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

Benoît Libert, Radu Titiu

▶ To cite this version:

Benoît Libert, Radu Titiu. Multi-Client Functional Encryption for Linear Functions in the Standard Model from LWE. ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Dec 2019, Kobe, Japan. pp.1-54, 10.1007/978-3-030-34618-8_18 . hal-02352139

HAL Id: hal-02352139

https://inria.hal.science/hal-02352139

Submitted on 6 Nov 2019

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) [64,20] 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 [18,30], searchable encryption [17], attribute-based encryption [64,45], broadcast encryption [33] and many others.

As formalized by Boneh, Sahai and Waters [20], 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 [39,38] and multi-client [44,38] 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 [44,38], 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 [53,25], 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.* [28]. 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.* [28] 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.* [28] 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.* [28] 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 [28] 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 [44,38] 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 [28]), we construct the first MCFE scheme in the standard model and prove it fully secure under the Learning-With-Errors assumption [62] in the adaptive corruption setting (note that only static corruptions were considered in [44, 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 [28], 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 [28].

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.* [28] can be interpreted as relying on (a variant of) the key-homomorphic pseudorandom function [19] of Naor, Pinkas and Reingold [60]. Namely, the scheme of [28] 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 [28] 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 [47] 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.* [46] hint that such programmable hash functions are hardly instantiable in prime-order DDH groups. While the multi-linear setting [34] allows bypassing the impossibility results of [46], it is not known to enable standard assumptions.

A natural idea is to replace the random-oracle-based key-homomorphic PRF of [60] by an LWE-based key-homomorphic PRF [19,12]. However, analogously to Chotard *et al.* [28],[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 [56]. The latter can be seen as instantiating the programmable hash function of Freire *et al.* [34] 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 [37]. More precisely, the matrix $\mathbf{A}(x)$ is obtained as the product of GSW ciphertexts dictated by the output of an admissible hash function [16] applied to the PRF input. The security proof of [56] 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 [13], 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 [58] 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 [56] appeals to the lossy mode of LWE [40] 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 [56], except that we add a Gaussian noise instead of using the Learning-With-Rounding technique [13]. 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 [56] is non-trivial. We cannot rely on the DPRF of [56] 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 [56], partial evaluations are not proven pseudorandom: [56] 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 [56], 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

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

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 [55]). 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 $\mathbf{s}'_i = \mathbf{s}_i + \mathbf{T} \cdot (\boldsymbol{x}^\star_{0,i} - \boldsymbol{x}^\star_{1,i}) \in \mathbb{Z}^n$. 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 [69,14], the simulator can guess $\boldsymbol{x}^\star_{0,i} - \boldsymbol{x}^\star_{1,i}$ 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 $\boldsymbol{x}^\star_{1,i}$ since $\mathbf{C}^\star_i = \mathbf{G}_0^\top \cdot \boldsymbol{x}^\star_{1,i} + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}'_i + \mathsf{noise}$ is statistically close to $\mathbf{C}^\star_i = \mathbf{G}_0^\top \cdot \boldsymbol{x}^\star_{0,i} + \mathbf{A}(t^\star)^\top \cdot \mathbf{s}_i + \mathsf{noise}$.

We insist that our construction and proof are not merely obtained by plugging the DPRF of [56] into the high-level design principle of [28]. 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 [56], 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 [28] 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 [28], 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 [32] is thus needed in [56].

$\mathsf{ek}_i = (\mathbf{s}_i, \mathbf{t}_i)$ 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 [64], where they also constructed a scheme for threshold functions. Constructions of FE for point functions (known as identity-based encryption) [18,30] 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 [51,7], Boolean formulas [45,52,61,67,54], membership checking [21] and even finite state automaton [68]. Recently, the landscape of functional encryption improved considerably. Gorbunov *et al.* [43] and Garg *et al.* [35] provided the first constructions of attribute-based encryption for all circuits; Goldwasser *et al.* [42] constructed succinct simulation-secure single-key FE scheme for all circuits and also obtained FE for Turing machines [41]. In a breakthrough result, Garg *et al.* [35] designed indistinguishability-secure multi-key FE schemes for all circuits. However, while the constructions of [43,42] rely on standard assumptions, the assumptions underlying the other constructions [35,41] 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 [63,43]. Quite recently, the IPFE functionality [3,8] was extended into FE schemes supporting the evaluation of quadratic functions over encrypted data [57,11]. The schemes of [57,11] 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 [35] , or are restricted to bounded collusions [43,9].

Multi-Input and Multi-Client Functional Encryption. Goldwasser *et al.* [39,38] 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., [39,5,28]). 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.* [23] 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 [31,4]. 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 [39,38] 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 [28], 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 [44,38] and relies on the DDH assumption and on indistinguishability obfuscation to handle general circuits. The notion of aggregator-oblivious encryption (AOE) [66,24,49,15] 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 [66,49,15] rely on the random oracle model.

The first efficient MCFE scheme with multi-key security was described by Chotard *et al.* [28] 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) [26,27]. Lewko and Waters [53] gave the first ciphertext-policy ABE where users' attributes may be certified by completely independent authorities. Boneh and Zhandry [22] suggested distributed broadcast encryption systems, which dispense with the need for an authority handing out keys to registered users. Chandran *et al.* [25] considered decentralized general-purpose FE using obfuscation. The decentralization of multi-client FE was first considered by Chotard *et al.* [28] 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.* [29] 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 [28] 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 [29, Section 4.2] is implicitly based on the Boneh-Franklin IBE [18]. 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.* [29] also showed how to transform the ROM-based scheme from [28] 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, [2,29] 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 [2,29] implies MCFE with labels in the standard model from LWE (nor any standard assumption), even for the security definition of [28]. 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 [28]. In Appendix A, 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 $\|\mathbf{M}\| := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{Mx}\|}{\|\mathbf{x}\|}$ and $\|\mathbf{M}\|_\infty := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{Mx}\|_\infty}{\|\mathbf{x}\|_\infty}$. 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 $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ 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:*

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

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 [58] 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 ([58, 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 [58] 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 ([40, 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 ([36, 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 [16] as a combinatorial tool for partitioning-based security proofs for which Freire *et al.* [34] gave a simplified definition. Jager [48] considered the following generalization in order to simplify the analysis of reductions under decisional assumption.

Definition 2.6 ([48]). *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*

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

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

Theorem 2.7 ([48, 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 ([50, 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 [44].

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:*

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*.*

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* $C_{t,i}$*.*

DKeygen(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*.*

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 [44] 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 $\mathbf{Adv}^{\text{1Ch-IND}}(\mathcal{A}) := \left|\Pr[\beta = b] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda)$.*

In Appendix B, 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 Proposition B.2 in Appendix, 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.* [29] independently showed constructions that can be proven secure in the sense of a stronger definition which eliminates restriction 3 from the "Finalize" stage. In Appendix A, 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 [56]. Chotard *et al.* [29] 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.* [28] with the difference that we explicitly assume common public parameters cp. As in [28], we assume that each function f can be injectively encoded as a tag t_f (called "label" in [28]) 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.*

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 **(IND-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 sta-IND-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 [56], 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 [58]. 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 [56] 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 [56] 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 [56], 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 $\mathbf{U} = \left[\begin{smallmatrix}\mathbf{V}\\ \mathbf{C}\end{smallmatrix}\right]$ 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 [56] 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 [12]) in order for the noise matrix $\mathbf{R}_\tau$ to retain small entries.

We note that, as was suggested by [56, Section 4.3] in the context of distributed PRFs, a technique due to Yamada [70] can be used to compute $\mathbf{A}(\tau)$ as a function of $\Theta(\log^2 \lambda)$ (instead of $\Theta(\lambda)$)) public matrices. For simplicity, we chose to describe the scheme using public parameters containing a linear number of matrices here.

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 $\ell \le \ell_{max}$, 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.

[7] In [56], 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 $\mathbb{Z}_q$.

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

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 Appendix C.1.

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 [62] 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 [65] 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

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

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 Appendix C.2.)

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

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 [69,14] 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

$$\mathbf{U} = \left[\frac{\mathbf{V}}{\mathbf{C}}\right] \in \mathbb{Z}_q^{(n_0+n_1)\times n},$$

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 (20). 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 $|\Pr[W'_k] - \Pr[W_k]| \leq 2^{-\Omega(\lambda)}$.

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 $\mathbf{T}_{\mathbf{U}}$ 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 Appendix C.3.)

4 A DMCFE Scheme for Linear Functions

As in [28], 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 [28], 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 $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{s}_y)$.

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

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

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

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

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 Appendix D. It proceeds with a sequence of games where the first (resp. last) game is the real experiment of Definition 2.14 where the challenger's bit is $b = 0$ (resp. $b = 1$).

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 reduction has the particularity that it has to send dummy encryption queries to its challenger in order to comply with Condition 2 of the "finalize" step in Definition 2.11. 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 $\boldsymbol{x}_{i,0}$. For this purpose, 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. In order to make sure that the simulation does not break Condition 3 in the "finalize" step of Definition 2.11, the reduction only invokes its centralized key generation oracle in the very last partial key generation query involving an uncorrupted user. We note that a similar technique was used in [28], where keys were computed modulo a prime. Here, our reduction has to apply the same strategy over the integers. However, the technique still works as long as the message space of the first MCFE instance is large enough to contain functional secret keys $\sum_{i=1} y_i \cdot \mathbf{s}_i$ in each coordinate. 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* $\mathsf{sta\text{-}IND\text{-}sec}$ *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. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *Crypto*, 2013.
11. C. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Crypto*, 2017.

12. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudo-random functions. In *Crypto*, 2014.
13. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Eurocrypt*, 2012.
14. F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC*, 2017.
15. 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.
16. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Crypto*, 2004.
17. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Eurocrypt*, 2004.
18. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Crypto*, 2001.
19. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key-homomorphic PRFs and their applications. In *Crypto*, 2013.
20. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
21. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.
22. D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *Crypto*, 2014.
23. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *Eurocrypt*, 2016.
24. T. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *FC*, 2012.
25. N. Chandran, V. Goyal, A. Jain, and A. Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive: Report 2015/1017.
26. M. Chase. Multi-authority attribute based encryption. In *TCC*, 2007.
27. M. Chase and S. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM-CCS*, 2009.
28. J. Chotard, E. Dufour Sans, R. Gay, D.-H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Asiacrypt*, 2018.
29. 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.
30. C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA International Conference on Cryptography and Coding*, 2001.
31. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *PKC*, 2018.
32. Y. Dodis. *Exposure-resilient cryptography.* PhD thesis, MIT, 2000.
33. A. Fiat and M. Naor. Broadcast encryption. In *Crypto*, 1993.
34. E. Freire, D. Hofheinz, K. Paterson, and C. Striecks. Programmable hash functions in the multilinear setting. In *Crypto*, 2013.
35. 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.
36. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

37. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Crypto*, 2013.
38. 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.
39. S. Goldwasser, V. Goyal, A. Jain, and A. Sahai. Multi-input functional encryption. Cryptology ePrint Archive: Report 2013/727, 2013.
40. S. Goldwasser, Y. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the Learning with Errors assumption. In *ICS*, 2010.
41. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run Turing machines on encrypted data. In *Crypto*, 2013.
42. S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.
43. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Crypto*, 2012.
44. S. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive: Report 2013/774, 2014.
45. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM-CCS*, 2006.
46. 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.
47. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *Crypto*, 2008.
48. T. Jager. Verifiable random functions from weaker assumptions. In *TCC*, 2015.
49. M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *FC*, 2013.
50. S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In *Asiacrypt*, 2016.
51. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Eurocrypt*, 2008.
52. 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.
53. A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *Eurocrypt*, 2011.
54. A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Crypto*, 2012.
55. 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.
56. B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed PRFs from LWE. In *TCC*, 2018.
57. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *Crypto*, 2017.
58. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, 2012.
59. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput*, 37(1):267–302, 2007.
60. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Eurocrypt*, 1999.

61. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Crypto*, 2010.
62. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
63. A. Sahai and H. Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *ACM-CCS*, 2010.
64. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Eurocrypt*, 2005.
65. C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, 1987.
66. E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.
67. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, 2011.
68. B. Waters. Functional encryption for regular languages. In *Crypto*, 2012.
69. H. Wee. Dual system encryption via predicate encoding. In *TCC*, 2014.
70. S. Yamada. Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques. In *Crypto*, 2017.

A One-or-less MCFE Compiler

In the definition of 1-challenge IND-sec security, (i.e., Definition 2.12), the adversary has to make the challenge encryption queries $\mathsf{CQEncrypt}(j, x_j^{\star 0}, x_j^{\star 1}, t^\star)$ for all uncorrupted slots $j \in \mathcal{HS}$ as long as a query of the form $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ is made for some slot $i \in \mathcal{HS}$. Hence, it is not allowed to use the partial information that could be revealed by incomplete challenge queries.

It is desirable to strengthen the security definition and prove security even when the adversary is allowed to obtain partial ciphertexts by making challenge queries $\{\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)\}_{i \in I}$ for a *proper* subset $I \subset \mathcal{HS}$. In this section, we recall the definition of this security requirement and provide a compiler that takes a secure MCFE satysfying 1-challenge IND-sec security and upgrades it so as to achieve security in the sense of the stronger definition. The transformation relies on pseudorandom generators and *adaptively secure multi-instance PRFs*. A definition of the latter notion is given in this subsection.

Definition A.1 (1-or-less-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 answers subsequent challenge queries for the same $t^\star$. 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 one 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 $t^\star$ and an index $i \in [\ell]$*
3. *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 $\leq$ 1Ch-IND *security if, for any efficient adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\leq \mathsf{1Ch\text{-}IND}}(\mathcal{A}) := \left|\Pr[\beta = b] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda)$.*

The definition of 1-or-less IND security is identical to the 1Ch-IND definition, except that we removed Condition 3 of the Finalize step in Definition 2.12. In Definition A.1, we insist that the last condition of the Finalize step does not impose any restriction on the functions queried to $\mathsf{QDKeygen}$ if there exists a single index $i \in \mathcal{HS}$ for which no challenge query $\mathsf{CQEncrypt}(i, \cdot, \cdot, t^\star)$ was made.

In order to achieve security in the sense of Definition A.1, our compiler uses as a building block a pseudorandom function family that satisfies a definition of *adaptive multi-instance PRF*. This notion is defined as follows.

Definition A.2 (ad-mi-PRF). *An efficiently computable function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is an adaptively secure N-instance PRF, for some $N \in \mathsf{poly}(\lambda)$, if no PPT adversary $\mathcal{A}$ has non-negligible advantage in winning the following game.*

Initialization: *The challenger $\mathcal{C}$ samples N secret keys $k_1, k_2, \ldots k_N \hookleftarrow \mathcal{K}$ and a uniformly random bit $d \hookleftarrow U(\{0,1\})$.*

Queries. *The adversary $\mathcal{A}$ adaptively interleaves the following kinds queries:*

Evaluation: *Upon receiving a query* $\mathsf{Eval}(i, X)$*, where $i \in [N]$ and $X \in \mathcal{X}$, the challenger returns $\perp$ if it previously replied to a challenge query for the same pair (i, X). Otherwise, it replies with $F_{k_i}(X) \in \mathcal{Y}$.*

Corruption: *When the adversary makes a query* $\mathsf{Corrupt}(i)$*, the challenger returns $\perp$ if it previously replied to a challenge query for i. Otherwise, the challenger returns $k_i \in \mathcal{K}$.*

Challenge: *$\mathcal{A}$ makes challenge queries* $\mathsf{Challenge}(i^\star, X^\star)$ *for a unique arbitrary input $X^\star$ (any subsequent challenge query involving a different input $X \neq X^\star$ is ignored). If* $\mathsf{Eval}(i^\star, X^\star)$ *or* $\mathsf{Corrupt}(i^\star)$ *was queried before, the challenger returns $\perp$. Else, it returns $F_{k_{i^\star}}(X^\star)$ if $d = 1$ and a uniformly random value from $\mathcal{Y}$ if $d = 0$.*

Guess. *$\mathcal{A}$ outputs a guess $d' \in \{0,1\}$ and wins if $d' = d$.*

The adversary's advantage is $\mathbf{Adv}_{\mathcal{A}}^{\text{ad-mi-PRF}}(\lambda) := \left|\Pr[d' = d] - \frac{1}{2}\right|$.

It is possible to show that any PRF that provides single-instance security in the sense of a "find-then-guess" definition is also secure in the sense of Definition A.2. In short, the proof proceeds using a hybrid argument over the challenge queries and guesses the index of an uncorrupted index in each hybrid. However, this reduction incurs a quadratic security loss in N. Since Lemma A.4 uses a reduction with $N = \ell^2$ PRF instances, this would translate into a degradation factor ℓ^4, which would eventually become ℓ^5 because of the bound (17). In Appendix A.2, we show that a PRF construction described in [56] can be proven secure in the sense of Definition A.2 without a $O(N^2)$ loss in the reduction, so that our security loss is only linear in the number of slots ℓ. In comparison, the compiler of [1] loses a factor $O(\ell^2)$ but relies on any PRF.

A.1 The Transformation

The compiler is similar to the random-oracle-based transformation of Abdalla *et al.* [2, Section 4.2] – which is itself inspired by [4] – and its intuition is the following. The ciphertexts of individual slots contain partial MCFE ciphertexts $\mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t)$ which are super-encrypted using a symmetric encryption scheme with secret key $K_{t,i}$. The partial ciphertexts of the compiled scheme also contain shares k_{ijt} of the secret key $K_{i,t}$ in such a way that, once all the partial ciphertexts are gathered for a given tag t, the decryptor can recover the secret $K_{t,i} = \bigoplus_{j=1}^{\ell} k_{ijt}$ and thus the encryption $\mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, i)$. If only one such ciphertext is missing, then the secret key $K_{t,i}$ remains computationally hidden because the decryptor does not have all the shares. The symmetric encryption layer thus hides all the information that could leak when the adversary does not make encryption queries for all uncorrupted slots.

Given an MCFE scheme $\mathcal{MCFE}' = (\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{DKeygen}', \mathsf{Decrypt}')$, an adaptively secure ℓ^2-instances $\mathsf{ad\text{-}mi\text{-}PRF}$ $F : \mathcal{K} \times \mathcal{T} \to \{0,1\}^\lambda$ and a PRG $G : \{0,1\}^\lambda \to \{0,1\}^{|\mathsf{ct}'|}$, where $|\mathsf{ct}'|$ is the length of the output of the algorithm $\mathsf{Encrypt}'$, we obtain the following compiled scheme $\mathcal{MCFE}$.

$\mathsf{\textbf{Setup}}(\mathsf{cp}, 1^\ell)$: Runs $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}'(\mathsf{cp}, 1^\lambda)$. Samples ℓ^2 secret keys $k_{ij} \hookleftarrow \mathcal{K}$ for all $i, j \in [\ell]$. Then, set $\mathsf{mpk} := \mathsf{mpk}'$ $\mathsf{msk} := \mathsf{msk}'$ and $\mathsf{ek}_i := (\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j \in [\ell]})$

$\mathsf{\textbf{Encrypt}}(\mathsf{ek}_i, x_i, t)$: Given $\mathsf{ek}_i = (\mathsf{ek}'_i, \{k_{ij}, k'_{ji}\}_{j=1}^{\ell})$ it computes $k_{ijt} := F_{k_{ij}}(t)$, $k_{jit} := F_{k_{ji}}(t)$ for all $j \in [\ell]$, $C'_{t,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t)$ and $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Next, it computes $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$. It outputs $\mathsf{ct}_{t,i} = \left(C_{t,i}, \{k_{jit}\}_{j=1}^{\ell}\right)$.

DKeygen(msk, f) : Given msk and an ℓ-argument function $f : \mathcal{M}^\ell \to \mathcal{R}$. It outputs a functional decryption key $\mathsf{dk}_f := \mathsf{DKeygen}'(\mathsf{msk}', f)$.

Decrypt$(\mathsf{dk}_f, t, \{\mathsf{ct}_{t,1}, \mathsf{ct}_{t,2}, \ldots, \mathsf{ct}_{t,\ell}\})$: Takes as input a functional decryption key dk_f, a tag $t \in \mathcal{T}$, and an ℓ-vector of ciphertexts $\mathbf{C} = (\mathsf{ct}_{t,1}, \ldots, \mathsf{ct}_{t,\ell})$. For all $i \in [\ell]$ it computes $K_{t,i} = \bigoplus_{j=1}^{\ell} k_{ijt}$ and $C'_{t,i} := C_{t,i} \oplus G(K_{t,i})$ and runs $\mathsf{Decrypt}'(\mathsf{dk}_f, t, \{C'_{t,1}, C'_{t,2}, \ldots, C'_{t,\ell}\})$

The security proof relies on the idea that, if the adversary $\mathcal{A}$ does not make all the allowed challenge encryptions queries, there must exist an honest slot $j_0 \in \mathcal{HS}$ such that $\mathsf{CQEncrypt}(j_0, x_{j_0}^{0\star}, x_{j_0}^{1\star}, t^\star)$ is never asked. This ensures that none of the PRF values $\{k_{ij_0t^\star} = F_{k_{ij_0}}(t^\star)\}_{i \in \mathcal{HS}}$ is ever exposed to the adversary, which implies that $\{k_{ij_0t^\star}\}_{i \in \mathcal{HS}}$ are pseudorandom in the adversary's view since F is a PRF. This implies that secret keys $K_{t^\star,i} := \bigoplus_{j=1}^{\ell} k_{ijt^\star}$ are also pseudorandom for all $i \in \mathcal{HS}$. Hence, the ciphertexts $C_{t^\star,i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, i) \oplus G(K_{t^\star,i})$ computationally hide the x_i's for all uncorrupted slots i. To translate this intuition into a formal security proof, we need a PRF family that provides security in the sense of Definition A.2.

Theorem A.3. *If $\mathcal{MCFE}'$ is* 1Ch-IND *secure then the compiled $\mathcal{MCFE}$ scheme is* $\leq$ 1Ch-IND *secure.*

Proof. Given an adversary $\mathcal{A}$ that breaks the $\mathcal{MCFE}$ scheme by winning the game of Definition A.1 with non-negligible probability, we give a construction of an adversary $\mathcal{B}$ that wins with the game of Definition 2.12 with noticeable advantage, thus breaking the $\mathcal{MCFE}'$ scheme. The adversary $\mathcal{B}$ works as follows.

Initialization: The challenger $\mathcal{C}$ gives $\mathcal{B}$ the public parameters mpk' and keeps to itself the secret keys $(\mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell})$. In order to initialize the game with $\mathcal{A}$, the reduction $\mathcal{B}$ samples ℓ^2 secret keys $k_{ij} \hookleftarrow \mathcal{K}$ for all $i, j \in [\ell]$. Then gives $\mathcal{A}$ the public parameters $\mathsf{mpk} := \mathsf{mpk}'$.

Encryption queries: For each encryption query $\mathsf{QEncrypt}(i, x, t)$ made by $\mathcal{A}$, $\mathcal{B}$ sends to same query to its challenger and obtains $C'_{t,i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x, t)$. Then, $\mathcal{B}$ computes $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$, $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$, $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$ and gives $\mathsf{ct}_{t,i} := (C_{t,i}, \{k_{jit}\}_{j=1}^{\ell})$ to $\mathcal{A}$.

Challenge queries: If $\mathcal{A}$ makes a query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$, $\mathcal{B}$ relays it to its challenger and obtains $C'_{t^\star,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i^{\star b}, t^\star)$. Then, $\mathcal{B}$ computes $\{k_{ijt^\star} := F_{k_{ij}}(t^\star), k_{jit^\star} := F_{k_{ji}}(t^\star)\}_{j=1}^{\ell}$, $K_{t^\star i} := \bigoplus_{j=1}^{\ell} k_{ijt^\star}$, as well as $C_{t^\star,i} := C'_{t^\star,i} \oplus G(K_{t^\star i})$. The adversary $\mathcal{A}$ is given $\mathsf{ct}_{t^\star i} := (C_{t^\star i}, \{k_{jit^\star}\}_{j=1}^{\ell})$.

Functional decryption key queries: Each query $\mathsf{QDKeygen}(f)$ made by $\mathcal{A}$ is relayed by $\mathcal{B}$ to its challenger, which replies with dk'_f. Then, $\mathcal{B}$ gives $\mathsf{dk}_f := \mathsf{dk}'_f$ to $\mathcal{A}$.

Corruption queries: For each corruption query $\mathsf{QCorrupt}(i)$ that $\mathcal{A}$ makes, $\mathcal{B}$ replies with $\mathsf{ek}_i := (\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell})$, where ek'_i is the key given by its challenger in response to the same corruption query.

Finalize: When $\mathcal{A}$ outputs a result b', $\mathcal{B}$ distinguishes two situations.

- If $\mathcal{A}$ makes queries $\mathsf{CQEnc}(i, \star, \star, t^\star)$ for all slots $i \in \mathcal{HS}$ and the conditions of the Finalize step in Definition A.1 are all satisfied (i.e., $\mathcal{A}$'s output is not overwritten by a random bit β), $\mathcal{B}$ outputs the same bit $\tilde{b} := b'$ as $\mathcal{A}$.
- Otherwise, $\mathcal{B}$ outputs a random $\tilde{b} \hookleftarrow U(\{0,1\})$. We call this event abort.

Next, we analyze the probability of $\mathcal{B}$ winning the game.

$$\begin{aligned}
\Pr[\tilde{b} = b] &= \Pr[\tilde{b} = b | \mathsf{abort}] \cdot \Pr[\mathsf{abort}] + \Pr[\tilde{b} = b | \overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] \\
&= \frac{1}{2} \cdot \Pr[\mathsf{abort}] + \Pr[b' = b | \overline{\mathsf{abort}}] \cdot \Pr[\overline{\mathsf{abort}}] \\
&= \Pr[b' = b] + \left(\frac{1}{2} - \Pr[b' = b | \mathsf{abort}]\right) \cdot \Pr[\mathsf{abort}] \\
&= \Pr[b' = b] + \left(\frac{1}{2} - \Pr[b' = b | \mathsf{Ab}]\right) \cdot \Pr[\mathsf{Ab}]
\end{aligned}$$

where Ab is the event that $\mathcal{A}$ does not make encryption queries for all honest slots in the challenge phase and the conditions in the finalize step of Definition A.1 are all satisfied.

In order to finish the proof, we need to prove the following claim.

Claim. If F is a $\mathsf{ad\text{-}mi\text{-}PRF}$ and G a PRG, then $\left|\Pr[b' = b | \mathsf{Ab}] - \frac{1}{2}\right|$ is negligible.

Proof. We will prove this claim via a series of games. For each $i \in \{0, 1, 2\}$, we call W_i the event that the challenger outputs 1 in Game_i.

Game_0: This is the original game in the $\mathsf{one\text{-}or\text{-}less}$ security experiment (cf. Definition A.1) for the compiled scheme. The challenger runs the setup algorithm of MCFE' to obtain $(\mathsf{mpk}, \mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell})$, samples $k_{ij} \hookleftarrow \mathcal{K}$, for all $i, j \in [\ell]$ and sets $\mathsf{mpk} := \mathsf{mpk}'$, $\mathsf{msk} := \mathsf{msk}'$ and $\mathsf{ek}_i := (\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell})$. It gives mpk to $\mathcal{A}$ and handles queries $\mathsf{QCorrupt}(i)$ using $\mathsf{ek}_i := (\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell})$. To answer queries $\mathsf{QEnc}(i, x_i, t)$, it computes $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$ and $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Then, it computes $C_{t,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t) \oplus G(K_{t,i})$. The adversary $\mathcal{A}$ is given $\mathsf{ct}_{t,i} := (C_{t,i}, \{k_{jit}\}_{j=1}^{\ell})$. To answer challenge encryption queries $\mathsf{CQEncrypt}(i, x_i^0, x_i^1, t^\star)$, it replies with the result of $\mathsf{QEnc}(i, x_i^b, t^\star)$. As for functional decryption queries $\mathsf{QDKeygen}(f, \mathsf{msk})$, the challenger replies with $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}', f)$. When $\mathcal{A}$ halts, it outputs a bit b' and the challenger outputs 1 if and only if $b' = b$, so that $\Pr[W_0] = \Pr[b' = b | \mathsf{Ab}]$.

Game_1: The challenger interacts with $\mathcal{A}$ exactly as in Game_0 except that, upon receiving a challenge query $\mathsf{CQEncrypt}(i, x_i^0, x_i^1, t^\star)$, it does the following. If $x_i^0 = x_i^1$, it replies as in Game_0. Otherwise, for each slot $i \in [\ell]$ such that $x_i^0 \neq x_i^1$, it samples a uniform value for $K_{t^\star,i} \hookleftarrow U(\{0,1\}^\lambda)$ which it uses as a seed for the pseudorandom generator G in order to compute $C_{t^\star,i}$.

For any index i such that $x_i^0 \neq x_i^1$, corruption queries $\mathsf{Corrupt}(i)$ are disallowed by the conditions of the Finalize step which must be satisfied (recall that we are conditioning on event Ab). The only information that $\mathcal{A}$ can gather about $K_{t^\star,i}$ is thus obtained from $\mathsf{QEnc}(j, x_j, t)$ and $\mathsf{CQEnc}(j, x_j^0, x_j^1, t^\star)$ queries. By making these types of queries, $\mathcal{A}$ only learns the evaluations $\{k_{ijt} = F_{k_{ij}}(t)\}_{j\in[\ell]}$, for any $t \in \mathcal{T} \setminus \{t^\star\}$, and $\left\{F_{k_{ij}}(t^\star)\right\}_{j\in\mathcal{HS}\setminus\{j_0\}}$, respectively. Since we are conditioning on Ab, there exists an index $j_0 \in \mathcal{HS}$ which is never the input of a query of the form $\mathsf{CQEnc}(j_0, \cdot, \cdot, t^\star)$. In order to prove that Game_0 and Game_1 are computationally indistinguishable, we define Game_0' and Game_1' in the following way.

Game_b' ($b \in \{0, 1\}$): This game is identical to Game_b with the difference that the challenger initially chooses $j_0 \hookleftarrow U([\ell])$ as a guess for the smallest index $j_0 \in \mathcal{HS}$ such that no challenge query $\mathsf{CQEnc}(j_0, \cdot, \cdot, t^\star)$ is ever made. The challenger aborts and outputs 0 if the guess eventually turns out to be wrong. For each $b \in \{0, 1\}$, we have $\Pr[W_b] = \ell \cdot \Pr[W_b']$.

Lemma A.4 shows that Game_0' and Game_1' are computationally indistinguishable if F is a secure PRF in the multi-instance setting. This implies $|\Pr[W_0] - \Pr[W_1]| \le \ell \cdot \mathbf{Adv}_{\mathcal{B}}^{\text{ad-mi-PRF}}(\lambda)$.

Game_2: This game identical to Game_1 except that, at each challenge query $\mathsf{CQEncrypt}(i, x_i^0, x_i^1, t^\star)$ for which $x_i^0 \neq x_i^1$, it also sets $C_{t^\star,i}$ uniformly random. In this game, we have $\Pr[W_2] = \Pr[b' = b|\mathsf{Ab}] = \frac{1}{2}$.

To see that Game_1 and Game_2 are computationally indistinguishable, recall that $C_{t^\star,i} = C_{t^\star,i} \oplus G(K_{t^\star,i})$. When the seed $K_{t^\star,i}$ is uniformly generated the value $G(K_{t^\star,i})$ is pseudorandom, thus Game_1 and Game_2 are computationally indistinguishable. We thus obtain $|\Pr[W_2] - \Pr[W_1]| \le \mathbf{Adv}^{\text{PRG}}(\lambda)$.

When combining the above, we obtain

$$|\Pr[W_0] - \Pr[W_2]| \le \ell \cdot \mathbf{Adv}^{\text{ad-mi-PRF}}(\lambda) + \mathbf{Adv}^{\text{PRG}}(\lambda), \tag{17}$$

which proves the result. □

Lemma A.4. *If F is secure in the* ad-mi-PRF *sense, then* Game_0' *and* Game_1' *are computationally indistinguishable. More precisely, there exists a PRF adversary $\mathcal{B}$ such that $|\Pr[W_0'] - \Pr[W_1']| \le \mathbf{Adv}_{\mathcal{B}}^{\text{ad-mi-PRF}}(\lambda)$.*

Proof. Suppose that $\mathcal{A}$ is an adversary that is able to distinguish between the two games. We construct an adversary $\mathcal{B}$ that breaks the ad-mi-PRF security of F when the challenger samples $N = \ell^2$ secret keys $k_{ij} \hookleftarrow \mathcal{K}$ for all $i, j \in [\ell]$ in the game of Definition A.2. Algorithm $\mathcal{B}$ proceeds in the following way.

Initialization: $\mathcal{B}$ starts by drawing $j_0 \hookleftarrow U([\ell])$ as a guess that j_0 is the smallest index such that no challenge query $\mathsf{CQEncrypt}(j_0, \cdot, \cdot, t^\star)$ is made. Then, it runs $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}_i'\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{mpk} := \mathsf{mpk}'$ is given to $\mathcal{A}$.

Corruption queries: In order to answer a corruption query $\mathsf{Corrupt}(j)$, the reduction $\mathcal{B}$ aborts if $j = j_0$ since its guess for j_0 was incorrect. Otherwise, it obtains the values $\{k_{j\tau}, k_{\tau j}\}_{\tau=1}^{\ell}$ by making corruption queries to its PRF challenger. Then, it replies with $(\mathsf{ek}'_j, \{k_{j\tau}, k_{\tau j}\}_{\tau=1}^{\ell})$. Notice that, if j_0 is a correct guess, we have $j_0 \in \mathcal{HS}$, which implies that the adversary never learns $\{k_{j_0 i}, k_{ij_0}\}_{i \in \mathcal{HS}}$ from corruption queries.

Encryption queries: To answer $\mathsf{QEncrypt}(i, x, t)$ queries, $\mathcal{B}$ asks its PRF challenger for the evaluations $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$ and computes $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Then, it computes $C'_{t,i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x, t)$ as well as $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$. The ciphertext $\mathsf{ct}_{t,i} := (C_{t,i}, \{k_{jit}\}_{j=1}^{\ell})$ is given to $\mathcal{A}$.

Challenge queries: When receiving a challenge query $\mathsf{CQEncrypt}(i, x_i^{0\star}, x_i^{1\star}, t^\star)$, $\mathcal{B}$ aborts if $i = j_0$. Otherwise, it considers the following two cases.

- If $x_i^{0\star} = x_i^{1\star} = x_i^\star$, then $\mathcal{B}$ responds as it would answer the encryption query $\mathsf{QEncrypt}(i, x_i^\star, t^\star)$.
- If $x_i^{0\star} \neq x_i^{1\star}$ (which is only possible if $i \in \mathcal{HS}$), the reduction $\mathcal{B}$ asks its challenger for the PRF evaluations $\{k_{ijt^\star} := F_{k_{ij}}(t^\star)\}_{j \neq j_0}$. It also makes a challenge query for the input $t^\star$ and the secret key k_{ij_0}. The challenger replies with a value $y_{ij_0t^\star}$, which is either the evaluation $F_{k_{ij_0}}(t^\star)$ for the unknown secret key k_{ij_0} chosen by the PRF challenger or a random value. Then, $\mathcal{B}$ computes $K_{t^\star,i} := (\bigoplus_{j \neq j_0} k_{ijt^\star}) \oplus y_{ij_0t^\star}$ and uses $K_{t^\star,i}$ as a seed for the PRG, by computing $C^b_{t^\star,i} := C^{b\prime}_{t^\star,i} \oplus G(K_{t^\star,i})$. The adversary $\mathcal{A}$ is given $\mathsf{ct}_{t,i} := (C^b_{t,i}, \{k_{jit^\star}\}_{j=1}^{\ell})$.

Guess: When $\mathcal{A}$ halts, it outputs a bit b'. At this point, $\mathcal{B}$ aborts and outputs 0 if j_0 is not the smallest $i \in [\ell]$ such that no query $\mathsf{CQEncrypt}(i, \cdot, \cdot, t^\star)$ is made. Otherwise, $\mathcal{B}$ outputs 1 (meaning that its challenger always outputs pseudorandom values) if $b' = b$ and 0 otherwise.

If j_0 is not a correct guess, Game'_0 and Game'_1 have the same outcome since the challenger outputs 0 in both games. We thus assume that j_0 is a correct guess, in which case we know that $j_0 \in \mathcal{HS}$. Moreover, if $x_i^{0\star} \neq x_i^{1\star}$, we necessarily have $i \in \mathcal{HS}$. Therefore the reduction $\mathcal{B}$ never obtains the secret key k_{ij_0} by making a $\mathsf{Corrupt}(\cdot)$ query in the game of Definition A.2. In the same game, we also observe that the value $F_{k_{ij_0}}(t^\star)$ is never asked by $\mathcal{B}$ to its PRF evaluation oracle. Indeed, encryption queries $\mathsf{QEncrypt}(\cdot, \cdot, t^\star)$ are not allowed on the challenge tag $t^\star$ and the constraint Ab implies that no challenge query $\mathsf{CQEncrypt}(j_0, \cdot, \cdot, t^\star)$ can be made for the index j_0.

We now observe that, if $\mathcal{B}$'s challenger always returns truly random values $y_{ij_0t^\star}$ at each challenge query, then $\mathcal{A}$'s view is identical to that of Game_1. If $\mathcal{B}$'s challenger always returns pseudorandom values, $\mathcal{A}$'s view is as in Game_0. We thus conclude that $\mathbf{Adv}_{\mathcal{B}}^{\text{ad-mi-PRF}}(\lambda) \geq |\Pr[W'_0] - \Pr[W'_1]|$. $\square$

A.2 A PRF Construction With Multi-Instance Security in the Adaptive Corruption Setting

We show that the centralized version of the distributed PRF of Libert *et al.* [56] provides security in the sense of Definition A.2. We first recall the construction.

Let λ be a security parameter and let $\ell \in \Theta(\lambda)$, $L \in \Theta(\lambda)$. The scheme uses public parameters consisting of prime moduli p and q such that $q/p > 2^{L+\lambda} \cdot N \cdot r$, dimensions $n, m, k \in \mathsf{poly}(\lambda)$ such that $m \geq 2n \cdot \lceil \log q \rceil$, an integer $\beta > 0$, a real $\alpha > 0$ and a rounding parameter $r = m^{L+2} \cdot n \cdot \beta \cdot \alpha q$. The PRF family relies on the following building blocks.

- A balanced admissible hash function $\mathsf{AHF} : \{0,1\}^{\ell} \to \{0,1\}^{L}$.
- A family Π_λ of ξ-wise independent hash functions $\pi_i : \mathbb{Z}_p^m \to \mathbb{Z}_p^k$ for a suitable $\xi > 0$ that will be determined later on. Let a random member π of Π_λ.

Let a Gaussian parameter $\sigma > 0$ such that the interval $[-\beta, \beta] = [-\sigma\sqrt{n}, \sigma\sqrt{n}]$ contains the coordinates of the secret key with overwhelming probability. The PRF family assumes the availability of public parameters

$$\mathsf{pp} := \Big(q,\ \pi,\ \mathbf{A}_0,\ \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\ \in \mathbb{Z}_q^{n \times m}\ \}_{i=1}^{L},\ \mathsf{AHF},\ r,\ \sigma\Big),$$

where $\mathbf{A}_0 \sim U(\mathbb{Z}_q^{n \times m})$ and $\mathbf{A}_{i,0}, \mathbf{A}_{i,1} \sim U(\mathbb{Z}_q^{n \times m})$ for each $i \in [L]$. Importantly, $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$ should be chosen in such a way that $\mathbf{G}^{-1}(\mathbf{A}_{i,b}) \in \mathbb{Z}^{m \times m}$ is $\mathbb{Z}_q$-invertible for all $i \in [L]$ and $b \in \{0,1\}$ (see [56, Section 3.1] for details).

Keygen(pp)**:** Given pp, sample a vector $\mathbf{s} \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ so that $\|\mathbf{s}\|_\infty < \beta = \sigma\sqrt{n}$ with overwhelming probability. The secret key is $SK := \mathbf{s} \in [-\beta, \beta]^n$.

Eval(pp$, SK, X$)**:** Given $SK = \mathbf{s} \in \mathbb{Z}^n$ and an input $X \in \{0,1\}^{\ell}$,

1. Compute $x = \mathsf{AHF}(X) \in \{0,1\}^{L}$ and parse it as $x = x_1 \dots x_L$.
2. Compute $\mathbf{A}(x) = \mathbf{A}_0 \cdot \prod_{i=1}^{L} \mathbf{G}^{-1}\big(\mathbf{A}_{i,x_i}\big)$ and

$$\mathbf{z} = \Big\lfloor \big(\mathbf{A}(x)\big)^{\top} \cdot \mathbf{s} \Big\rfloor_p \ \in \mathbb{Z}_p^m, \tag{18}$$

and output $\mathbf{y} = \pi(\mathbf{z}) \in \mathbb{Z}_p^k$.

We now prove that this construction is secure in the sense of Definition A.2 under the LWE assumption. The proof of Theorem A.5 is essentially identical to that of [56, Theorem 3.2]. Intuitively, it exploits the fact that the reduction knows the secret key at any time, which makes it easy to consistently answer adaptive corruption queries.

Theorem A.5. *The above construction provides adaptive multi-instance security under the* LWE *assumption.*

The proof relies on the following lemmas.

Lemma A.6 (Adapted from [59, Lemma 4.4]). *For any n-dimensional lattice Λ, $\mathbf{x}', \mathbf{c} \in \mathbb{R}^n$ and symmetric positive definite $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ satisfying $\sigma_n(\sqrt{\mathbf{\Sigma}}) \geq \eta_{2^{-n}}(\Lambda)$, we have*

$$\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda + \mathbf{x}') \ \in\ [1 - 2^{-n}, 1 + 2^{-n}] \cdot \det(\mathbf{\Sigma})^{1/2} / \det(\Lambda).$$

Lemma A.7 ([36, Lemma 5.3]). *Let $m \geq 2n \cdot \log q$ and $q \geq 2$ prime and let $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$. With probability $\geq 1 - 2^{-\Omega(n)}$, we have $\lambda_1^{\infty}(\Lambda(\mathbf{A})) \geq q/4$.*

Lemma A.8 ([32, Corollary 3]). *Fix any integers $\bar{n}$, m, M, any real $\varepsilon < 1$ and any collection $\mathcal{X}$ of M distributions over $\{0,1\}^{\bar{m}}$ of min-entropy $\bar{n}$ each. Define*

$$\xi = \bar{n} + \log M, \qquad \bar{k} = \bar{n} - \Big(2\log\frac{1}{\varepsilon} + \log\log M + \log\bar{n} + O(1)\Big),$$

and let $\mathcal{F}$ be any family of ξ-wise independent functions from $\bar{m}$ bits to $\bar{k}$ bits. With probability at least $(1 - 1/M)$, a random function $f \hookleftarrow U(\mathcal{F})$ is a good deterministic extractor for the collection $\mathcal{X}$. Namely, the distribution $f(X)$ is ε-close to $U(\{0,1\}^{\bar{k}})$ for any distribution $X \in \mathcal{X}$.

Lemma A.9. *Let $\mathcal{F}$ be a family of δ-sure ϵ-extractor for the finite collection $\mathcal{X}$ of distributions over the input space of $\mathcal{F}$: namely, $\mathcal{F}$ satisfies*

$$\Pr_{f \hookleftarrow F}[\forall X \in \mathcal{X} : \Delta(f(X), U) \leq \epsilon] \geq 1 - \delta,$$

where F is the uniform distribution over $\mathcal{F}$ and U is the uniform distribution over the range of $\mathcal{F}$. For any independent distributions $Z_1, Z_2, \ldots, Z_\ell \in \mathcal{X}$, we have $\Delta\left((F, F(Z_1), F(Z_2), \ldots, F(Z_\ell)), (F, U_1, U_2, \ldots U_\ell)\right) \leq \ell \cdot \epsilon + \delta$.

Proof. For any function $f \in \mathcal{F}$, the distributions $f(Z_1), f(Z_2), \ldots, f(Z_\ell)$ are independent since the initial distributions $Z_1, Z_2, \ldots, Z_\ell$ are independent. This implies that for, any function $f \in \mathcal{F}$, we have

$$\Delta(f(Z_1), f(Z_2), \ldots, f(Z_\ell), (U_1, U_2, \ldots, U_\ell)) \leq \sum_{j=1}^{\ell} \Delta(f(Z_j), U_j).$$

Let $U_{1\ell} := (U_1, U_2, \ldots, U_\ell)$. For any $f \in \mathcal{F}$, we also define the distribution $Z_f := (f(Z_1), f(Z_2), \ldots, f(Z_\ell))$. Since $\mathcal{F}$ is a δ-sure ϵ-extractor for the collection $\mathcal{X}$, it follows that

$$\Pr_{f \hookleftarrow \mathcal{F}}[\Delta(Z_f, U_{1\ell}) \leq \ell \cdot \epsilon] \geq 1 - \delta.$$

By using the identity $\Delta\left((F, Z_F), (F, U_{1\ell})\right) = \mathsf{Exp}_{f \hookleftarrow F}\left[\Delta(Z_f, U_{1\ell})\right]$, we obtain that $\Delta\left((F, Z_F), (F, U_{1\ell})\right) \leq \ell\epsilon \cdot (1 - \delta) + \delta < \ell \cdot \epsilon + \delta$. $\square$

Lemma A.10 ([10, Lemma 2.7]). *Let p, q be positive integers such that $p < q$. Given $R > 0$ an integer, the probability that there exists $e \in [-R, R]$ such that $\lfloor y \rceil_p \neq \lfloor y + e \rceil_p$, when $y \hookleftarrow U(\mathbb{Z}_q)$, is smaller than $\frac{2Rp}{q}$.*

Proof (of Theorem A.5). The proof considers a sequence of hybrid games. In each game, we call W_i the event that $b' = b$.

Game$_0$: This is the real experiment of Definition A.2, where the challenger initially samples N secret keys $SK_j = \mathbf{s}_j \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ which are revealed to the adversary $\mathcal{A}$ in case of corruption queries. At each evaluation query (j, X), the challenger replies by returning

$$\mathbf{y}_j = \pi\Big(\big\lfloor (\mathbf{A}(x))^\top \cdot \mathbf{s}_j \big\rfloor_p \Big) \ \in \mathbb{Z}_p^k,$$

where $\mathbf{A}(x) = \mathbf{A}_0 \cdot \prod_{i=1}^L \mathbf{G}^{-1}\big(\mathbf{A}_{i,x_i}\big)$. When $\mathcal{A}$ halts, it outputs $\hat{d} \in \{0,1\}$ and the challenger defines $d' := \hat{d}$. By definition, the adversary's advantage is $\mathbf{Adv}(\mathcal{A}) := |\Pr[W_0] - 1/2|$, where W_0 is event that $d' = d$.

Game$_1$: This game is like Game$_0$ with some 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 $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$. When the adversary eventually outputs $\hat{d} \in \{0,1\}$, the challenger checks if the conditions

$$P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0 \tag{19}$$

are satisfied, where $X^\star$ is the challenge input and $X^{(1)}, \ldots, X^{(Q)}$ are the evaluation queries. If these conditions do not hold, the challenger ignores $\mathcal{A}$'s output $\hat{d} \in \{0,1\}$ and replaces it by a random bit $d'' \hookleftarrow \{0,1\}$ to define $d' = d''$. If the conditions of (19) are satisfied, the challenger sets $d' = \hat{d}$. By Lemma 2.8, we have

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

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

Game$_2$: In this game, we modify the generation of pp. Initially, the challenger samples a uniformly random matrix $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$. 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 $b \in \{0,1\}$:

$$\mathbf{A}_{i,b} := \begin{cases} \mathbf{A} \cdot \mathbf{R}_{i,b} & \text{if} \quad (b \neq K_i) \ \wedge \ (K_i \neq \perp) \\ \mathbf{A} \cdot \mathbf{R}_{i,b} + \mathbf{G} & \text{if} \quad (b = K_i) \ \vee \ (K_i = \perp) \end{cases} \tag{20}$$

It also defines $\mathbf{A}_0 = \mathbf{A} \cdot \mathbf{R}_0 + \mathbf{G}$ for a random $\mathbf{R}_0 \hookleftarrow U(\{-1,1\}^{m \times m})$. Since $\mathbf{A} \sim U(\mathbb{Z}_q^{n \times m})$, the Leftover Hash Lemma implies that $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^L$ are statistically independent and uniformly distributed over $\mathbb{Z}_q^{n \times m}$. The challenger keeps sampling $\mathbf{R}_{ib}$ until $\mathbf{G}^{-1}(\mathbf{A}_{ib})$ is $\mathbb{Z}_q$-invertible in such a way that the simulated $\mathbf{A}_{ib}$ are statistically uniform under the constraint that $\mathbf{G}^{-1}(\mathbf{A}_{ib})$ be invertible over $\mathbb{Z}_q$. The analysis of [56] shows that the probability that $\mathbf{G}^{-1}(\mathbf{A}_{ib})$ is invertible in $\mathbb{Z}_q$ is roughly $\Big(\mathsf{exp}(-3\sqrt{\mathsf{exp}(1)})\Big)^2$, so that the public parameters can be simulated in polynomial time. Since the distribution of pp is statistically unchanged, it follows that $|\Pr[W_2] - \Pr[W_1]| \leq L \cdot 2^{-\lambda}$.

At each query X, we can view $\mathbf{A}(x)$ as a GSW encryption

$$\mathbf{A}(x) = \mathbf{A} \cdot \mathbf{R}_x + (\prod_{i=1}^{L} \mu_i) \cdot \mathbf{G},$$

for some small norm $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$, where

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

If the conditions (19) are satisfied, at each evaluation query $(j, X^{(i)})$, the admissible hash function ensures that $x^{(i)} = \mathsf{AHF}(X^{(i)})$ satisfies

$$\mathbf{A}(x^{(i)}) = \mathbf{A} \cdot \mathbf{R}_{x^{(i)}}, \tag{21}$$

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

$$\mathbf{A}(x^\star) = \mathbf{A} \cdot \mathbf{R}_{x^\star} + \mathbf{G}. \tag{22}$$

Game_3: We modify the distribution of pp and replace the uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ by a lossy matrix of the form

$$\mathbf{A}^\top = \bar{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E} \ \in \mathbb{Z}_q^{m \times n}, \tag{23}$$

where $\bar{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n' \times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n' \times n})$ and $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m \times n}, \alpha q}$, for $n' \ll n$. Under the LWE assumption in dimension n', this change should not affect $\mathcal{A}$'s output distribution and we can easily build an LWE distinguisher $\mathcal{B}$ shows that $|\Pr[W_3] - \Pr[W_2]| \leq n \cdot \mathbf{Adv}_{\mathcal{B}}^{\mathsf{LWE}_{q,m,n',\alpha}}(\lambda)$, where the factor n comes from the use of an LWE assumption with n secrets.

Due to the modification introduced in Game_3, if the conditions (19) are satisfied, each evaluation query $(j, X^{(i)})$ obtains a response of the form

$$\left\lfloor \left(\mathbf{A} \cdot \mathbf{R}_{x^{(i)}}\right)^\top \cdot \mathbf{s}_j \right\rfloor_p = \left\lfloor \left(\mathbf{R}_{x^{(i)}}^\top \cdot \bar{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{R}_{x^{(i)}}^\top \cdot \mathbf{E}\right) \cdot \mathbf{s}_j \right\rfloor_p. \tag{24}$$

Note that the right-hand-side member of (24) uniquely determines $\mathbf{C} \cdot \mathbf{s}_j$ with high probability: observe that $\mathbf{R}_{x^{(i)}}^\top \cdot \bar{\mathbf{A}}^\top$ is statistically uniform over $\mathbb{Z}_q^{m \times n'}$, so by Lemma A.7, the quantity $\lfloor \mathbf{R}_{x^{(i)}}^\top \cdot \bar{\mathbf{A}}^\top \cdot (\mathbf{C} \cdot \mathbf{s}_j) \rfloor_p$ is an injective function of $\mathbf{C} \cdot \boldsymbol{s} \bmod q$. It comes that evaluation queries (j, X) information-theoretically reveal $\mathbf{C} \cdot \mathbf{s}_j \bmod q$.

Game_4: We modify the evaluation oracle and introduce a bad event. We define bad_x to be the event that for the input $x \in \{0,1\}^L$ such that $\mathbf{A}(x) = \mathbf{A} \cdot \mathbf{R}_x$, for some small-norm $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$, we have

$$\left\lfloor \left(\mathbf{A} \cdot \mathbf{R}_x\right)^\top \cdot \mathbf{s}_j \right\rfloor_p \neq \left\lfloor \left(\mathbf{R}_x^\top \cdot \bar{\mathbf{A}}^\top \cdot \mathbf{C}\right) \cdot \mathbf{s}_j \right\rfloor_p. \tag{25}$$

for some $j \in [N]$. We also define BAD as the event that $\mathcal{A}$ makes an evaluation query (j, X) such that the event bad_x occurs. Note that the challenger can detect bad_x since it knows $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n' \times m}$, $\mathbf{C} \in \mathbb{Z}_q^{n' \times n}$ and $\mathbf{E} \in \mathbb{Z}^{m \times n}$ satisfying (23). If bad_x occurs, the challenger replaces $\mathcal{A}$'s output $\hat{d}$ by a random $d'' \hookleftarrow \{0,1\}$ and sets $d' = d''$. Otherwise, it sets $d' = \hat{d}$ as before.

The proof of the following claim is adapted from [56, Appendix D].

Claim. If $q/p > 2^{L+\lambda} \cdot N \cdot r$, where $r = m^{L+2} \cdot n \cdot \beta \cdot \alpha q$, we have the inequality

$$|\Pr[W_4] - \Pr[W_3]| \leq \Pr[\mathsf{BAD}] \leq 2^{-\Omega(\lambda)}.$$

Proof. We use the fact that, for each query $x = x_1 \ldots x_L$, the matrix $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$ is of the form

$$\begin{aligned}
\mathbf{R}_x = \mathbf{R}_0 \cdot \prod_{i=1}^{L} & \mathbf{G}^{-1}\big(\mathbf{A}_{i,x_i}\big) \\
& + \mathbf{R}_{1,x_1} \cdot \prod_{i=2}^{L} \mathbf{G}^{-1}(\mathbf{A}_{i,x_i}) \\
& + \mu_{1,x_1} \cdot \mathbf{R}_{2,x_2} \cdot \prod_{i=3}^{L} \mathbf{G}^{-1}\big(\mathbf{A}_{i,x_i}\big) \\
& \vdots \\
& + \mu_{1,x_1}\mu_{2,x_2} \cdots \mu_{L-2,x_{L-2}} \cdot \mathbf{R}_{L-1,x_{L-1}} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L,x_L}\big) \\
& + \mu_{1,x_1}\mu_{2,x_2} \cdots \mu_{L-1,x_{L-1}} \cdot \mathbf{R}_{L,x_L}.
\end{aligned}$$

In order to bound $\Pr[\mathsf{bad}_x]$, we note that, as argued in [10, Proof of Theorem 7.3], $\mathbf{R}_0^\top \cdot \bar{\mathbf{A}}^\top$ is statistically uniform over $\mathbb{Z}_q^{m \times n'}$, conditionally on $\mathbf{R}_0^\top \cdot \mathbf{E} \in \mathbb{Z}^{m \times n}$. Hence, for a fixed choice of $j \in [N]$ and $x \in \{0,1\}^L$, the product $\mathbf{R}_x^\top \cdot \bar{\mathbf{A}}^\top \cdot (\mathbf{C} \cdot \mathbf{s}_j)$ contains a term

$$\underbrace{\Big(\prod_{i=1}^{L} \mathbf{G}^{-1}(\mathbf{A}_{i,x_i})\Big)^\top}_{\mathbf{G}_x^\top} \cdot \mathbf{R}_0^\top \cdot \bar{\mathbf{A}}^\top \cdot (\mathbf{C} \cdot \mathbf{s}_j),$$

for which we know that $\mathbf{G}_x^\top \cdot \mathbf{R}_0^\top \cdot \bar{\mathbf{A}}^\top$ is statistically uniform over $\mathbb{Z}_q^{m \times n'}$ since $\mathbf{G}_x^\top$ is a $\mathbb{Z}_q$-invertible matrix by construction. We know that $\mathbf{G}_x^\top \cdot \mathbf{R}_0^\top \cdot \bar{\mathbf{A}}^\top \cdot (\mathbf{C} \cdot \mathbf{s}_j)$ is statistically close to the uniform distribution over $\mathbb{Z}_q^m$. We thus have

$$\Pr[\mathsf{bad}_x] = \Pr\left[\Big\lfloor \big(\mathbf{A} \cdot \mathbf{R}_x\big)^\top \cdot \mathbf{s}_j \Big\rfloor_p \neq \big\lfloor \big(\mathbf{R}_x^\top \cdot \bar{\mathbf{A}}^\top \cdot \mathbf{C}\big) \cdot \mathbf{s}_j \big\rfloor_p \right] \leq m \cdot \frac{2rp}{q}. \quad (26)$$

where $r = m^{L+2} \cdot n \cdot \beta \cdot \alpha q$. Indeed, we have $\|\mathbf{R}_x^\top\|_\infty \leq \frac{m(m^{L+1}-1)}{m-1}$, $\|\mathbf{s}_j\|_\infty \leq \beta$ and

$$\|\mathbf{E}\|_\infty = \max_{i \in [m]} \left(\sum_{j=1}^{n} |e_{i,j}| \right) \leq \sqrt{n} \max_{i \in [m]} \|\mathbf{e}_i\| \leq n \cdot \alpha q.$$

The conditions of Lemma A.10 are satisfied since

$$\|\mathbf{R}_x^\top \cdot \mathbf{E} \cdot \mathbf{s}_j\|_\infty \le \|\mathbf{R}_x^\top\|_\infty \cdot \|\mathbf{E}\|_\infty \cdot \|\mathbf{s}_j\|_\infty \le \frac{m(m^{L+1}-1)}{m-1} \cdot n \cdot \alpha q \cdot \beta \le r,$$

which yields inequality (26).

By taking the union bound over all possible inputs $x \in \{0,1\}^L$ and all indexes $j \in [N]$, we can bound the probability $\Pr[\mathsf{BAD}] \le N \cdot 2^L \cdot m \cdot \frac{2rp}{q}$.

Since Game_3 and Game_4 are identical if BAD does not occur, we obtain the inequality

$$|\Pr[W_4] - \Pr[W_3]| \le \Pr[\mathsf{BAD}] \le 2^L \cdot N \cdot m \cdot \frac{2rp}{q} = 2m \cdot 2^{-\lambda}$$

which is negligible in λ. □

Recall that, if BAD does not occur, we have

$$\left\lfloor \left(\mathbf{A} \cdot \mathbf{R}_{x^{(i)}}\right)^\top \cdot \mathbf{s}_j \right\rfloor_p = \left\lfloor \left(\mathbf{R}_{x^{(i)}}^\top \cdot \bar{\mathbf{A}}^\top \cdot \mathbf{C}\right) \cdot \mathbf{s}_j \right\rfloor_p . \tag{27}$$

at each query $(j, X^{(i)})$. We note that the right-hand-side member of (27) is completely determined by $\mathbf{R}_{x^{(i)}}^\top \cdot \bar{\mathbf{A}}^\top$ and the product $\mathbf{C} \cdot \mathbf{s}_j \in \mathbb{Z}_q^{n'}$. This means that, for each uncorrupted index $j \in [N]$, $\mathcal{A}$'s evaluation queries (j, X) always reveal the same $n' \cdot \log q$ bits of information about $\mathbf{s}_j \in \mathbb{Z}^n$. Consequently, each uncorrupted $SK_j = \mathbf{s}_j \in \mathbb{Z}_n$ retains a lot of entropy after all evaluation queries.

Game_5: We modify the challenge oracle. If $d = 1$, at each challenge query (j, X), the adversary obtains a fresh random $\mathbf{y}_j^\star \hookleftarrow U(\mathbb{Z}_p^k)$ instead of the real evaluations $\mathbf{y} = \pi\left(\left\lfloor (\mathbf{A}(x))^\top \cdot \mathbf{s}_j \right\rfloor_p\right)$. Clearly, we have $\Pr[W_5] = 1/2$ since the output distribution of the challenge oracle does not depend on $d \in \{0,1\}$. In Game$_4$, if $d = 1$, each challenge query $(j, X^\star)$ obtains a response

$$\mathbf{y}_j^\star = \pi\Big(\left\lfloor \left(\mathbf{A}(x^\star)\right)^\top \cdot \mathbf{s}_j \right\rfloor_p\Big) = \pi\Big(\left\lfloor \left(\mathbf{A} \cdot \mathbf{R}_{x^\star} + \mathbf{G}\right)^\top \cdot \mathbf{s}_j \right\rfloor_p\Big). \tag{28}$$

The same arguments as in [56, Appendix D] show that the right-hand-side member of (28) is statistically close to $U(\mathbb{Z}_p^k)$. Indeed, $\mathbf{z}_j^\star = \left\lfloor (\mathbf{A} \cdot \mathbf{R}_{x^\star} + \mathbf{G})^\top \cdot \mathbf{s}_j \right\rfloor_p$ depends on a term $\mathbf{G}^\top \cdot \mathbf{s}_j \in \mathbb{Z}_q^m$, which is an injective function of $\mathbf{s}_j$. More precisely, if BAD does not occur and $SK_j = \mathbf{s}_j$ has not been corrupted, $\mathcal{A}$ obtains at most $n' \cdot \log q$ bits of information about $\mathbf{s}_j$.

We also note that $\mathbf{z}_j^\star$ can be written

$$\mathbf{z}_j^\star = \left\lfloor \left(\mathbf{A} \cdot \mathbf{R}_{x^\star} + \mathbf{G}\right)^\top \cdot \mathbf{s}_j \right\rfloor_p = \left\lfloor \mathbf{R}_{x^\star}^\top \cdot \mathbf{A}^\top \cdot \mathbf{s}_j \right\rfloor_p + \left\lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \right\rfloor_p + \mathbf{e}_{s,j,x^\star}, \tag{29}$$

for some $\mathbf{e}_{s,j,x^\star} \in \{0,1\}^m$. Moreover, the proof of the claim implies that

$$\left\lfloor \mathbf{R}_{x^\star}^\top \cdot \mathbf{A}^\top \cdot \mathbf{s}_j \right\rfloor_p = \left\lfloor \mathbf{R}_{x^\star}^\top \cdot \bar{\mathbf{A}}^\top \cdot \mathbf{C} \cdot \mathbf{s}_j \right\rfloor_p$$

since it considers a union bound to show that equality (25) holds for all $x \in \{0,1\}^L$ and $j \in [N]$ with overwhelming probability. This implies $H_\infty\big(\lfloor \mathbf{R}_{x^\star}^\top \cdot \mathbf{A}^\top \cdot \mathbf{s}_j \rfloor_p \mid \mathbf{C} \cdot \mathbf{s}_j\big) = 0$ with high probability. In the right-hand-side member of (29), we also observe that $\lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \rfloor_p + \mathbf{e}_{s,j,x^\star}$ is an injective function of $\mathbf{s}_j$. Indeed, we have

$$\lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \rfloor_p + \mathbf{e}_{s,j,x^\star} = (p/q) \cdot \mathbf{G}^\top \cdot \mathbf{s}_j - \mathbf{t}_{s,j,x^\star} + \mathbf{e}_{s,j,x^\star}$$

for some $\mathbf{e}_{s,j,x^\star} \in (0,1)^m$, so that

$$(q/p) \cdot (\lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \rfloor_p + \mathbf{e}_{s,j,x^\star}) = \mathbf{G}^\top \cdot \mathbf{s}_j + \mathbf{e}'_{s,j,x^\star} \tag{30}$$

for some $\mathbf{e}'_{s,j,x^\star} \in (-q/p, 2 \cdot q/p)^m$.

When assessing the entropy of $\mathbf{z}_j^\star$ conditionally on $\mathcal{A}$'s view, we thus obtain

$$\begin{aligned} H_\infty(\mathbf{z}_j^\star \mid \mathbf{C} \cdot \mathbf{s}_j) &= H_\infty\big(\lfloor \mathbf{R}_{x^\star}^\top \cdot \mathbf{A}^\top \cdot \mathbf{s}_j \rfloor_p + \lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \rfloor_p + \mathbf{e}_{s,x^\star} \mid \mathbf{C} \cdot \mathbf{s}\big) \\ &= H_\infty\big(\lfloor \mathbf{G}^\top \cdot \mathbf{s}_j \rfloor_p + \mathbf{e}_{s,x^\star} \mid \mathbf{C} \cdot \mathbf{s}_j\big) \\ &= H_\infty(\mathbf{s}_j \mid \mathbf{C} \cdot \mathbf{s}_j) \geq n \cdot \log \sigma - n' \cdot \log q - 1. \end{aligned} \tag{31}$$

Here, the second equality follows from the fact that, for any random variables X, Y, Z defined over an additive group, we have $H_\infty(Y + Z \mid X) = H_\infty(Z|X)$ if $H_\infty(Y|X) = 0$. The last inequality follows from Lemma A.6: the distribution of $\mathbf{s}_j \in \mathbb{Z}^n$ conditionally on $\mathbf{C} \cdot \mathbf{s}_j$ is $\mathbf{s}_{j,0} + D_{\Lambda^\perp(\mathbf{C}),\sigma,-\mathbf{s}_{j,0}}$, where $\mathbf{s}_{j,0} \in \mathbb{Z}_q^n$ is an arbitrary solution of $\mathbf{C} \cdot \mathbf{s}_{j,0} = \mathbf{C} \cdot \mathbf{s}_j$. Then, Lemma A.6 implies that the point with highest probability in $D_{\Lambda^\perp(\mathbf{C}),\sigma,-\mathbf{s}_{j,0}}$ occurs with probability $\leq 2 \det(\Lambda^\perp(\mathbf{C}))/\sigma^n$. Since $\det(\Lambda^\perp(\mathbf{C})) = q^{n'}$ with probability $1 - 2^{\Omega(\lambda)}$, we have $H_\infty(\mathbf{s}_j \mid \mathbf{C} \cdot \mathbf{s}_j) \geq n \cdot \log \sigma - n' \cdot \log q - 1$.

To extract statistically uniform bits from each uncorrupted $\mathbf{z}_j^\star$, we need to take into account the fact that $x^\star$ may depend on pp. Assuming that $P_K(X^\star) = 0$, for each uncorrupted $j \in [N]$, the source $\mathbf{z}_j^\star$ is taken from a distribution determined by $X^\star$ within a collection of less than 2^ℓ distributions (i.e., those for which $P_K(X^\star) = 0$), which all have min-entropy $\bar{n} = n \cdot \log \sigma - n' \cdot \log q - 1$. Moreover, the $\{\mathbf{z}_j^\star\}_{j \in [N]}$ are independent since they are obtained by applying the same function to N independent variables. If we call $\mathcal{C} \subset [N]$ the subset of corrupted indexes, by applying Lemma A.9 and Lemma A.8 with $\varepsilon = 2^{-\lambda}$ for a collection $\mathcal{X}$ of at most $M = 2^\ell$ distributions, we obtain that the joint distribution of $\{\pi(\mathbf{z}_j^\star)\}_{j \in [N] \setminus \mathcal{C}}$ is $(\ell \cdot 2^{-\lambda} + 2^{-\ell})$-close to the uniform distribution over $(\mathbb{Z}_p^k)^{|[N] \setminus \mathcal{C}|}$.
□

B Relation Between Definitions 2.11 and 2.12

Proposition B.1. *For any MCFE scheme,* 1Ch-IND *security implies* IND *security.*

Proof. Let us consider an efficient MCFE adversary $\mathcal{A}$ in the IND security game. We show that $\mathcal{A}$ implies an MCFE adversary in the $\mathsf{1Ch\text{-}IND}$ game.

Suppose that $\mathcal{A}$ makes encryption queries for Q distinct tags during the game. The proof uses a standard hybrid argument over the distinct tags that $\mathcal{A}$ queries throughout the attack. Let H_k with $k \in \{0, 1, \ldots, Q\}$ be the game in which the challenger replies to encryption queries $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ in the following way.

– If the tag t is one of the first k distinct tags appearing in $\mathcal{A}$'s encryption queries, it replies with $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^0, t)$.
– Otherwise, it replies with $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^1, t)$.

Note that an IND adversary $\mathcal{A}$ in the sense of Definition 2.11 implies a distinguisher between H_0 and H_Q.

We claim that, for any $k \in \{0, \ldots, Q-1\}$, an efficient distinguisher $\mathcal{A}_k$ between H_k and H_{k+1} implies the existence of an efficient adversary $\mathcal{B}_k$ in $\mathsf{1Ch\text{-}IND}$ game and such that $\mathbf{Adv}^{\mathsf{1Ch\text{-}IND}}(\mathcal{B}_k) = \mathbf{Adv}^{k,k+1}(\mathcal{A}_k)$. This adversary $\mathcal{B}_k$ proceeds as follows.

Initialization: Having received mpk, $\mathcal{B}_k$ runs $\mathcal{A}_k$ by feeding it with the same public parameters mpk.

Encryption queries: For each query $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ sent by $\mathcal{A}_k$, $\mathcal{B}_k$ does the following.

– If t is among the first k distinct tags queried by $\mathcal{A}_k$, $\mathcal{B}_k$ sends the query $\mathsf{QEncrypt}(i, x_i^0, t)$ to its own challenger in the $\mathsf{1Ch\text{-}IND}$ game and relays the answer back to $\mathcal{A}_k$.
– If t coincides with $(k+1)$-th distinct tag queried by $\mathcal{A}_k$, $\mathcal{B}_k$ makes the challenge query $\mathsf{QCEncrypt}(i, x_i^0, x_i^1, t)$ and transmits the response to $\mathcal{A}_k$.
– Otherwise, $\mathcal{B}_k$ sends the encryption query $\mathsf{QEncrypt}(i, x_i^1, t)$ to its $\mathsf{1Ch\text{-}IND}$ challenger and forwards the answer to $\mathcal{A}_k$.

Functional decryption queries: For each query $\mathsf{QDKeygen}(f)$ made by $\mathcal{A}_k$, $\mathcal{B}_k$ sends the same query to its own challenger and passes the answer to $\mathcal{A}_k$.

Corruption queries: For each query $\mathsf{QCorrupt}(i)$ made by $\mathcal{A}_k$, $\mathcal{B}_k$ makes the same query to its own challenger and forwards the answer to $\mathcal{A}_k$.

Finalize: When $\mathcal{A}_k$ halts, $\mathcal{B}_k$ outputs the same bit b' as $\mathcal{A}_k$.

We can conclude that, for any efficient IND adversary $\mathcal{A}$, there exists an efficient $\mathsf{1Ch\text{-}IND}$ adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathsf{IND}}(\mathcal{A}) \leq Q \cdot \mathbf{Adv}^{\mathsf{1Ch\text{-}IND}}(\mathcal{B}).$$

□

Proposition B.2. $\mathsf{1Ch\text{-}IND}$ *security is equivalent to* $\mathsf{1Ch\text{-}IND}'$ *security.*

Proof. Recall that $\mathsf{1Ch\text{-}IND}'$ refers to the same security notion as described in Definition 2.12, except that Condition 2 in the Finalize step is replaced by

$2'$: 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 $x_i^{0\star} \neq x_i^{1\star}$.

The non-trivial implication to prove is that $\mathsf{1Ch\text{-}IND}$ security implies $\mathsf{1Ch\text{-}IND}'$ security. Towards a contradiction, suppose that there exists an adversary $\mathcal{A}'$ that wins the $\mathsf{1Ch\text{-}IND}'$ game with noticeable advantage. We build an adversary $\mathcal{A}$ that breaks the $\mathsf{1Ch\text{-}IND}$ security game. Our adversary $\mathcal{A}$ answers both corruption and functional decryption key queries made by $\mathcal{A}'$ by relaying them to its challenger and forwarding the responses to $\mathcal{A}'$. For a query $\mathsf{QEncrypt}(i, x, t)$ made by $\mathcal{A}'$, $\mathcal{A}$ checks if a challenge query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t)$ was made earlier. If no challenge query has been made so far for the pair (i, t), the reduction sends the same $\mathsf{QEncrypt}(i, x, t)$ query to its challenger and passes the the answer to $\mathcal{A}'$. Otherwise, it makes a $\mathsf{Corrupt}(i)$ query to its challenger and thereby obtains the encryption key ek_i. To answer any subsequent query for slot i, the reduction simply uses ek_i.

To answer a challenge query $\mathsf{QCEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$, the reduction $\mathcal{A}$ forwards the query to its challenger and transmits the answer to $\mathcal{A}$' if it did not previously make an encryption $\mathsf{QEncrypt}(i, x, t^\star)$ for the pair $(i, t^\star)$. If $\mathcal{A}$ previously made an encryption query for the pair $(i, t^\star)$, it also invokes its challenger and sends it the query $\mathsf{Corrupt}(i)$. Upon receiving ek_i, $\mathcal{A}$ is able to answer any query concerning the i-th slot.

Notice that the adversary $\mathcal{A}'$ is only allowed to make both encryption and challenge queries on the same pair $(i, t^\star)$ if $x_i^{\star 0} = x_i^{\star 1}$. This implies that the reduction is allowed to make the query $\mathsf{Corrupt}(i)$ to its challenger, which allows it to answer all queries concerning the i-th slot. □

C Deferred Proofs for the Scheme in Section 3

C.1 Proof of Lemma 3.1 (Correctness)

Proof. Suppose that $\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i \ \in \mathbb{Z}_q^m$, with each error vector $\boldsymbol{e}_i \leftarrow D_{\mathbb{Z}^m, \alpha q}$ and $\tau = \mathsf{AHF}(t)$. Defining $\mathbf{E} = [\boldsymbol{e}_1 | \cdots | \boldsymbol{e}_\ell] \in \mathbb{Z}^{m \times \ell}$, we have

$$\mathbf{f}_{t,\boldsymbol{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{C}_{t,i} - \mathbf{A}(\tau)^\top \cdot \mathbf{s}_{\boldsymbol{y}} = \mathbf{G}_0^\top \cdot \mathbf{X} \cdot \boldsymbol{y} + \mathbf{E} \cdot \boldsymbol{y} \mod q.$$

We know that $\Lambda^\perp(\mathbf{G}_0)$ has a public trapdoor $\mathbf{T}_0 \in \mathbb{Z}^{m \times m}$ such that $\mathbf{G}_0 \cdot \mathbf{T}_0 = \mathbf{0} \mod q$ and $\|\mathbf{T}_0^\top\|_\infty \leq \log q$ where $\|\mathbf{T}_0^\top\|_\infty := \sup_{\boldsymbol{x} \neq \mathbf{0}} \frac{\|\mathbf{T}_0^\top \cdot \boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}$. By using this short trapdoor, we are able to recover $\mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$ in the following way: observe that $\mathbf{T}_0^\top \cdot \mathbf{f}_{t,\boldsymbol{y}} = \mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y} \mod q$. By [59, Lemma 4.4], we can bound the Euclidean norm of a Gaussian vector $\mathbf{e} \hookleftarrow D_{\mathbb{Z}^\ell, \alpha q}$ by $\|\mathbf{e}\| \leq \sqrt{\ell} \cdot \alpha q$ with probability exponentially close to 1. It comes that

$$\|\mathbf{E}\|_\infty = \max_{i \in [m]} \sum_{j=1}^{\ell} |e_{i,j}| \leq \sqrt{\ell} \cdot \max_{i \in [m]} \sqrt{\sum_{j=1}^{\ell} e_{i,j}^2} \leq \ell \cdot \alpha q.$$

Moreover, the hypotheses imply

$$\|\mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y}\|_\infty \leq \|\mathbf{T}_0^\top\|_\infty \cdot \|\mathbf{E}\|_\infty \cdot \|\boldsymbol{y}\|_\infty \leq \log q \cdot \ell \cdot \alpha q \cdot Y < q/2.$$

Hence, $\mathbf{T}_0^\top \cdot \mathbf{f}_{t,\boldsymbol{y}} \bmod q$ actually reveals $\mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y}$ over $\mathbb{Z}^m$. Since $\mathbf{T}_0^\top$ is a $\mathbb{Z}$-basis, it comes that we can recover $\mathbf{E} \cdot \boldsymbol{y} \in \mathbb{Z}^m$, at which point we also get $\mathbf{X} \cdot \boldsymbol{y} \bmod q$. Since $\|\mathbf{X} \cdot \boldsymbol{y}\|_\infty \leq \ell \cdot X \cdot Y < q/2$ by hypothesis, the modular product $\mathbf{X} \cdot \boldsymbol{y} \bmod q$ is nothing but $\mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$. □

C.2 Proof of Lemma 3.3

Proof. If the matrices $\mathbf{A}_{i,b} = \mathbf{A} \cdot \mathbf{R}_{i,b} + \mu_{i,b} \cdot \mathbf{G} \in \mathbb{Z}^{n \times m}$, for $i \in [L]$ and $b \in \{0,1\}$, with $\mathbf{R}_{i,b} \leftarrow \{-1,1\}^{m \times m}$ and $\mu_{i,b} \in \{0,1\}$, defined as in equation (20), then $\mathbf{R}_\tau = \mathbf{R}'_\tau \cdot \mathbf{G}^{-1}(\mathbf{W}^\top)$, where $\mathbf{R}'_\tau$ is given below:

$$\begin{aligned}
\mathbf{R}'_\tau = {} & \mathbf{R}_{L,\tau[L]} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{L-1,\tau[L-1]} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{L-2,\tau[L-2]} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{1,\tau[1]}\right)\right)\right)\right) \\
& + \mu_{L,\tau[L]} \cdot \mathbf{R}_{L-1,\tau[L-1]} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{L-2,\tau[L-2]} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{1,\tau[1]}\right)\right)\right) \\
& + \mu_{L,\tau[L]} \cdot \mu_{L-1,\tau[L-1]} \cdot \mathbf{R}_{L-2,\tau[L-2]} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{L-3,\tau[L-3]} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{1,\tau[1]}\right)\right)\right) \\
& \vdots \\
& + \mu_{L,\tau[L]} \mu_{L-1,\tau[L-1]} \cdots \mu_{3,\tau[3]} \cdot \mathbf{R}_{2,\tau[2]} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{1,\tau[1]}\big) \\
& + \mu_{L,\tau[L]} \mu_{L-1,\tau[L-1]} \cdots \mu_{2,\tau[2]} \cdot \mathbf{R}_{1,\tau[1]}
\end{aligned}$$

$\mathbf{R}_\tau^\top$ is thus a sum of L terms of products of at most 3 binary matrices. Since each binary matrix has infinity norm less than m we obtain the upper bound $\|\mathbf{R}_\tau^\top\|_\infty \leq L \cdot m^3$. By [59, Lemma 4.4], we can bound the Euclidean norm of a Gaussian vector $\boldsymbol{e} \hookleftarrow D_{\mathbb{Z}^n, \alpha_1 q}$ by $\|\mathbf{e}\| \leq \sqrt{n} \cdot \alpha_1 q$, with probability exponentially close to 1. We thus obtain

$$\|\mathbf{E}\|_\infty = \max_{i \in [m]} \sum_{j=1}^{n} |e_{i,j}| \leq \sqrt{n} \cdot \max_{i \in [m]} \sqrt{\sum_{j=1}^{n} e_{i,j}^2} \leq n \cdot \alpha_1 q,$$

so that $\|\mathbf{R}_\tau^\top \cdot \mathbf{E} \cdot \mathbf{s}\|_\infty \leq \|\mathbf{R}_\tau^\top\|_\infty \cdot \|\mathbf{E}\|_\infty \cdot \|\mathbf{s}\|_\infty \leq Lm^3 \cdot n\alpha_1 q \cdot \sigma\sqrt{n}$. By Lemma 2.4, the statistical distance to be bounded is at most $\leq m \cdot \frac{Lm^3 \cdot n\alpha_1 q \cdot \sigma\sqrt{n}}{\alpha q} = 2^{-\lambda}$. □

C.3 Proof of lemma 3.5

Proof. By Remark 2.3, each column of $\mathbf{T}$ has norm smaller than

$$\|\mathbf{t}_i\| \leq O(\sqrt{n \cdot (n_0 + n_1) \log q}) \cdot \omega(\sqrt{\log n}) = \omega(n\sqrt{\log n}),$$

so that $\|\mathbf{T} \cdot \mathit{\Delta}\boldsymbol{x}_i\|_\infty \leq \|\mathbf{T} \cdot \mathit{\Delta}\boldsymbol{x}_i\| \leq 2Xn_0 \cdot \omega(n\sqrt{\log n})$. By Lemma 2.4, the considered distance is smaller than $n \cdot \frac{2Xn_0 \cdot \omega(n\sqrt{\log n})}{\sigma} \leq 2^{-\lambda} \cdot (4X)^{-n_0 \ell}$. □

D Proof of Theorem 4.1

Proof. The proof considers a sequence of games. The first game corresponds to the real sta-IND-sec game where the challenger's bit is $b = 0$. The final game is the sta-IND-sec game where $b = 1$. We will prove that they are computationally indistinguishable assuming that the scheme of Section 3 provides IND-sec security in the sense of Definition 2.11. For each i, we denote by W_i the event that the adversary outputs $\beta = 1$ in Game_i.

Game_0: This is the real security sta-IND-sec security game, which corresponds to the game of Definition 2.14 with the sets $\mathcal{CS}$ and $\mathcal{HS}$ chosen before the initialization phase. Letting $\ell_h = |\mathcal{HS}|$ and $|\mathcal{CS}| = \ell - \ell_h$, we assume w.l.o.g. that $\mathcal{A}$ chooses to corrupt the last $\ell - \ell_h$ senders: i.e., $\mathcal{HS} = \{1, \ldots, \ell_h\}$ and $\mathcal{CS} = \{\ell_h + 1, \ldots, \ell\}$. By definition, the adversary outputs $\beta = 1$ with probability $\Pr[W_0]$.

Game_1: In this game, we modify the $\mathsf{QDKeygen}(\cdot, \cdot)$ oracle, which generates partial functional decryption keys. At each partial functional decryption key query $\mathsf{QDKeygen}(i, f)$ for an index $i \in \mathcal{HS}$ and a function f described by a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in [-Y, Y]^\ell$, the challenger $\mathcal{C}$ responds as follows:

- If the pair (i, f) is such that not all indexes $j \in \mathcal{HS} \setminus \{i\}$ have been the input of a query $\mathsf{QDKeygen}(j, f)$ for the same function f, $\mathcal{C}$ samples a Gaussian vector $\mathbf{s}'_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ and encrypts $y_i \cdot \mathbf{s}'_i \in \mathbb{Z}^n$ as

$$\mathsf{dk}_{f,i} = \bar{\mathbf{G}}^\top \cdot (y_i \cdot \mathbf{s}'_i) + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i} \ \in \mathbb{Z}_{\bar{q}}^{\bar{m}},$$

where $\mathbf{e}_{f,i} \hookleftarrow D_{\mathbb{Z}^{\bar{m}},\alpha q}$.

- If the pair (i, f) is such that similar queries (j, f) have been made earlier for the same function f and for all indexes $j \in \mathcal{HS} \setminus \{i\}$, $\mathcal{C}$ recalls the vectors $\{\mathbf{s}'_j\}_{j \in \mathcal{HS}\setminus\{i\}}$ that were chosen to answer earlier queries. It computes $\mathbf{s_y} = \sum_{i=1}^{\ell} \mathbf{s}_i \cdot y_i \in \mathbb{Z}^n$ as in the centralized scheme of Section 3. Then, it defines

$$\mathbf{s}'_{f,i} = \mathbf{s_y} - \sum_{j \in \mathcal{HS}\setminus\{i\}} y_j \cdot \mathbf{s}'_j - \sum_{i=\ell_h+1}^{\ell} \mathbf{s}_i \cdot y_i \qquad \text{over } \mathbb{Z}, \tag{32}$$

The challenger $\mathcal{C}$ then responds the query $\mathsf{QDKeygen}(i, f)$ by encrypting $\mathbf{s}'_{f,i} \in \mathbb{Z}^n$ and returning

$$\mathsf{dk}_{f,i} = \bar{\mathbf{G}}^\top \cdot \mathbf{s}'_{f,i} + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i} \ \in \mathbb{Z}_{\bar{q}}^{\bar{m}}, \tag{33}$$

where $\mathbf{e}_{f,i} \hookleftarrow D_{\mathbb{Z}^{\bar{m}},\alpha\bar{q}}$.

We note that the vectors $\mathbf{s}'_j$ sampled at each query do not have to be consistent across partial key generation queries for distinct functions (they just have to be consistent across queries $\mathsf{QDKeygen}(\cdot, f)$ involving the same function f). This will not be necessary since we are just seeking to simulate the partial key generation oracle $\mathsf{QDKeygen}(\cdot,\cdot)$ in a way that is computationally indistinguishable from the previous game. We also note that the vector $\mathbf{s}'_{f,i} \in \mathbb{Z}^n$ encrypted in (33) may not even be in the lattice $y_i \cdot \mathbb{Z}^n$ (in contrast with a vector encrypted by the real $\mathsf{DKeygenShare}$ algorithm). However, this will not be a problem here. Indeed, our goal is solely to simulate the $\mathsf{QDKeygen}(\cdot,\cdot)$ oracle in such a way that Game_1 remains computationally indistinguishable from Game_0 and, in the next transition from Game_1 to Game_2, allows reducing the security of the scheme in Section 3 to that of our DMCFE scheme.

Lemma D.1 shows that Game_1 is computationally indistinguishable from Game_0 as long as the centralized MCFE scheme of Section 3 is secure in the sense of Definition 2.11. We note that a weaker security definition than Definition 2.11 would be sufficient: as a centralized MCFE adversary, the reduction only needs to: (i) send a single functional secret key query for the vector $(1,1,\ldots,1)^\top \in \mathbb{Z}^\ell$ to its centralized MCFE challenger; (ii) make all its corruption queries at once after having received the master public key.

Game_2: In this game, we modify the encryption oracle. At each encryption query $\mathsf{QEncrypt}(i, \boldsymbol{x}_{0,i}, \boldsymbol{x}_{1,i}, t)$, the challenger $\mathcal{CH}$ encrypts $\boldsymbol{x}_{1,i} \in [-X,X]^{n_0}$ instead of $\boldsymbol{x}_{0,i} \in [-X,X]^{n_0}$ (we assume that either $\boldsymbol{x}_{0,i} = \boldsymbol{x}_{1,i}$ or $i \in \mathcal{HS}$ since, otherwise, $\mathcal{A}$'s output is eventually replaced by a random bit). Namely, it sample $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and computes

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{1,i} + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i \ \in \mathbb{Z}_q^m.$$

where

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

with $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$.

Lemma D.2 relies on the $\mathsf{IND\text{-}sec}$ security of the scheme in Section 3 to show that Game_2 is computationally indistinguishable from Game_2 under the $\mathsf{LWE}_{q,m,n_1,\alpha_1}$ assumption.

Game_3: In this game, we modify again the $\mathsf{QDKeygen}(\cdot,\cdot)$ oracle and restore it to its original behavior, which is that of the real scheme. This game transition is exactly identical to the one from Game_0 to Game_1, but in the converse direction. Lemma D.3 states that, as long as the MCFE scheme of Section 3 provides $\mathsf{IND\text{-}sec}$ security in the sense of Definition 2.11, Game_3 and Game_2 are computationally indistinguishable.

In Game_3, it is easy to see that $\mathcal{C}$ and $\mathcal{A}$ are playing the actual $\mathsf{sta\text{-}IND\text{-}sec}$ game where the challenge bit of $\mathcal{C}$ is $b = 1$.

Putting the above altogether, we find that Game_0 and Game_3 are computationally indistinguishable so long as the centralized MCFE scheme of Section 3 provides security in the sense of Definition 2.11. □

Lemma D.1. Game_0 *and* Game_1 *are computationally indistinguishable assuming that the MCFE scheme of Section 3 provides* $\mathsf{IND}\text{-}\mathsf{sec}$ *security. Under the* $\mathsf{LWE}_{\bar{q},\bar{m},\bar{n},\bar{\alpha}_1}$ *assumption, we have* $|\Pr[W_1] - \Pr[W_0]| \in \mathsf{negl}(\lambda)$.

Proof. Let us assume that an adversary $\mathcal{A}$ can distinguish Game_1 from Game_0 with noticeable advantage ε. We construct a $\mathsf{IND}\text{-}\mathsf{sec}$ adversary $\mathcal{B}$ against the centralized MCFE scheme with the same advantage ε.

Our MCFE adversary $\mathcal{B}$ obtains from its $\mathsf{IND}\text{-}\mathsf{sec}$ challenger a master public key

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

where the common public parameters

$$\mathsf{cp} := \Big(\ \lambda,\ \ell_{\max},\ 2\ell\cdot\beta\cdot Y,\ 1,\ n,\ \bar{n}_1,\ \bar{n},\ \bar{m},\ \alpha,\ \bar{\sigma},\ \ell_t,\ L,\ \bar{q},\ \mathsf{AHF}_f\Big).$$

specify the message space $\mathcal{M} = [-2\ell\beta Y, 2\ell\beta Y]^n$ and the functional secret key space $[-1,1]^\ell$. Then, $\mathcal{B}$ first sends its challenger a unique functional secret key query for the vector $(1,1,\ldots,1)^\top$ and obtains in return $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i \in \mathbb{Z}^{\bar{n}}$. Next, $\mathcal{B}$ chooses random matrices $\mathbf{A}_{i,b} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, for each $i \in [L]$, $b \in \{0,1\}$. It also chooses random matrices $\mathbf{V} \hookleftarrow U(\mathbb{Z}_q^{n_0\times n})$. For each $i \in [\ell]$, it chooses users' secret keys $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ in the second MCFE instance. This allows $\mathcal{B}$ to create a master public key

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

where

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

is augmented with a description of a second balanced admissible hash function family $\mathsf{AHF}_t : \{0,1\}^{\ell_t} \to \{0,1\}^L$.

Before handing mpk to $\mathcal{A}$, our centralized MCFE adversary $\mathcal{B}$ waits until $\mathcal{A}$ chooses the set $\mathcal{CS} \subset [\ell]$ of players it wants to corrupt. We assume w.l.o.g. that $\mathcal{A}$ chooses $\mathcal{CS} = \{\ell_h + 1, \ldots, \ell\}$. At this point, $\mathcal{B}$ sends corruption queries $\mathsf{QCorrupt}(i)$ to its own challenger for all $i \in \mathcal{CS}$. The MCFE challenger replies by sending $\{\mathbf{t}_i\}_{i\in\mathcal{CS}}$, at which point $\mathcal{B}$ gives $\{\mathsf{sk}_i = (\mathbf{s}_i, \mathbf{t}_i)\}_{i\in\mathcal{CS}}$ and mpk to $\mathcal{A}$.

At any time, $\mathcal{B}$ can make a query $\mathsf{QDKeygen}(i,f)$ for an index $i \in \mathcal{HS}$ and a function f described by a vector $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in \mathbb{Z}^\ell$. At each such query, $\mathcal{B}$ proceeds as follows:

- If the pair (i,f) is such that not all indexes $j \in \mathcal{HS} \setminus \{i\}$ have been the input of a query (j,f) for the same function f, $\mathcal{B}$ samples $\mathbf{s}'_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ and defines $\mathbf{s}'_{f,i} = y_i \cdot \mathbf{s}'_i \in \mathbb{Z}^n$.

- If the pair (i, f) is such that queries $\mathsf{QDKeygen}(j, f)$ have been made before for the same function f on all indexes $j \in \mathcal{HS} \setminus \{i\}$, $\mathcal{C}$ recalls the vectors $\{\mathbf{s}'_j\}_{j \in \mathcal{HS} \setminus \{i\}}$ and $\{\mathbf{s}'_{f,j}\}_{j \in \mathcal{HS} \setminus \{i\}}$ that were chosen to handle earlier queries for f. It then defines $\mathbf{s}_{\mathbf{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ which is the secret key that would be computed for $\boldsymbol{y} \in [-Y, Y]^\ell$ using the real master secret key $\{\mathbf{s}_i\}_{i=1}^{\ell}$ in the centralized scheme of Section 3. It finally defines

$$\mathbf{s}'_{f,i} = \mathbf{s}_{\mathbf{y}} - \sum_{j \in \mathcal{HS} \setminus \{i\}} \mathbf{s}'_{f,j} - \sum_{i=\ell_h+1}^{\ell} \mathbf{s}_i \cdot y_i \qquad \text{over } \mathbb{Z}, \tag{34}$$

Then, $\mathcal{B}$ sends the query $\mathsf{QEncrypt}(i, y_i \cdot \mathbf{s}_i, \mathbf{s}'_{f,i}, t_f)$ to its encryption oracle in the $\mathsf{IND\text{-}sec}$ game. Depending on the value of its secret bit $b \in \{0, 1\}$, the MCFE challenger replies with either

$$\mathsf{dk}_{f,i} = \bar{\mathbf{G}}^\top \cdot (y_i \cdot \boldsymbol{s}_i) + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i}$$

or

$$\mathsf{dk}_{f,i} = \bar{\mathbf{G}}^\top \cdot \mathbf{s}'_{f,i} + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i}, \tag{35}$$

where $\mathbf{e}_{f,i} \hookleftarrow D_{\mathbb{Z}^{\bar{m}}, \alpha q}$, and the response $\mathsf{dk}_{f,i}$ is returned to $\mathcal{A}$ as a partial functional secret key for the vector $\boldsymbol{y}$.

At each encryption query $(i, \boldsymbol{x}_{0,i}, \boldsymbol{x}_{1,i}, t)$ made by $\mathcal{A}$, $\mathcal{B}$ can compute

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{0,i} + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i \ \in \mathbb{Z}_q^m, \tag{36}$$

itself by faithfully running the encryption algorithm as it knows $\{\mathbf{s}_i\}_{i=1}^{\ell}$.

When $\mathcal{A}$ terminates, $\mathcal{B}$ checks if $\mathcal{A}$ made a query $\mathsf{QDKeygen}(i, f)$ for a function f such that not all indexes $j \in \mathcal{HS} \setminus \{i\}$ were the input of a query $\mathsf{QDKeygen}(j, f)$. If so, $\mathcal{B}$ makes the missing queries for itself. Namely, for each such function f, $\mathcal{B}$ parses the corresponding vector as $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in [-Y, Y]^\ell$ and defines $\mathcal{Q}_f \subset \mathcal{HS}$ to be the set of indexes j for which no query $\mathsf{QDKeygen}(j, f)$ was made by $\mathcal{A}$. For each $j \in \mathcal{Q}_f$, $\mathcal{B}$ does the following.

- If j is not the last unqueried index for f, $\mathcal{B}$ samples a Gaussian $\mathbf{s}'_j \hookleftarrow D_{\mathbb{Z}^n, \sigma}$ and sends the query $\mathsf{QEncrypt}(j, y_j \cdot \mathbf{s}_j, y_j \cdot \mathbf{s}'_j, t_f)$ to its MCFE challenger.
- If j is the last unqueried index for f, $\mathcal{B}$ defines

$$\mathbf{s}'_{f,j} = \sum_{i \in \mathcal{HS}} y_i \cdot \mathbf{s}_i - \sum_{i \in \mathcal{HS} \setminus \{j\}} y_i \cdot \mathbf{s}'_i \tag{37}$$

  and makes the encryption query $\mathsf{QEncrypt}(j, y_j \cdot \mathbf{s}_j, \mathbf{s}'_{f,j}, t_f)$. Since the only functional secret key query made by $\mathcal{B}$ is for the vector $\mathbf{1} = (1, 1, \ldots, 1)^\top$, $\mathcal{B}$ only needs to make sure that dummy encryption queries are made for pairs of vectors that sum to the same value over indexes in $\mathcal{HS}$. This is ensured by defining the last term as (37).

In both cases, $\mathcal{B}$ just ignores the responses of its challenger as these dummy encryption queries are only made to make sure that $\mathcal{B}$'s advantage will not be annihilated by Condition 2 in Definition 2.11.

Finally, $\mathcal{B}$ halts and outputs whatever $\mathcal{A}$ outputs. We observe that $\mathcal{B}$ strictly complies with the rules of the $\mathsf{IND\text{-}sec}$ game at any time. In particular:

- It never makes a $\mathsf{QEncrypt}(i,\cdot,\cdot,\cdot)$ query or a corrupted index $i \in \mathcal{CS}$.
- By construction, $\mathcal{B}$'s encryption queries $\mathsf{QEncrypt}(i, y_i \cdot \mathbf{s}_i, \mathbf{s}'_{f,i}, t_f)$ are all made for message pairs $(y_1 \cdot \mathbf{s}_1, \ldots, y_\ell \cdot \mathbf{s}_\ell)$ and $(\mathbf{s}'_{f,1}, \ldots, \mathbf{s}'_{f,\ell_h}, y_{\ell_h+1} \cdot \mathbf{s}_{\ell_h+1}, \ldots, y_\ell \cdot \mathbf{s}_\ell)$ such that
$$\sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i = \sum_{i=1}^{\ell_h} \mathbf{s}'_{f,i} + \sum_{i=\ell_h+1}^{\ell} y_i \cdot \mathbf{s}_i \qquad (\text{over } \mathbb{Z})$$

The lemma's claim follows from the fact that, if the secret bit of $\mathcal{B}$'s MCFE challenger is $b = 0$, $\mathcal{A}$'s view is exactly the same as in Game_0. If $b = 1$, the distribution of partial functional secret keys is given by (35), which corresponds to Game_1. ☐

Lemma D.2. Game_2 *and* Game_1 *are computationally indistinguishable assuming that the MCFE scheme of Section 3 provides security in the* $\mathsf{IND\text{-}sec}$ *sense. Under the* $\mathsf{LWE}_{q,m,n_1,\alpha_1}$ *assumption, we have* $|\Pr[W_2] - \Pr[W_1]| \in \mathsf{negl}(\lambda)$.

Proof. Let $\mathcal{A}$ be an adversary that can distinguish between Game_1 and Game_2 with noticeable advantage ε. We build an adversary $\mathcal{B}$ against the centralized scheme that wins the $\mathsf{IND\text{-}sec}$ game (cf. Definition 2.11) with the same advantage.

The reduction $\mathcal{B}$ begins by obtaining from its challenger a master public key
$$\widetilde{\mathsf{mpk}} := \Big(\mathsf{cp}, \mathbf{V},\ \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\ \in \mathbb{Z}_q^{n \times m}\ \}_{i=1}^{L}\Big),$$
where the common public parameters
$$\tilde{\mathsf{cp}} := \Big(\ \lambda,\ \ell_{\max},\ X,\ Y,\ n_0,\ n_1,\ n,\ m,\ \alpha,\ \sigma,\ \ell_t,\ L,\ q,\ \mathsf{AHF}_t\Big).$$

The adversary $\mathcal{A}$ declares the set $\mathcal{CS}$ of corrupted senders, as mandated by the static security game. Then, $\mathcal{B}$ makes the relevant queries $\mathsf{QCorrupt}(i)$ for each $i \in \mathcal{CS}$ to its challenger and the latter replies with $\{\mathbf{s}_i \in \mathbb{Z}^n\}_{i \in \mathcal{CS}}$.

Next, $\mathcal{B}$ chooses $\bar{\mathbf{V}} \hookleftarrow U(\mathbb{Z}_{\bar{q}}^{n \times \bar{n}})$ and $\mathbf{B}_{i,b} \hookleftarrow U(\mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}})$ for each $i \in [L]$ and $b \in \{0,1\}$. It also samples $\mathbf{t}_i \hookleftarrow D_{\mathbb{Z}^{\bar{n}},\bar{\sigma}}$ for $i \in [\ell]$ and computes $\mathbf{t} = \sum_{i=1}^{\ell} \mathbf{t}_i$. This allows $\mathcal{B}$ to answer $\mathcal{A}$'s static corruption queries by revealing $(\mathbf{s}_i, \mathbf{t}_i) \in \mathbb{Z}^n \times \mathbb{Z}^{\bar{n}}$ for all $i \in \mathcal{CS}$. In addition, $\mathcal{B}$ can feed $\mathcal{A}$ with the public parameters for the decentralized scheme, which consist of $\mathsf{mpk} := (\ \widetilde{\mathsf{mpk}},\ \bar{\mathsf{mpk}},\ \mathbf{t})$, where
$$\bar{\mathsf{cp}} := \Big(\ \lambda,\ \ell_{\max},\ \bar{X},\ \bar{Y}, n,\ \bar{n}_1,\ \bar{n},\ \bar{m},\ \bar{\alpha},\ \bar{\sigma},\ \ell_f,\ L,\ \bar{q},\ \mathsf{AHF}_f\Big),$$
where $\bar{X} = 2\ell\beta Y$, and
$$\bar{\mathsf{mpk}} := \Big(\bar{\mathsf{cp}},\ \bar{\mathbf{V}},\ \{\mathbf{B}_{i,0},\ \mathbf{B}_{i,1} \in \mathbb{Z}_{\bar{q}}^{\bar{n} \times \bar{m}}\}_{i=1}^{L}\Big)$$

For each encryption query $\mathsf{QEncrypt}(i, \boldsymbol{x}_{0,i}, \boldsymbol{x}_{1,i}, t)$ made by $\mathcal{A}$ for some honest sender $i \in \mathcal{HS}$, $\mathcal{B}$ passes the same encryption query to its challenger and relays the response back to $\mathcal{A}$. This response is of the form

$$\mathbf{C}_{t,i} := \mathbf{G}_0^\top \cdot \boldsymbol{x}_{b,i} + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i, \tag{38}$$

where $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and $b \in \{0,1\}$ is the MCFE challenger's random bit.

When $\mathcal{A}$ makes an functional decryption query $\mathsf{QDKeygen}(i, f_{\boldsymbol{y}})$, $\mathcal{B}$ parses the corresponding vector as $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in [-Y, Y]^\ell$ and does the following:

– If there exists $j \in \mathcal{HS} \setminus \{i\}$ such that $\mathsf{QDKeygen}(j, f_{\boldsymbol{y}})$ has not been asked yet, then $\mathcal{B}$ samples $\mathbf{s}'_i \leftarrow D_{\mathbb{Z}^n,\sigma}$ and returns

$$\mathsf{dk}_{f,i} := \bar{\mathbf{G}}^\top \cdot (y_i \cdot \mathbf{s}'_i) + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i}$$

with $\mathbf{e}_{f,i} \leftarrow D_{\mathbb{Z}^{\bar{m}},\bar{\alpha}\bar{q}}$.

– If $\mathsf{QDKeygen}(j, f_{\boldsymbol{y}})$ queries have been made for all $j \in \mathcal{HS} \setminus \{i\}$, $\mathcal{B}$ makes the functional decryption query $\mathsf{QDKeygen}(f_{\boldsymbol{y}})$ to its centralized MCFE challenger. The challenger replies with $\mathbf{s}_{\mathbf{y}} \in \mathbb{Z}^n$. Using the $\mathbf{s}'_j \in \mathbb{Z}^n$ that were previously chosen for the function f, $\mathcal{B}$ computes

$$\mathbf{s}'_{f,i} = \mathbf{s}_{\mathbf{y}} - \sum_{j \in \mathcal{HS} \setminus \{i\}} y_j \cdot \mathbf{s}'_j - \sum_{i=\ell_h+1}^{\ell} y_i \cdot \mathbf{s}_i \quad \in \mathbb{Z}^n,$$

and returns

$$\mathsf{dk}_{f,i} := \bar{\mathbf{G}}^\top \cdot \mathbf{s}'_{f,i} + \mathbf{B}(\tau_f)^\top \cdot \mathbf{t}_i + \mathbf{e}_{f,i}$$

with $\mathbf{e}_{f,i} \leftarrow D_{\mathbb{Z}^{\bar{m}},\bar{\alpha}\bar{q}}$

When $\mathcal{A}$ halts, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. We observe that we have $y_j \cdot \mathbf{s}'_j \in [-\bar{X}, \bar{X}]^n$, $\bar{X} = 2\ell Y \beta$, for all $j \in \mathcal{HS}$. We also observe that, if $\mathcal{A}$ makes all encryption queries $\mathsf{QEncrypt}(j, \boldsymbol{x}_{0,j}, \boldsymbol{x}_{1,j}, t)$ for all indexes $j \in \mathcal{HS}$ for a given tag t, so does $\mathcal{B}$ for the same tag t.

By construction, $\mathcal{B}$ only makes a functional decryption query $\mathsf{QDKeygen}(f_{\boldsymbol{y}})$ for a function $f_{\boldsymbol{y}}$ when $\mathcal{A}$ makes a partial functional key query $\mathsf{QDKeygen}(j, f_{\boldsymbol{y}})$ for the last honest sender's index $j \in \mathcal{HS}$. In the Finalize step of Definition 2.14, Condition 3 requires that, for any function $f_{\boldsymbol{y}}$ involved in queries $\mathsf{QDKeygen}(i, f_{\boldsymbol{y}})$ for all honest senders' indexes $i \in \mathcal{HS}$, the condition $f_{\boldsymbol{y}}(\mathbf{X}_0) = f_{\boldsymbol{y}}(\mathbf{X}_1)$ be satisfied for any pair of messages $\mathbf{X}_0 = [\boldsymbol{x}_{0,1}| \cdots |\boldsymbol{x}_{0,\ell}]$ and $\mathbf{X}_1 = [\boldsymbol{x}_{1,1}| \cdots |\boldsymbol{x}_{1,\ell}]$ satisfying the two sub-conditions of Condition 3. This ensures that $\mathcal{B}$'s functional key queries never break the rules imposed by Condition 3 of the Finalize step of the game described by Definition 2.11.

From (38), it follows that, if secret bit of $\mathcal{B}$'s challenger is $b = 0$, $\mathcal{B}$ is playing Game_1 with $\mathcal{A}$. Similarly, they are playing Game_2 if $b = 1$. $\square$

Lemma D.3. Game_3 *and* Game_2 *are computationally indistinguishable as long as that the MCFE scheme of Section 3 provides* IND-sec *security. Under the* $\mathsf{LWE}_{q,m,n_1,\alpha_1}$ *assumption, we have* $|\Pr[W_3] - \Pr[W_2]| \in \mathsf{negl}(\lambda)$.

Proof. The proof is almost identical to that of Lemma D.1, the only difference being that, at each encryption query $(i, \boldsymbol{x}_{0,i}, \boldsymbol{x}_{1,i}, t)$ made by $\mathcal{A}$, the reduction $\mathcal{B}$ encrypts $\boldsymbol{x}_{1,i}$ (instead of $\boldsymbol{x}_{0,i}$ in (36)). The details are omitted. □