Goyal, A. Jain, and A. Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive: Report 2015/1017.
25. M. Chase. Multi-authority attribute based encryption. In *TCC*, 2007.
26. M. Chase and S. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM-CCS*, 2009.
27. J. Chotard, E. Dufour Sans, R. Gay, D.-H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Asiacrypt*, 2018.
28. J. Chotard, E. Dufour Sans, R. Gay, D.-H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive: Report 2018/1021, 2018.
29. C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA International Conference on Cryptography and Coding*, 2001.
30. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *PKC*, 2018.
31. Y. Dodis. *Exposure-resilient cryptography.* PhD thesis, MIT, 2000.
32. A. Fiat and M. Naor. Broadcast encryption. In *Crypto*, 1993.
33. E. Freire, D. Hofheinz, K. Paterson, and C. Striecks. Programmable hash functions in the multilinear setting. In *Crypto*, 2013.
34. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
35. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
36. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Crypto*, 2013.
37. S. Goldwasser, S. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Eurocrypt*, 2014.
38. S. Goldwasser, V. Goyal, A. Jain, and A. Sahai. Multi-input functional encryption. Cryptology ePrint Archive: Report 2013/727, 2013.
39. S. Goldwasser, Y. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the Learning with Errors assumption. In *ICS*, 2010.
40. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run Turing machines on encrypted data. In *Crypto*, 2013.
41. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.
42. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Crypto*, 2012.
43. S. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive: Report 2013/774, 2014.
44. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM-CCS*, 2006.

45. G. Hanaoka, T. Matsuda, and J. Schuldt. On the impossibility of constructing efficient key encapsulation and programmable hash functions in prime order groups. In *Crypto*, 2012.
46. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *Crypto*, 2008.
47. T. Jager. Verifiable random functions from weaker assumptions. In *TCC*, 2015.
48. M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *FC*, 2013.
49. S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In *Asiacrypt*, 2016.
50. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Eurocrypt*, 2008.
51. A. Lewko, E. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Eurocrypt*, 2010.
52. A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *Eurocrypt*, 2011.
53. A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Crypto*, 2012.
54. B. Libert, A. Sakzad, D. Stehlé, and R. Steinfeld. All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In *Crypto*, 2017.
55. B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed PRFs from LWE. In *TCC*, 2018.
56. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *Crypto*, 2017.
57. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, 2012.
58. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Eurocrypt*, 1999.
59. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto*, 2010.
60. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
61. A. Sahai and H. Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *ACM-CCS*, 2010.
62. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Eurocrypt*, 2005.
63. C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, 1987.
64. E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.
65. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, 2011.
66. B. Waters. Functional encryption for regular languages. In *Crypto*, 2012.
67. H. Wee. Dual system encryption via predicate encoding. In *TCC*, 2014.