Efficient Lattice-Based Inner-Product Functional Encryption

Jose Maria Bermudo Mera[1], Angshuman Karmakar[1], Tilen Marc[2,3], and Azam Soleimanian[4,5]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
{Jose.Bermudo,Angshuman.Karmakar}@esat.kuleuven.be
[2] Faculty of Mathematics and Physics, University of Ljubljana, Slovenia
[3] XLAB d.o.o., Ljubljana, Slovenia
tilen.marc@xlab.si
[4] Equipe Grace, LIX, École Polytechnique, Palaiseau, France
soleimanian@lix.polytechnique.fr
[5] INRIA, Saclay, France

Abstract. In the recent years, many research lines on Functional Encryption (FE) have been suggested and studied regarding the functionality, security, or efficiency. Nevertheless, an open problem on a basic functionality, the single-input inner-product (IPFE), remains: can IPFE be instantiated based on the Ring Learning With Errors (RLWE) assumption?
The RLWE assumption provides quantum-resistance security while in comparison with LWE assumption gives significant performance and compactness gains. In this paper we present the first RLWE-based IPFE scheme. We carefully choose strategies in the security proofs to optimize the size of parameters. More precisely, we develop two new results on ideal lattices. The first result is a variant of Ring-LWE, that we call multi-hint extended Ring-LWE, where some hints on the secret and the noise are given. We present a reduction from RLWE problem to this variant. The second tool is a special form of Leftover Hash Lemma (LHL) over rings, known as Ring-LHL.
To demonstrate the efficiency of our scheme we provide an optimized implementation of RLWE-based IPFE scheme and show its performance on a practical use case.
We further present new compilers that, combined with some existing ones, can transfer a single-input FE to its (identity-based, decentralized) multi-client variant with linear size of the ciphertext (w.r.t the number of clients).

Keywords: Functional Encryption, Inner-Product, Lattice-Based Cryptography, Learning with Errors over Ring, Multi-Client Functional Encryption.

1 Introduction

Functional Encryption (FE) [11,31] is an extended form of traditional public-key encryption, which can overcome the all-or-nothing access, inherent to the public-key encryption. It allows an authorized user holding a functional-key sk_f to get a function of the message as $f(m)$, by applying sk_f to the encryption of the message m. The functionality provided by this primitive can be useful in practical scenarios such as cloud computing and computation over encrypted data without interactions. The FE schemes supporting general computation circuits either are secure only against a bounded numbers of collusions [21,22], or rely on strong primitives [18]. More importantly, they all suffer from severe inefficiency.

For these reasons a research area emerged with the goal of designing FE with limited but still wide classes of functionalities that are efficient enough to be implemented and used in practice. Particularly, FE for Inner-Product (IP) functionality [4,7], is one of the most popular special cases of FE.

Inner-Product FE (IPFE) [4,7] is a special case of FE supporting the inner-product functionality. In an IPFE scheme the message is a vector $\mathbf{x} \in \mathcal{M}^n$ encrypted as ct_x and the decryption-key sk_y is associated with a n-dimensional vector $\mathbf{y}$. The decryption (of ct_x using sk_y) gets $\langle \mathbf{x}, \mathbf{y} \rangle$, i.e. the inner-product.

IPFE is a well studied problem which is already instantiated based on different assumptions such as the *Decisional Diffie-Helman* (DDH), *Decisional Composite Reminder* (DCR), and *Learning With Errors* (LWE) [4,7] assumption. Despite of all the progress in this field, it has still remained an open problem to present an efficient IPFE based on quantum-secure assumptions. The only quantum-secure assumption that an IPFE has been realized on, is LWE assumption [4,7] with the resulting public key IPFE construction being computationally demanding.

Security of FE. Indistinguishability (IND) [11] is the standard security notion for FE. Informally, it says that an adversary given a ciphertext ct_{m^b}, for $b \xleftarrow{R} \{0,1\}$, cannot distinguish between challenges m^0 and m^1, even if it has access to decryption-keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_k}$, for $k = \text{poly}(\kappa)$, conditioned on $f_i(m^0) = f_i(m^1)$.

One can further consider two kinds of IND-security: selective and adaptive. In selective-IND (sel-IND), the adversary is restricted to submit its challenges m^0 and m^1 at the very beginning of the game and before seeing the public-key, while in adaptive-IND there is no such restriction.

Multi-Client FE (MCFE) is a stronger form of FE where the data comes from different sources and therefore each client should be able to encrypt its data individually and without thrusting other clients [14]. This means that the security definition of multi-client FE considers corruptions of users as well. Multi-client is usually defined w.r.t to a label set, which brings more flexibility, in the sense that ciphertexts can be combined if and only if they are encrypted under the same label. This is necessary for many applications since otherwise an adversary could mix ciphertexts that were not intended so. In fact, in many applications data may have already been defined w.r.t a label, such as a time-stamp (e.g. monthly data) or others.

Decentralized MCFE (DMCFE) avoids the need for a trusted authority who has access to all the secret keys in the system in order to generate the functional keys [3,14]. Particularly, in DMCFE, the clients take the role of authority together, without trusting each other.

Lattice-Based IPFE. Informally, a lattice $\mathcal{L}$ is a discrete subset of $\mathbb{R}^n$ which can be generated by (integer) linear combinations of several vectors, known as the basis. In this setting, the nice variety of computationally-hard problems against quantum adversaries make it interesting for the cryptography purpose [8].

The problem of Learning With Errors (LWE) [36] discusses solving a system of noisy equations and is known to be as hard as standard hard lattice-problems in the worst case. This problem is usually used as a bridge between cryptosystems and standard hard lattice-problems. Agrawal et al. [7] proposed an IPFE relying on hardness of LWE problem. Unfortunately, due to the large-dimension matrices in the LWE problem (leading to the large keys and slow operations), the resulting construction is not truly practical. The scheme of [4] suffers from similar issues while it is only selectively-secure. In [39], authors tried to improve the standard deviation of error term (by using re-randomization technique of [24] instead of using multi-hint extended LWE assumption), but the size of the public key still grows quadratically w.r.t the length of the message and the LWE-parameter n.
RLWE. The Ring-LWE (RLWE) problem, introduced by Lyubashevsky et al. [28], is the problem of distinguishing between two distributions in a special ring of polynomials R_q:

$$(a, as + e) \quad \text{and} \quad (a, u)$$

with $a, u \overset{R}{\leftarrow} \mathsf{R}_q$, the secret $s \leftarrow \chi$, and noise $e \leftarrow \chi$, where χ is a special distribution over the ring, and all the samples share the same secret s. It was introduced as a more efficient and compact version of LWE problem, which can be defined in a similar way, but simply over $\mathbb{Z}_q$ (i.e., $a, s \in \mathbb{Z}_q^n$,$e, u \in \mathbb{Z}_q$) rather than R_q.

Note that the hardness of RLWE depends on the choice of ring R_q and distribution χ. In [28] it was shown that RLWE, with properly chosen parameters, is as hard as standard hard lattice problems.

Due to its compact form, relying on RLWE usually leads to practical encryption systems with smaller keys. Thanks to the Fast Fourier Transform, multiplication in rings can be further accelerated. Moreover, the ring structure allows to encrypt multiple messages in parallel allowing SIMD type of calculations on encrypted data. These properties make RLWE one of the most interesting and competitive assumptions to develop a post-quantum cryptosystem based on [13, 38].

Challenges and Contributions

Although RLWE can provide significant efficiency gains, reducing the security of an encryption systems to RLWE assumption is usually more complicated and tricky, compared with the ones based on LWE. The main obstacles here are: either the lack of common cryptographic-tools compatible with the ring structure, or the lack of variants of RLWE (which are as hard as RLWE) compatible with certain encryption systems. In comparison, LWE is a better understood problem with several variants, and thanks to its matrix-based structure in $\mathbb{Z}_q$, it can be more easily combined with other tools and assumptions during security proofs.

Primary Task: In this work, we study the IPFE cryptosystem and the required tools for the security reduction from IPFE to RLWE.
Secondary Task: We optimize, implement and further extend the scheme to make it applicable to real-world use cases. This includes extending the IPFE scheme to its MCFE and DMCFE versions through new general compilers, implementing it in a highly optimized way, and demonstrating its benefits (including SIMD processing) on a machine learning task.

The first IPFE scheme based on quantum-secure assumption was developed in [4]. This scheme is based on the LWE assumption and proved to be selectively secure. In [7], authors presented an adaptively secure IPFE scheme relying on the same assumption. To extend the security to the adaptive case, they used a variant of LWE assumption, named multi-hint extended-LWE (mhe-LWE) in which some hints on the noise terms are considered. The mhe-LWE says that samples are still indistinguishable from uniform, even given these hints. They proved a reduction from mhe-LWE to LWE problem, for a proper choice of parameters. This variant of LWE is then used directly in the security proof of their IPFE scheme, where hints help to simulate the queries. In the first step, by mhe-LWE, they manage to insert a uniformly random vector in the ciphertext. But as this randomness is multiplied by another vector, in the second step, they still need to apply the Leftover Hash Lemma (LHL) to get a uniform term in the ciphertext.

In this work we follow a somewhat similar approach, while due to the algebraic structure of RLWE and the mentioned obstacles, the details need to be crafted carefully. We build our required tools step by step, namely we extend the similar variants of mhe-LWE and LHL over rings (called mhe-RLWE and Ring-LHL respectively). Our mhe-RLWE assumption not only supports the hints over the error but also over the secret. This property gives special flexibility in the security proof to still improve the size of the parameters. We then construct two IPFE schemes based on RLWE assumption: an adaptively secure whose security proof employs mhe-RLWE and Ring-LHL, and a more efficient but just selectively secure scheme relying only on mhe-RLWE. Thanks to the extra property of our mhe-RLWE, we can remove the need for the Ring-LHL in the security proof of our selective IPFE. Our security proof for the adaptive IPFE avoids the complex entropy discussion appeared in the previous works [4, 7, 39] and consequently improves the size of the public key.

Contribution 1. We present a ring version of mhe-LWE that we call mhe-RLWE. The mhe-RLWE problem is to distinguish two RLWE samples, given additional information on the secret and noise term through some hints of a special form. More precisely:

- The task of mhe-RLWE is to distinguish between the distributions

  $$(a, ar+f, (e_i, s_i, e_ir+g_i, s_if+h_i)_{i\in[\ell]}) \text{ and } (a, u, (e_i, s_i, e_ir+g_i, s_if+h_i)_{i\in[\ell]}).$$

  where a, u are uniformly sampled from R_q, polynomials r, f, g_i, h_i are sampled from Gaussian distributions, and s_i, e_i with $\|s_i\|_\infty, \|e_i\|_\infty \leq C$ are arbitrary polynomials with bounded coefficients.

In comparison with mhe-LWE, where hints are scalar products $\langle s_i, f\rangle$ with (high dimensional) vectors s_i sampled from a specific distribution τ, in mhe-RLWE hints are ring products of the form $s_i f + h_i$ with s_i arbitrary bounded elements of R_q and additional noise h_i is introduced. An important observation is that our mhe-RLWE not only includes hints over the noise but also over the secret, which makes it of independent interest and flexible to be used in more complex cryptosystems. Moreover, the reduction from mhe-LWE to LWE requires $m = \Omega(n \log n)$ samples, which directly affects the performance and the size of the keys in IPFE scheme, while no such requirement is needed in mhe-RLWE.

Intuitively, to prove the reduction from mhe-RLWE to RLWE, the main idea is that for a given RLWE sample $(a, b = ar + f)$ one can sample additional randomnesses r', f', g'_i, h'_i from specific distributions, so that $(a, b' = b + ar' + f', (e_i, s_i, e_i r' + g'_i, s_i f' + h'_i))$ has the right distribution to be submitted to the mhe-RLWE solver.

To show that the distribution obtained in this way is statistically close to the the one in the real game, we generalize a lemma expressing that the sum of two particular discrete Gaussian distributions (one on $\mathbb{Z}^n$ and the other one on a sub-lattice) is (close to) Gaussian. Intuitively, we define these distributions based on values e_i, jointly sample polynomials r', g'_i and use the mentioned lemma to show that hints $e_i r' + g'_i$ and simulated secret $r + r'$ have the right distribution (similarly for the hints over the error). The reduction is not trivial by itself as one needs to build the correct lattice allowing to apply the mentioned lemma.

The second required tool (to develop our RLWE-based IPFE scheme) is a ring version of LHL (Ring-LHL). Informally, in Ring-LHL the main goal is to show that the distribution $\sum_{i=1}^{k} a_i t_i \in \mathsf{R}_q$ is close to uniform when $\boldsymbol{a} = (a_1, \ldots, a_k)$ is fixed with a_i uniformly sampled from the ring and $\boldsymbol{t} = (t_1, \ldots, t_k)$ is sampled from a distribution with high min-entropy over the ring. In [38], authors presented a special case of Ring-LHL where $\boldsymbol{t}$ is sampled from a Gaussian distribution and no extra information is available.

For our RLWE-based IPFE, Ring-LHL is needed to show that $\sum_{i=1}^{k} a_i t_i$ is close to uniform even in the presence of additional information leaking on $\boldsymbol{t}$ through the public-key. While the result from [38] enjoys small entropy demands on values t_i and small value k, it can not handle the information leakage. On the other hand, the result from [25] is theoretically sufficient and can handle the leakage, however, it suffers from large parameters, specially the size of k (length of vector $\boldsymbol{a}$) is of order of the security parameter. There are still similar versions of Ring-LHL (such as [29]) but due to the need for clear and efficient choice of parameters, we propose a special version of Ring-LHL which manages to handle the information leaking from the public-key and still enjoys small parameters. In fact, we generalize the Ring-LHL version of [38] from $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{t}\rangle)$ to the matrix-coefficient $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{t})$, which is enough for our aim in the security proof of IPFE.

Contribution 2. Apart from relying on LWE, both schemes [4] and [7] require LHL to insert a uniform term in the ciphertext. We present two IPFE constructions based on RLWE, our first IPFE scheme is selectively-secure with smaller parameters, while our second scheme is adaptively-secure. The compactness of RLWE brings two benefits to our schemes: it not only improves the efficiency of encryption in general, but also allows for parallel encryptions while the computational-complexity does not grow by the number of encryptions. Technically, this means a single decryption returns a matrix-multiplication, rather than an inner-product value.

For each of our schemes we follow a somehow different proof technique. Particularly, in our first construction, for the sake of a higher efficiency, we avoid the use of Ring-LHL in the security proof. More precisely, in our selectively-secure IPFE (sel-IPFE) scheme, at the first step, we use mhe-RLWE which leads to the appearance of a term $u \cdot s_i$ in the ciphertext associated with the i-th slot, where $u \in \mathsf{R}_q$ is uniform and $s_i \in \mathsf{R}$ is the secret-key sampled from Gaussian distribution. Then in the second step, we change the structure of the secret-key in an indistinguishable way, which is only possible in the selective setting. This new structure allows us to remove the secret s_i from the functional-key, while it is still present in the public-key $\mathsf{pk}_i = as_i + e_i$. Having the noise term in the public-key and an extra noise in the ciphertext allow us to see s_i as the secret for two samples of RLWE in the public-key and in the ciphertext. Thus we rely on two samples of RLWE rather than relying on Ring-LHL.

For our adaptively secure IPFE, the first step is similar to the one in sel-IPFE while here $\boldsymbol{u}$ and $\boldsymbol{s}_i$ belong to R_q^m (vector-of-polynomials). Then we step back to the selective-game and change the structure of $\boldsymbol{s}_i$ to get rid of it in the functional-key. Interestingly, we have the freedom to come back to the adaptive-game via a mechanism similar to the Complexity Leveraging (CL) and without losing any factor of the security. The prominent observation here is that after stepping back to the selective-security, all of our upcoming games (in the sequence of the games) are statistically-indistinguishable, thanks to the use of Ring-LHL rather than RLWE assumption (unlike how we proceeded in our sel-IPFE). This means all these games can be upgraded to their adaptive versions by the correct setting of the parameters in the statistical arguments. The approach is similar to the one in [39] based on LWE assumption. But we manage to avoid a rather complicated entropy discussion, needed for their version of leftover hash lemma, since it results in a big parameter m reflected in the size of the public key. Instead, we indistinguishably change the generation of the secret key and remove it from the functional key.

This simplifies the proof, since we can use our simple extension of Ring-LHL for $\boldsymbol{A} = \left(\begin{smallmatrix} \boldsymbol{a} \\ \boldsymbol{u} \end{smallmatrix}\right)$ to replace $\boldsymbol{a}\boldsymbol{s}_i$ and $\boldsymbol{u}\boldsymbol{s}_i$ with uniform values, respectively, in the public-key and in the ciphertext. In Ring-LHL with $\boldsymbol{A} \in \mathsf{R}_q^{k \times m}$, the only condition on m is that $m \geq k + 1$, where in our case $k = 2$. Thus, we can consider $m = 3$, which means that in comparison with our sel-IPFE the size of the key increases only by a constant size. The use of Ring-LHL demands the variance of secrets to be greater than the one in the selective case, but still giving a reasonable efficiency.

In Fig. 1 we present a general comparison of our scheme with related works.

Contribution 3. We provide an efficient implementation to substantiate our claims of efficiency. Our scheme needs large polynomials where each coefficient can span multiple machine words. Further, the number of polynomial multiplications required in our inner-product functional scheme increases linearly with the length of the vectors. To overcome this, we provide a residue number system based implementation using Chinese remainder theorem and number theoretic transform based multiplication. We further show how the construction of the functional encryption scheme can be exploited to speed-up the multiplication. To reduce the risk of side-channel attacks we avoid all secret dependent branching and use a state-of-the-art constant-time discrete Gaussian sampler to generate error and secret polynomials. Finally, we show using a real-world use case that our work can be helpful for providing practical solutions for privacy-preserving machine learning applications.

	$\|\mathsf{mpk}\|$	$\|\mathsf{msk}\|$	$\|\mathsf{ct}\|$	$\|\mathsf{sk}_f\|$
ALS16 [7]	$O(n^2 \log^2 q + \ell n \log q)$	$O(\ell n \log^2 q)$	$O(n \log q^2 + \ell \log q)$	$O(n \log^2 q)$
ABDP15 [4]	$O((n+\ell) n \log^2 q)$	$O(\ell n \log q)$	$O((n+\ell) \log q)$	$O(n \log q)$
RLWE-FE	$O(\ell n \log q)$	$O(\ell n \log q)$	$O(\ell n \log q)$	$O(n \log q)$

	Setup	Encryption	KeyGen	Decryption
ALS16 [7]	$O(\ell n^2 \log q)$	$O(n^2 \log q + \ell n)$	$O(\ell n \log q)$	$O(n \log q + \ell)$
ABDP15 [4]	$O(\ell n^2 \log q)$	$O((\ell + n) n \log q)$	$O(\ell n)$	$O(\ell + n)$
RLWE-FE	$O(\ell n \log n)$	$O(\ell n \log n)$	$O(\ell n)$	$O(\ell n + n \log n)$

Fig. 1: Complexity comparison with related works. Upper and bottom part of the table respectively present the space and time complexity where the operations are in $\mathbb{Z}_q$. Value ℓ is the length of the message-vector, n and q are LWE or RLWE parameters. Since in our adaptively-secure FE scheme $m = 3$, all the above complexity arguments are the same for both of our schemes. However, other parameters, such as the choice of standard deviations, are different.

Contribution 4. In order to bring our IPFE scheme closer to the practical use, we extend our scheme to a (D)MCFE scheme without significantly increasing its complexity. In [6], the authors presented a general compiler to transfer a single-input IPFE to a multi-input IPFE[6]. Later in [3] it was argued that the resulting scheme is also secure against corruptions (removing the trust among the users), while still it does not support labels. In practice labels are needed to prevent undesired mixing of ciphertexts. Hence in this paper, we additionally develop a compiler which can transfer a multi-input FE supporting corruptions (but not labels) to a multi-client scheme (supporting labels). Similar outcome can be achieved with a compiler from [2], but with the ciphertext-size growing quadratically w.r.t the number of clients. On the other hand, one can built a

[6] Multi-input FE can be seen as a weaker version of MCFE where it may not support labels or corruptions.

MCFE based on RLWE in a non black-box way, similar to the MCFE scheme in [5] which is based on LWE problem. More precisely, the construction is based on LWE with rounding, which, intuitively, needs a bigger modulus q. Our compiler would not change the size of the modulus or the ciphertext.

The main idea of the construction in [6] is that, to encrypt a message $\mathbf{x}_i$ one indeed encrypts the message $\mathbf{x}_i + \boldsymbol{u}_i$ by the single-input IPFE scheme, where $\boldsymbol{u}_i$ is the secret key of user i. The functional key sk_f has two main parts one to apply the decryption of IPFE and the other one to remove terms involved with $\boldsymbol{u}_i$. Our compiler extends this idea to the labeled multi-client setting (in RO model) by adding another secret key $H(\boldsymbol{u}'_i, \gamma)$ such that the client i now encrypts $\mathbf{x}_i + \boldsymbol{u}_i + H(\boldsymbol{u}'_i, \gamma)$. This leads to what is known as identity-based MCFE where each functional key is associated with a label (or identity) γ as $\mathsf{sk}_{f,\gamma}$. Let ℓ be the number of clients, L be the number of issued labels and m be the number of different vectors $\mathbf{y}$ for which the functional key is issued. Then our scheme results in a joint ciphertext of size ℓL and a functional key of size mL, while the general compiler of [2] generates ciphertexts of size $\ell^2 L$ and functional key of size m. This means for the applications which ℓ is big, our scheme obtains a much better efficiency. This can include the applications such as aggregation and analyse of data from thousands of clients (health centers, data servers, etc.) during one year such that the data is processed daily (i.e., $n = 10000$ and $L = 365$)). Moreover, the fact that the functional key depends on γ can be seen as a kind of fine-grained access control, that can be use even data encrypted in parallel in one ciphertext.

In [3], authors present a general compiler to transfer a MCFE to a decentralized MCFE scheme, when the underlying scheme satisfies a special form of the functional key. More in details, at the setup phase the vector $\mathbf{0}$ is shared among the clients such that $\boldsymbol{v}_i$ is the secret key of the client i and $\sum \boldsymbol{v}_i = \mathbf{0}$. Then the functional key sk_f, which has a inner-product form $\mathsf{sk}_f = \sum_i \langle \boldsymbol{u}_i, \mathbf{y}_i \rangle$, is decentralized via generating $\langle \boldsymbol{u}_i, \mathbf{y}_i \rangle + \langle \boldsymbol{v}_i, \mathbf{y} \rangle$ by the i-th client. For our MCFE scheme, since the secret key $H(\boldsymbol{u}'_i, \gamma)$ depends on γ we can not directly apply their compiler over our scheme. To go around this problem we present a generalized distributed sum (GDSum) protocol which allows us to generate functions $H_i(\gamma)$ (depending on a label γ) such that $\sum H_i(\gamma) = 0$. We use GDSum as a building block to extend the compiler of [3] to an identity-based DMCFE. Finally, we show that our RLWE-based IPFE scheme has all the required properties to be used in our compilers, and so be extended to a identity-based MCFE or DMCFE scheme based on RLWE assumption.

2 Preliminaries

2.1 Notations

In this paper we shall denote with R a polynomial ring $\mathsf{R} = \mathbb{Z}[x]/\Phi$ where Φ is an irreducible polynomial. For the sake of simplicity (and implementation) Φ will be equal to $x^n + 1$, where n is a power of 2. We shall use a standard notation R_q to denote $\mathsf{R}/q\mathsf{R} = \mathbb{Z}_q[x]/\Phi$. The modulus q is chosen such that polynomial Φ of degree n factors into n distinct linear polynomials over $\mathbb{Z}_q$, i.e. $\Phi = \prod_i \phi_i$, where each ϕ_i is linear. Therefore, by Chinese Remainder Theorem (CRT), the ring R_q factors into n ideals and can be written as $\mathsf{R}_q \cong \prod_i \mathsf{R}_q/\phi_i$. Since each R_q/ϕ_i is isomorphic to $\mathbb{Z}_q$, this gives an isomorphism between R_q and $\mathbb{Z}_q^n$. The latter is specifically useful in the Ring-LHL argument, and consequently for our adaptively secure IPFE scheme. Moreover, if Φ factors as explained, then the multiplication of elements in R_q can be implemented particularly eficient in time $O(n \log n)$ using so called Fast Fourier Transform, which is important for a practical performance.

For $a \in \mathsf{R}$ (or $a \in \mathsf{R}_q$) a polynomial of degree less than n, we shall denote $\boldsymbol{a} \in \mathbb{Z}^n$ (or $\boldsymbol{a} \in \mathbb{Z}_q^n$) the vector of the coefficients of a, and vice versa. When the coefficients of a are sampled from some distribution χ we write $a \leftarrow \chi$. In this paper, $[\ell]$ stands for the set $\{1, \ldots, \ell\}$ and $\|\boldsymbol{v}\|_\infty$ and $||\boldsymbol{v}||$ stand for the infinity and Euclidean norm, respectively. We write $x \overset{R}{\leftarrow} X$ to show that the element x is sampled uniformly at random from the set X. The security parameter is denoted by κ (which is independent from parameters for RLWE problem).

2.2 Discrete Gaussian Distribution

In this section we give a definition of the discrete Gaussian distribution and present some results regarding it that will be used latter in the paper.

Definition 1. *A discrete Gaussian distribution $D_{\Lambda,\sqrt{\Sigma},\boldsymbol{c}}$, for $\boldsymbol{c} \in \mathbb{R}^n$, Σ a positive semi-definite matrix in $\mathbb{R}^{n\times n}$, and $\Lambda \subset \mathbb{Z}^n$ a lattice, is a distribution with values in Λ and probabilities*

$$\Pr(X = \boldsymbol{x}) \propto \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{e})^T \Sigma^+ (\boldsymbol{x} - \boldsymbol{e})).$$

Note that Σ^+ denotes the pseudoinverse of a matrix. If $\Lambda = \mathbb{Z}^n$ we shall write just $D_{\sqrt{\Sigma},\boldsymbol{c}}$. Furthermore, if $\boldsymbol{c} = 0$, then we shall write just $D_{\sqrt{\Sigma}}$, and if $\sqrt{\Sigma} = \sigma I_n$ for $\sigma \in \mathbb{R}^+$ and I_n an identity matrix, we write D_σ.

We define $\rho_B(\boldsymbol{x}) = \exp(-\boldsymbol{x}^T(BB^T)^{-1}\boldsymbol{x})$. It follows directly from the definition that for any invertible matrix β it holds $\rho_{\sqrt{\Sigma}}(\beta^{-1}\boldsymbol{x}) = \rho_{\beta\sqrt{\Sigma}}(\boldsymbol{x})$. For a lattice Λ we shall write $\rho_B(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda} \rho_B(\boldsymbol{x})$.

Discrete Gaussian distribution has many nice properties, for example: its samples can be easily bounded, and sampling from it is computationally feasible (see Appendix A.2). It is well known that the sum of continuous independent Gaussian distributions is also Gaussian. The following lemma discusses that the sum of *discrete* Gaussian variables is (close to) Gaussian under certain conditions over Gaussian parameters. A special case of this lemma was proved and used in [4].

Lemma 1. *Let $L(B) \subseteq \mathbb{Z}^n$ be a sub-lattice with dimension k whose basis is given by the columns of $(n \times k)$-matrix B. Let $\Sigma \in \mathbb{R}^{n\times n}$ be a positive definite matrix and define $\Sigma' = \sigma'^2 BB^T$. Then sampling $\boldsymbol{e}$ from a discrete Gaussian distribution $D_{\sqrt{(\Sigma+\Sigma')}}$ is indistinguishable from sampling $\boldsymbol{e} = \boldsymbol{e_1} + \boldsymbol{e_2}$, where $\boldsymbol{e_1}$ is sampled from $D_{\sqrt{\Sigma}}$ and $\boldsymbol{e}_2 \in L(B)$ is independently sampled from $D_{\sqrt{\Sigma'}}$,* *as long as the eigenvalues of* $\Gamma_{\Sigma,\Sigma'} := \sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B}$ *are greater than the smoothing parameter* $\eta_\epsilon(\mathbb{Z}^k)$*.*

Proof. Define

$$\Sigma'' = \begin{bmatrix} \Sigma & 0 \\ 0 & \sigma'^2 I_k \end{bmatrix}, \beta = \begin{bmatrix} I_n & B \end{bmatrix}, \beta' = \begin{bmatrix} I_n & B \\ X^T & I_k + X^T B \end{bmatrix}, X = -\sigma'^2(\Sigma+\Sigma')^{-1}B$$

Defining $\Sigma''' = (\beta'\sqrt{\Sigma''})(\beta'\sqrt{\Sigma''})^T$ we have by a simple calculation

$$\Sigma''' = \begin{bmatrix} \Sigma+\Sigma' & 0 \\ 0 & \sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B \end{bmatrix}.$$

Let $\boldsymbol{e_1}$ be sampled from $D_{\sqrt{\Sigma}}$ and $\boldsymbol{e_2}$ be sampled from $D_{\sigma' I_k}$. Let $\boldsymbol{e} = \boldsymbol{e_1} + B\boldsymbol{e_2}$. Notice that sampling $\boldsymbol{e}_3 \in L(B)$ from $D_{\sqrt{\Sigma'}}$ is by definition equivalent to sampling $B\boldsymbol{e_2}$ where $\boldsymbol{e_2}$ is sampled from $D_{\sigma' I_k}$. Let $\boldsymbol{e'} = \begin{bmatrix} \boldsymbol{e_1} \\ \boldsymbol{e_2} \end{bmatrix}$, and notice that $\boldsymbol{e'}$ is sampled from $D_{\sqrt{\Sigma''}}$. Now

$$\begin{aligned}
\Pr(\boldsymbol{e} = \boldsymbol{z}) &= \Pr(\beta\boldsymbol{e'} = \boldsymbol{z}) \\
&= \sum_{\boldsymbol{s}\in\mathbb{Z}^k} \Pr(\beta'\boldsymbol{e'} = \begin{bmatrix} \boldsymbol{z} \\ X^T\boldsymbol{z} + \boldsymbol{s} \end{bmatrix} = \sum_{\boldsymbol{s}\in\mathbb{Z}^k} \Pr(\boldsymbol{e'} = \beta'^{-1} \begin{bmatrix} \boldsymbol{z} \\ X^T\boldsymbol{z} + \boldsymbol{s} \end{bmatrix} \\
&\propto \sum_{\boldsymbol{s}\in\mathbb{Z}^k} \rho_{\sqrt{\Sigma''}}(\beta'^{-1} \begin{bmatrix} \boldsymbol{z} \\ X^T\boldsymbol{z} + \boldsymbol{s} \end{bmatrix}) \propto \sum_{\boldsymbol{s}\in\mathbb{Z}^k} \rho_{\beta'\sqrt{\Sigma''}}(\begin{bmatrix} \boldsymbol{z} \\ X^T\boldsymbol{z} + \boldsymbol{s} \end{bmatrix}) \\
&\propto \sum_{\boldsymbol{s}\in\mathbb{Z}^k} \rho_{\sqrt{\Sigma+\Sigma'}}(\boldsymbol{z})\rho_{\sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B}}(X^T\boldsymbol{z} + \boldsymbol{s}) \\
&\propto \rho_{\sqrt{\Sigma+\Sigma'}}(\boldsymbol{z})\rho_{\sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B}}(X^T\boldsymbol{z} + \mathbb{Z}^k) \\
&\propto \rho_{\sqrt{\Sigma+\Sigma'}}(\boldsymbol{z})\rho_{\sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B}}(\mathbb{Z}^k)\mu_z \qquad \text{by Lemma 4} \\
&\propto \rho_{\sqrt{\Sigma+\Sigma'}}(\boldsymbol{z})\mu_z, \text{ for } \mu_z \in [\frac{1-\epsilon}{1+\epsilon}, 1]
\end{aligned}$$

Where Lemma 4 can be applied as long as the eigenvalues of matrix $\Gamma_{\Sigma,\Sigma'} > \eta_\epsilon(\mathbb{Z}^k)$, where $\Gamma_{\Sigma,\Sigma'} := \sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B}$. □

We shall be using Lemma 1 in the following cases. We will have $\Sigma = \sigma^2 I_n - \sigma'^2 BB^T, \Sigma' = \sigma'^2 BB^T$ so that $\Sigma + \Sigma' = \sigma^2 I_n$. Then

$$\sqrt{\sigma'^2 I_k - \sigma'^4 B^T(\Sigma+\Sigma')^{-1}B} = \sigma'\sqrt{I_k - \frac{\sigma'^2}{\sigma^2}BB^T}$$

which is $> \eta_\epsilon(\mathbb{Z}^k)$ for example if $\sigma^2 = 2||\sigma'^2 BB^T||$ and $\sigma' > 2\eta_\epsilon(\mathbb{Z}^k)$, but more specific bounds can be derived as well.

2.3 RLWE problem

In the seminal work [28], the authors introduced RLWE problem and study its hardness. In the following we define RLWE problem, while one can consult Theorem 6 in the Appendix A.3 or [28] for the choice of the parameters in the reduction from SIVP, a standard hard lattice-problem, to RLWE.

Definition 2 ((Decisional) RLWE[7]). *The Ring Learning With Errors problem, w.r.t the ring* R_q *and the distribution* D_σ*, is to distinguish between two following distributions with the secret* $s \leftarrow D_\sigma$ *fixed for all the samples,*

$$D = \{(a, as+e) \ : \ a \xleftarrow{R} \mathsf{R}_q,\ e \leftarrow D_\sigma\}, \qquad D' = \{(a,u) \ : \ a,u \xleftarrow{R} \mathsf{R}_q\}$$

2.4 Functional Encryption

This section discusses the syntax of a FE scheme and its security notion.

Definition 3 (Functional Encryption scheme). *A FE scheme parameterized by* $\rho = (X, Y, Z, f)$ *for functionality* $f : X \times Y \to Z$*, is defined by four following algorithms.*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa)$*: where* Setup *receives security parameter* κ*, and returns a pair of master public and secret key. The public-key implicitly defines the functionality-parameter* ρ*.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$*: where* Enc *receives the master public-key* mpk *and a message* $\mathbf{x} \in X$*, and it returns a ciphertext* ct*.*
- $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$*: where* KeyGen *receives the master secret-key* msk *and function* $\mathbf{y} \in Y$*, then it returns a functional-key* sk_y*.*
- $Y := \mathsf{Dec}(\mathsf{ct}, \mathsf{sk})$*: it receives a ciphertext* ct *and a functional-key* sk*, and returns* $\perp$ *or a value in the range of* f*.*

Correctness. For a correct execution of the above encryption system, $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}_F)$ would return $f_\mathbf{y}(\mathbf{x})$ with overwhelming probability, where $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$ and $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$. For the inner-product functionality we have $f_\mathbf{y}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i \in [\ell]} x_i y_i$, where $\mathbf{x} \in \mathcal{M}_1^\ell, \mathbf{y} \in \mathcal{M}_2^\ell$ for some $\mathcal{M}_1^\ell, \mathcal{M}_2^\ell$ message and function space.

Security Notion. Following the standard security notion for FE [4, 11], the game $\mathsf{IND}_\mathcal{A}^b(1^\kappa)$ between the adversary $\mathcal{A}$ and challenger is defined as follows, where $b \xleftarrow{R} \{0,1\}$.

- *Initialize:* The challenger runs $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\kappa)$ and send mpk to $\mathcal{A}$.
- *Query*: The adversary adaptively submits queries $\mathbf{y}$ and receives the response $\mathsf{sk}_y = \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$ from the challenger.
- *Challenge*: The adversary submits messages $\mathbf{x}^0, \mathbf{x}^1$, the challenger runs $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathbf{x}^b)$ and returns it to $\mathcal{A}$. The challenge should satisfy the constraint $f_\mathbf{y}(\mathbf{x}^0) = f_\mathbf{y}(\mathbf{x}^1)$ for all the previously issed queries $\mathbf{y}$.

[7] Here we have considered a special form of RLWE which would be used in this paper.

- *Query*: The adversary adaptively submits queries $\mathbf{y}$ and receives the response $\mathsf{sk}_y = \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$, where the queries $\mathbf{y}$ should satisfy the constraint $f_{\mathbf{y}}(\mathbf{x}^0) = f_{\mathbf{y}}(\mathbf{x}^1)$.
- *Finalize*: The adversary outputs a bit b' as its guess for the bit b.

We say a FE scheme is (adaptively) indistinguishable-secure (IND-secure), if for any PPT adversary $\mathcal{A}$ there is a negligible function negl such that,

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}}(\mathsf{IND}^b_{\mathcal{A}}) = |\Pr[\mathsf{IND}^1_{\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{IND}^0_{\mathcal{A}}(1^\kappa) = 1]| \leq \mathrm{negl}(\kappa)$$

Moreover, we say that a FE scheme is selectively secure, if the adversary submits its challenges $(\mathbf{x}^0, \mathbf{x}^1)$ at the very beginning of the game before seeing the public-key.

2.5 Multi-Client FE

In a MCFE scheme data comes from different clients and each client encrypts its data individually. Here we present the standard syntax of MCFE scheme and then clarify its identity-based version.

Definition 4 (Multi-Client Functional Encryption). *Let f be a functionality (indexed by ρ), and $\mathsf{Labels} = \{0,1\}^*$ or $\{\perp\}$ be a set of labels. A multi-client functional encryption scheme (MCFE) for the functionality f and the label set Labels is a tuple of four algorithms $\mathsf{MCFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:*

$\mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$**:** *Takes as input a security parameter κ, the number of clients ℓ, vectors dimension k and generates public parameters pp. The public parameters implicitly define the functionality-index ρ. It outputs ℓ secret-keys $\{\mathsf{ek}_i\}_{i\in[\ell]}$, the master secret-key $\mathsf{msk} = \{\mathsf{ek}_i\}_{i\in[\ell]}$ and pp (all other algorithms take public parameters pp).*

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$**:** *Takes the master secret-key msk and a function $\mathbf{y}$, and outputs a functional-key sk_y.*

$\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \gamma)$**:** *it receives the secret key ek_i and a label $\gamma \in \mathsf{Labels}$ and the message $\mathbf{x}_i \in \mathcal{M}^k$ to encrypt, it outputs the ciphertext $\mathsf{ct}_{i,\gamma}$.*

$\mathsf{Dec}(\mathsf{sk}_y, \mathsf{ct}_{1,\gamma}, \ldots, \mathsf{ct}_{\ell,\gamma})$**:** *Takes as input a functional-key sk_y and ℓ ciphertexts $\mathsf{ct}_{i,\gamma}$ under the same label γ and outputs $\perp$ or a value in range f.*

A MCFE scheme is correct, if for all $\kappa, \ell, k \in \mathbb{N}$, functionality f, $\gamma \in \mathsf{Labels}$, messages $\mathbf{x}_i$, when $(\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in\ell}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$, $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$,and $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, \gamma)$ we have

$$\Pr[\mathsf{Dec}(\mathsf{sk}_y, \{\mathsf{ct}_{i,\gamma}\}_{i\in[\ell]}) = f_y(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)] = 1.$$

If the algorithm KeyGen receives the label γ as input, we call the scheme an *identity-based MCFE scheme*, where the functional key can be applied only over the ciphertexts which share the same identity used in the functional key. Indeed, here the identity is the label.[8]

The security notion allows encryption queries on each individual slot i and the adversary can corrupt chosen clients, while the privacy of uncorrupted clients is still preserved. The details can be found in Appendix D.1.

3 New results on ideal lattices

In this section we present our new results on lattices which are used in the security proof of our IPFE constructions and might be of independent interest.

3.1 Multi-hint extended RLWE problem

We define a variant of the RLWE problem where additional information about the secrets and the noise is given through some hints. These hints are of the form $e_i r + g_i$ and $s_i f + h_i$, where $e_i, s_i \in \mathsf{R}$ are arbitrary, but with bounded norm $||\boldsymbol{s}_i||_\infty, ||\boldsymbol{e}_i||_\infty \leq C$ for some $C > 0$, and g_i, h_i are sampled from the same distribution as r and f. We give a formal definition below.

Definition 5 (multi-hint extended RLWE (mhe-RLWE)). *Let $s_i, e_i \in \mathsf{R}$ be arbitrary such that $||\boldsymbol{s}_i||_\infty, ||\boldsymbol{e}_i||_\infty \leq C$ for some $C > 0$, and fixed by the adversary in advance. Assume that $a, u \in \mathsf{R}_q$ are uniformly sampled, and $r, f, g_i, h_i \in \mathsf{R}_q$ sampled from $D_{\delta I_n}$ for $i \in [l]$, all by the challenger. The multi-hint extended RLWE problem is to distinguish the tuples*

$$(a, ar + f, (e_i, s_i, e_i r + g_i, s_i f + h_i)_{i\in[l]}) \text{ and } (a, u, (e_i, s_i, e_i r + g_i, s_i f + h_i)_{i\in[l]}).$$

We prove that, for properly chosen parameters, mhe-RLWE problem is at least as hard as the standard RLWE problem. Note that its hardness depends on the choice of R_q (implicitly on n and q), bound C and Gaussian parameter δ. Values s_i, e_i can be chosen arbitrary and if $s_i = e_i = 0$ for all $i \in [l]$, then the problem corresponds to the standard RLWE problem.

Theorem 1. *Let R_q, σ be such that the RLWE problem in R_q is hard, assuming the secret and errors are sampled from $D_{\sigma I_n}$. Then mhe-RLWE problem with bound C and Gaussian parameter δ is hard, when $\sigma\sqrt{1 - \frac{1}{\delta^2}(\sigma n C\sqrt{l+2})^2} > \eta_\epsilon(\mathbb{Z}^{n+nl})$.*

Proof. We start by analyzing the distributions of the variables in the definition. Let $\varDelta$ be a $(n + nl) \times (n + nl)$ diagonal matrix with values δ^2 on the diagonal, i.e. $\varDelta = \delta^2 I_{n+ln}$. Sampling r, g_i from $D_{\delta I_n}$ is by definition indistinguishable from sampling a vector $(\boldsymbol{r}, \boldsymbol{g_1}, \ldots, \boldsymbol{g_l})$ from $D_{\sqrt{\varDelta}}$.

Each multiplication $T_{e_i}(x) = e_i x \in \mathsf{R}$ for $e_i, x \in \mathsf{R}$ (as a linear function from R to R) can be represented as a matrix multiplication $E_i\boldsymbol{x}$ (and thus a liner

[8] The syntax of MIFE without label is defined similarly removing the labels from the syntax of MCFE.

function from $\mathbb{Z}^n$ to $\mathbb{Z}^n$) for some matrix E_i of dimension $n \times n$, independent of x. Let $\bar{\Lambda}$ be a subspace of $\mathbb{R}^{n+nl}$ defined on all the vectors $\boldsymbol{v} = (\boldsymbol{r}, -E_1\boldsymbol{r}, \dots, -E_l\boldsymbol{r})$ for arbitrary $\boldsymbol{r} \in \mathsf{R}$. Then $\Lambda = \mathbb{Z}^{n+nl} \cap \bar{\Lambda}$ is precisely the sub-lattice of all vectors $(\boldsymbol{r}, \boldsymbol{g_1}, \dots, \boldsymbol{g_l})$ for which the hints $e_i r + g_i = 0$.

Then elements of Λ can be written as $L\boldsymbol{r}$ for $r \in \mathsf{R}$, where L is a matrix of dimension $(n + nl) \times n$ as follows:

$$L = \begin{bmatrix} I \\ -E_1 \\ -E_2 \\ \vdots \\ -E_l \end{bmatrix}$$

When r is sampled from a Gaussian distribution $D_{\sigma I_n}$, the distribution of vector $L\boldsymbol{r}$ is $D_{\Lambda,\sqrt{B}}$, where the positive semi-definite matrix associated with Λ is defined as $B = \sigma^2 LL^T$.

Now we define matrix $A = \Delta - B$, that will be later used as a Gaussian parameter. We claim that matrix A is positive semi-definite, assuming the bounds from the theorem hold.

We use the following result to prove A is positive semi-definite for a proper choice of parameters. Recall that a matrix is $X = [x_{ij}]$ is diagonally dominated if $|x_{ii}| \geq \sum_{j \neq i} |x_{ij}|$ for any i. By a classical result from linear algebra, if a symmetric matrix X with real components is diagonally dominated, then A is positive semi-definite. Since A is symmetric with real components, it is enough to prove that A is diagonally dominated and the claim follows. Note that by the condition $\|e_i\|_\infty \leq C$ we have $\|E_iE_j\|_\infty \leq nC^2$, meaning that each component of E_iE_j is bounded by nC^2. By the definition of $A = \Delta - B$, we have $|A_{ii}| \geq \delta^2 - \sigma^2 nC^2$ and $\sum_{j \neq i} |A_{ij}| \leq \sigma^2(l-1)n^2C^2 + \sigma^2(n-1)nC^2 + \sigma^2 nC \leq \sigma^2 n^2 C^2 (l+1)$. Thus if $\delta \geq \sigma nC\sqrt{l+2}$ the matrix A is a diagonally dominated matrix. The assumption $\sigma\sqrt{1 - \frac{1}{\delta^2}(\sigma nC\sqrt{l+2})^2} > \eta_\epsilon(\mathbb{Z}^{n+nl})$ implies the latter.

A similar analysis can be made for vectors $(\boldsymbol{f}, \boldsymbol{h}_1, \dots, \boldsymbol{h}_l)$ that are also chosen from a Gaussian distribution with matrix parameter Δ. We would get positive semi-definite matrices A' and B' such that $A' = \Delta - B'$ and elements sampled from B' are in the sub-lattice of vectors of the form $(\boldsymbol{f}, -S_1\boldsymbol{f}, \dots, -S_l\boldsymbol{f})$ with probability as if $\boldsymbol{f}$ was sampled from $D_{\sigma I_n}$, where S_i is a matrix representation of s_i.

Now, we are ready to reduce the security of mhe-RLWE to the security of the RLWE problem. Let $\mathcal{A}$ and $\mathcal{B}$ be the adversary respectively to the problem mhe-RLWE and RLWE. Assume the adversary $\mathcal{B}$ is given a RLWE sample (a, b), where b is either uniformly sampled or calculated as $b = ar + f$, where r, f are sampled from $D_{\sigma I_n}$. We show how the adversary $\mathcal{B}$ uses the adversary $\mathcal{A}$ to win its game.

The adversary $\mathcal{A}$ chooses arbitrary $e_i,\ s_i$ such that $\|e_i\|_\infty, \|s_i\|_\infty \leq C$, $i \in [l]$ and gives them to $\mathcal{B}$. Based on $e_i,\ s_i$, the adversary $\mathcal{B}$ samples $(\boldsymbol{r}', \boldsymbol{g}'_1, \dots, \boldsymbol{g}'_l)$ from $D_{\sqrt{A}}$ and $(\boldsymbol{f}', \boldsymbol{h}'_1, \dots, \boldsymbol{h}'_l)$ from $D_{\sqrt{A'}}$ (as described above). Then it calculates $b' = b + ar' + f'$ as the sample and $e_i r' + g'_i$, and $s_i f' + h'_i$ as hints, for $i \in [l]$

and sends them to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit β as its guess, $\mathcal{B}$ outputs the same bit β.

If b was chosen uniformly at random, the distribution of b' is uniformly random. In the other case, $b' = a(r + r') + (f + f')$. To finish the proof we need to confirm that the distributions of b' and the hints are indistinguishable from the ones defined for mhe-RLWE.

Define $r^* = r + r'$, $f^* = f + f'$, $g_i = -e_i r + g_i'$, and $h_i = -s_i f + h_i'$. Note that this values are needed only to argue about the distributions of secrets and hints and are not known to $\mathcal{B}$, since r and f were chosen by the RLWE challenger. More precisely, if b in RLWE challenge was chosen uniformly at random, one can think of r and f as arbitrary sampled from $D_{\delta I_n}$. Since r is sampled from $D_{\sigma I_n}$, the distribution of vector $(\boldsymbol{r}, -\boldsymbol{e_1 r}, \dots, -\boldsymbol{e_l r})$ is as if it was sampled from $D_{\sqrt{B}}$. On the other hand, the vector $(\boldsymbol{r'}, \boldsymbol{g'}_1, \dots, \boldsymbol{g'}_l)$ is sampled from $D_{\sqrt{A}}$. Since A and B are positive semi-definite and $A + B = \varDelta$, Lemma 1 implies that the distribution of $(r + r', g_1, \dots, g_l)$ is indistinguishable from being sampled from $D_{\varDelta}$, which is the same as the distribution we have in the assumption. In fact, Lemma 1 can be applied since $\varGamma_{A,B} = \sigma\sqrt{I_{n+nl} - \frac{\sigma^2}{\delta^2} LL^T} \geq \sigma\sqrt{1 - \frac{1}{\delta^2}(\sigma n C\sqrt{l+2})^2} > \eta_\epsilon(\mathbb{Z}^{n+nl})$, by assumption.

A similar arguments show that $(f + f', h_1, \dots, h_l)$ are also indistinguishable from being sampled from $D_{\varDelta}$.

Since $b' = a(r + r') + (f + f') = ar^* + f^*$ this shows that b' has the right distribution. On the other hand,

$$\begin{aligned} e_i r^* + g_i &= e_i(r + r') - e_i r + g_i' = e_i r' + g_i' \\ s_i f^* + h_i &= s_i(f + f') - s_i f + h_i' = s_i f' + h_i'. \end{aligned}$$

Thus also the hints have the right distribution, and even though g_i and h_i are defined w.r.t. r and f, the hints are independent of r and f. This finishes the proof. $\square$

3.2 Leftover Hash Lemma in rings

Let $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ be a $k \times m$ matrix with elements from R_q. The goal of this section is to show that, with properly chosen parameters, the distribution of values $\boldsymbol{At} \in \mathsf{R}_q^k$, where $\boldsymbol{t} \in \mathsf{R}_q^m$ comes from a discrete Gaussian distribution, is close to uniform. This will be an important building block in designing an adaptively secure IPFE scheme in Section 5, but might as well be of an independent interest. Our result generalizes the result in [38], from $k = 1$ to an arbitrary k. We follow closely the ideas as well as notation used in [38].

Theorem 2. *Let n be a power of 2 such that $\varPhi = x^n + 1$ splits into n linear factors modulo a prime q. Let $k \geq 1, m \geq 1 + k, \epsilon > 0, \delta \in (0, 1/2)$ and $\boldsymbol{t} \in \mathsf{R}_q^m$ sampled from $D_{\mathbb{Z}^{mn},\sigma}$ with $\sigma \geq \sqrt{n \ln(2mn(1 + 1/\delta))/\pi}\, q^{\frac{k}{m}+\frac{\epsilon}{k}}$. Then except for at most a fraction of $2^n q^{-\epsilon n}(\frac{q^{mk}}{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})})^n$ of all $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})^*$ the* *distance to the uniformity of* $\boldsymbol{At} = (\sum_{i=1}^{m} a_{1,i}t_i, \ldots, \sum_{i=1}^{m} a_{k,i}t_i)$ *is* $\leq 2\delta$. *This implies,*

$$\Delta\big[\boldsymbol{A}, \boldsymbol{At}; U((\mathsf{R}_q^{k\times m})^*, \mathsf{R}_q^k)\big] \leq 2\delta + 2^n q^{-\epsilon n}\left(\frac{q^{mk}}{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})}\right)^n$$

We have provided all the required lemmas and the proof of this theorem in Appendix B.

4 Selectively-secure IPFE based on RLWE

Our IPFE construction is inspired by the LWE-based IPFE schemes from [4,7], but here we rely on the RLWE assumption to improve the efficiency. Our construction allows to encrypt ℓ-dimensional non-negative vectors, where infinity norms of the message $\mathbf{x}$ and the function $\mathbf{y}$ are bounded by B_x and B_y, respectively. We let K be greater than the maximal value of the resulting inner product i.e., $K > \ell B_x B_y$. We first describe the construction and postpone the parameters-setting, required for the correctness and the security, to Section 4.2.

Construction:

- **Setup:** We sample uniformly at random $a \in \mathsf{R}_q$ and elements $\{s_i \in \mathsf{R} \mid i \in [\ell]\}, \{e_i \in \mathsf{R} \mid i \in [\ell]\}$ from D_{σ_1}. Then $\mathsf{msk} = \{s_i \mid i \in [\ell]\}$ is the master secret-keys and the public-key is $\mathsf{mpk} = (a, \{\mathsf{pk}_i \mid i \in [\ell]\})$, where $\mathsf{pk}_i = as_i + e_i \in \mathsf{R}_q$.
- **Encryption:** To encrypt a vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}^\ell$ with $\|\mathbf{x}\|_\infty \leq B_x$ we sample polynomials r and $f_0 \in \mathsf{R}_q$ from D_{σ_2}, and polynomials $\{f_i \in \mathsf{R}_q \mid i \in [\ell]\}$ independently from D_{σ_3}. We fix 1_R to be the identity element of R_q (or it can be a polynomial of degree $n-1$ with all coefficients equal $1 \in \mathbb{Z}_q$) and calculate: $\mathsf{ct}_0 = ar + f_0 \in \mathsf{R}_q$, $\mathsf{ct}_i = \mathsf{pk}_i r + f_i + \lfloor q/K \rfloor x_i 1_\mathsf{R} \in \mathsf{R}_q$. Then $(\mathsf{ct}_0, \{\mathsf{ct}_i\}_{i\in[\ell]})$ is the encryption of $\mathbf{x}$.
- **KeyGen:** To generate a decryption key associated with $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}^\ell$ such that $\|y\|_\infty < B_y$, we calculate $\mathsf{sk}_y = \sum_{i=1}^{\ell} y_i s_i \in \mathsf{R}$.
- **Decryption:** To decrypt $(\mathsf{ct}_0, \{\mathsf{ct}_i\}_{i\in[\ell]})$ using sk_y and $\mathbf{y}$ we calculate $d = (\sum_{i=1}^{\ell} y_i \mathsf{ct}_i) - \mathsf{ct}_0 \mathsf{sk}_y \mod \mathsf{R}_q$. Then d should be close to $\lfloor q/K \rfloor \langle \mathbf{x}, \mathbf{y} \rangle 1_\mathsf{R}$ (a bit perturbed coefficients) and we can extract $\langle \mathbf{x}, \mathbf{y} \rangle$.

Correctness. We can write d as follows, by replacing ciphertexts and the functional key.

$$d = \sum_i (y_i e_i r + y_i f_i + f_0 y_i s_i) + \lfloor q/K \rfloor x_i y_i 1_\mathsf{R} = \mathsf{noise} + \lfloor q/K \rfloor \langle x, y \rangle 1_\mathsf{R}$$

For the correctness we need $\|\mathsf{noise}\|_\infty < \lfloor q/2K \rfloor$. By Lemma 2 in the Appendix A.2 for the security parameter κ, with overwhelming probability we have, $\|e_i\|_\infty, \|s_i\|_\infty \leq \sqrt{\kappa}\sigma_1$, also $\|r\|_\infty, \|f_0\|_\infty \leq \sqrt{\kappa}\sigma_2$ and $\|f_i\|_\infty \leq \sqrt{\kappa}\sigma_3$.Thus,

$$\left\| \sum_i y_i(e_i r + f_i + f_0 s_i) \right\|_\infty < \ell(2n\kappa\sigma_1\sigma_2 + \sqrt{\kappa}\sigma_3)B_y$$

Game	Description	justification
$\mathbf{G}_0$	$s_i \xleftarrow{R} D_{\sigma_1}$ $e_i \xleftarrow{R} D_{\sigma_1}$ $\mathsf{pk}_i = as_i + e_i$ $\mathsf{ct}_0 = ar + f_0$ $\mathsf{sk} = \sum_i y_i s_i$ $\mathsf{ct}_i = \mathsf{pk}_i r + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Real Game
$\mathbf{G}_1$	$s_i \xleftarrow{R} D_{\sigma_1}$ $e_i \xleftarrow{R} D_{\sigma_1}$ $\mathsf{pk}_i = as_i + e_i$ $\mathsf{ct}_0 = ar + f_0$ $\mathsf{sk} = \sum_i y_i s_i$ $\mathsf{ct}_i = \mathsf{ct}_0 s_i - f_0 s_i + e_i r + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Identical
$\mathbf{G}_2$	$\mathsf{pk}_i = as_i + e_i$ $\boxed{\mathsf{ct}_0 = u + ar + f_0}$ $\mathsf{sk} = \sum y_i s_i$ $\mathsf{ct}_i = \mathsf{ct}_0 s_i - f_0 s_i + e_i r + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	mhe-RLWE
$\mathbf{G}_3$	$\mathsf{pk}_i = as_i + e_i$ $\mathsf{ct}_0 = u + ar + f_0$ $\mathsf{sk} = \sum y_i s_i$ $\mathsf{ct}_i = \mathsf{pk}_i r + us_i + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Identical
$\mathbf{G}_4$	$\boxed{s_i = s^* \alpha_i + s'_i}$ $\boxed{f_i = f^* \alpha_i + f'_i}$ $\boxed{e_i = e^* \alpha_i + e'_i}$, $\alpha_i = (x_i^1 - x_i^0)$ $\mathsf{pk}_i = \mathsf{pk}_i = as_i + e_i$ $\mathsf{ct}_0 = u + ar + f_0$ $\mathsf{sk} = \sum_i y_i s_i$ $\mathsf{ct}_i = \mathsf{pk}_i r + us_i + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Stati. argu.
$\mathbf{G}_5$	$\mathsf{ct}_0 = u + ar + f_0$ $\mathsf{pk}_i = (as^* + e^*)\alpha_i + as'_i + e'_i$ $\mathsf{ct}_i = (as^* + e^*)r + (us^* + f^*)\alpha_i +$ $\mathsf{sk} = \sum_i y_i s'_i$ $(as'_i + e'_i)r + us'_i + f'_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Identical.
$\mathbf{G}_6$	$\mathsf{ct}_0 = u + ar + f_0$ $\mathsf{pk}_i = \boxed{u'}\alpha_i + as'_i + e'_i$ $\mathsf{ct}_i = \boxed{u'}r + \boxed{u''}\alpha_i +$ $\mathsf{sk} = \sum_i y_i s'_i$ $(as'_i + e'_i)r + us'_i + f'_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	RLWE independent of b

Fig. 2: Overview of games for selectively-secure IPFE.

Meaning that for the correctness we need $\ell(2n\kappa\sigma_1\sigma_2 + \sqrt{\kappa}\sigma_3)B_y < \lfloor q/2K \rfloor$.

4.1 Security proof

The following theorem proves the selective security of our construction. For the proof, we first rewrite ct_i based on ct_0 simply by replacing pk_i with its value $as_i + e_i$. This leads to the appearance of the term $\mathsf{ct}_0 s_i$ in the ciphertext, alongside some leakages on r and f_0. We try to formulate these leakages as the hints in the mhe-RLWE assumption, which from there by applying mhe-RLWE, we manage to replace $\mathsf{ct}_0 s_i$ with us_i for a uniform polynomial u. Note that s_i is appearing in the public-key, ciphertext and also the functional-key. To remove this term in the public-key and the ciphertext, one can see s_i as the secret for RLWE samples (with a, u as the coefficients) together with the noise terms present in the public-key and the ciphertext. Thus intuitively, all we need is to remove s_i from the functional-key (mainly because there is no error term in the functional-key, we cannot see s_i as the secret for RLWE samples here). For this, we (indistinguishably) change the structure of s_i to $s^*(x_i^1 - x_i^0) + s'_i$ allowing to remove s^* from the functional-key (thanks to the constraint $\langle \mathbf{y}, \mathbf{x}^1 - \mathbf{x}^0 \rangle = 0$) and looking at s^* as the secret for two samples of RLWE appearing in the ciphertext and in the public-key. This means a uniform term appears in the ciphertext which hides the bit b.

Theorem 3. *The IPFE scheme from Section 4 is sel-IND secure, for a proper choice of parameters (see Section 4.2). More precisely,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FE}}(\text{sel-}IND_{\mathcal{A}}^b) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{mheRLWE}}(\kappa) + \mathsf{Adv}_{\mathcal{B}'}^{\mathsf{RLWE}} + \text{negl}(\kappa).$$

where negl *comes from a statistical arguments.*

Proof. We define the following sequence of the games which are also summarized in Fig. 2. The first game is the real game associated with bit b, while the last game is independent of bit b. We will show that each two adjacent games are indistinguishable. Then since the last game is independent of b, the advantage of the adversary in the real game is negligible. The formal descriptions of games is given as follows.

$\boxed{\mathbf{G}_0}$: is the real game associated with the bit $b \overset{R}{\leftarrow} \{0,1\}$.

$\boxed{\mathbf{G}_1}$: is the same as game $\mathbf{G}_0$ when ct_i is computed using ct_0 (by replacing pk_i with $as_i + e_i$). Namely, $\mathsf{ct}_i = \mathsf{ct}_0 s_i - f_0 s_i + e_i r + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$.
Clearly, $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_0}(\kappa) = \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_1}(\kappa)$

$\boxed{\mathbf{G}_2}$: is similar to the game $\mathbf{G}_1$ except that $\mathsf{ct}_0 = ar + f_0$ is replaced with $\mathsf{ct}_0 = u + ar + f_0$ for a uniformly sampled $u \in R_q$.
Here we rely on the mhe-RLWE assumption. The hints of the mhe-RLWE problem are leaked through values ct_i where we replace f_i with $g_i - h_i$ where h_i and g_i are sampled from the same distribution $D_{\delta I_n}$. This is possible if in Lemma 1 the positive definite matrices $\Sigma = \Sigma' = \delta I_n$ satisfy the condition $\Gamma_{\Sigma,\Sigma'} \geq \eta_\epsilon(\mathbb{Z}^n)$ for $\epsilon = 2^{-k}$. Meaning that we should set $\sigma_3 = \sqrt{2}\delta$ where δ is such that the mhe-RLWE assumption holds and also satisfies $\Gamma_{\delta I_n, \delta I_n} \geq \eta_\epsilon(\mathbb{Z}^n)$. So, by these conditions,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_2}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_1}(\kappa)| \leq \mathsf{Adv}^{\mathsf{mheRLWE}}_{\mathcal{B}}(\kappa) + 2\epsilon.$$

$\boxed{\mathbf{G}_3}$: is the same as game $\mathbf{G}_2$ when ct_i is computed using pk_i (instead of ct_0). Namely, $\mathsf{ct}_i = \mathsf{pk}_i r + u s_i + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}, \quad \mathsf{ct}_0 = u + ar + f_0$.
Obviously, $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_3}(\kappa) = \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_2}(\kappa)$

To proceed to the next game, we first define the matrices $\boldsymbol{S}$, $\boldsymbol{E}$ and $\boldsymbol{F}$. Recall that the master secret-key is a vector of polynomials $(s_1, \ldots, s_\ell)$ where each polynomial is in R_q. This means one call represent the master secret-key via a matrix $\boldsymbol{S}$ of dimension $\ell \times n$, where the i-th row is the vector-representation of polynomial s_i i.e., $\boldsymbol{S} = \begin{pmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_\ell \end{pmatrix}$. We shall call $\bar{\boldsymbol{s}}_j$ the j-th column of matrix $\boldsymbol{S}$. Similarly matrices $\boldsymbol{E}$ and $\boldsymbol{F}$ are defined corresponding to the noise vectors $(e_1, \ldots, e_\ell)$ and $(f_1, \ldots, f_\ell)$. Consequently, $\bar{\boldsymbol{e}}_j$ and $\bar{\boldsymbol{f}}_j$ can be defined as the j-th columns of $\boldsymbol{E}$ and $\boldsymbol{F}$ (res.). Now we define the next game as follows.

$\boxed{\mathbf{G}_4}$: is similar to the game $\mathbf{G}_3$, except that, $\bar{\boldsymbol{s}}_j = (s_{1j}, \ldots, s_{lj})$, $\bar{\boldsymbol{e}}_j = (e_{1j}, \ldots, e_{lj})$ (note that s_{ij} is the j-th coordinate of polynomial s_i when s_i is seen as a vector) and $\bar{\boldsymbol{f}}_j = (f_{1j}, \ldots, f_{lj})$ for $s_{ij}, e_{ij} \leftarrow D_{\sigma_1}$ and $f_{ij} \leftarrow D_{\sigma_3}$, are respectively replaced with $s_j^* \boldsymbol{\alpha} + \bar{\boldsymbol{s}}'_j$, $e_j^* \boldsymbol{\alpha} + \bar{\boldsymbol{e}}'_j$ and $f_j^* \boldsymbol{\alpha} + \bar{\boldsymbol{f}}'_j$ where $\boldsymbol{\alpha} = \mathbf{x}^1 - \mathbf{x}^0$, such that scalars s_j^*, e_j^*, f_j^* are sampled as $s_j^*, e_j^* \leftarrow D_{\sigma'}$, $f_j^* \leftarrow D_{\sigma''}$ and vectors $\bar{\boldsymbol{s}}'_j, \bar{\boldsymbol{e}}'_j, \bar{\boldsymbol{f}}'_j$ are sampled as

$\bar{\boldsymbol{s}}'_j, \bar{\boldsymbol{e}}'_j \leftarrow D_{\Sigma}$, and $\bar{\boldsymbol{f}}'_j \leftarrow D_{\Sigma'}$ where $\Sigma = \sigma_1^2 I_\ell - \sigma'^2 \boldsymbol{\alpha}^T\boldsymbol{\alpha}$, $\Sigma' = \sigma_3^2 I_\ell - \sigma''^2 \boldsymbol{\alpha}^T\boldsymbol{\alpha}$ and σ', σ'' are positive values.

To show that this game is indistinguishable from its previous game, we apply Lemma 1. Note that since $\|\boldsymbol{\alpha}\|_\infty \leq 2B_x$, if $\sigma_1 > \sqrt{\ell}2B_x\sigma'$ and $\sigma_3 > \sqrt{\ell}2B_x\sigma''$, then matrices Σ and Σ' are positive definite which is the only requirement in Lemma 1. Thus we have,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_4}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_3}(\kappa)| \leq 2n(2\epsilon + \epsilon')$$

where $\epsilon, \epsilon' = 2^{-\kappa}/n$ come from applying Lemma 1 respectively for $\bar{\boldsymbol{s}}_j, \bar{\boldsymbol{e}}_j$ and $\bar{\boldsymbol{f}}_j$ with parameters $\sigma_1, \sigma_3, \sigma', \sigma''$ satisfying $\Gamma_{\Sigma,\sigma'^2\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \geq \eta_\epsilon(\mathbb{Z}^n)$ and $\Gamma_{\Sigma',\sigma''^2\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \geq \eta_\epsilon(\mathbb{Z}^n)$ for $j = 1, \ldots, n$.

Now note that with the mentioned changes in the game $\mathbf{G}_4$, one can rewrite $\boldsymbol{s}_i$ (i.e., i-th row of $\boldsymbol{S}$) as $\boldsymbol{s}_i = \boldsymbol{s}^*\alpha_i + \boldsymbol{s}'_i$ where $\boldsymbol{s}^* = (s_1^*, \ldots, s_n^*)$, $\boldsymbol{s}'_i = (s'_{i1}, \ldots, s'_{in})$ and s'_{ij} is the i-th component of vector $\bar{s}'_j$. Similarly we have, $\boldsymbol{e}_i = \boldsymbol{e}^*\alpha_i + \boldsymbol{e}'_i$ and $\boldsymbol{f}_i = \boldsymbol{f}^*\alpha_i + \boldsymbol{f}'_i$. In the next game, we will use the polynomial representation of the above vectors.

$\boxed{\mathbf{G}_5}$: is the same as game $\mathbf{G}_4$ where in pk_i, ct_i and sk_y, we have replaced s_i, e_i and f_i with their new values from game $\mathbf{G}_4$. Thus,

$$\mathsf{pk}_i = (as^* + e^*)\alpha_i + as'_i + e'_i, \quad \mathsf{sk}_y = \sum_i y_i s'_i$$
$$\mathsf{ct}_i = (as^* + e^*)r + (us^* + f^*)\alpha_i + (as'_i + e'_i)r + us'_i + f'_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}.$$

Since the adversary can query only for $\mathbf{y}$, with $\langle \mathbf{y}, \boldsymbol{\alpha} \rangle = 0$, the key sk_y can be rewritten without the term $\boldsymbol{s}^*$. We have, $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_5}(\kappa) = \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_4}(\kappa)$

$\boxed{\mathbf{G}_6}$: is similar to the game $\mathbf{G}_5$ except that, in pk_i and ct_i values $as^* + e^*$ and $us^* + f^*$ are respectively replaced with uniform polynomials u' and u''. Thus,

$$\mathsf{pk}_i = u'\alpha_i + as'_i + e'_i, \quad \mathsf{sk}_y = \sum_i y_i s'_i$$
$$\mathsf{ct}_i = u'r + u''\alpha_i + (as'_i + e'_i)r + us'_i + f'_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$$

We claim that relying on RLWE assumption $\mathbf{G}_6$ is indistinguishable from $\mathbf{G}_5$. Let $\mathcal{B}$ be the attacker to the RLWE problem with two samples (a, b) and (u, b'), it can simply simulate game $\mathbf{G}_6$ when it has received uniform samples $b = u'$ and $b' = u''$, and it simulates game $\mathbf{G}_5$ when it has received samples with RLWE structures $b = as^* + e^*$ and $b = us^* + f^*$. This is due to the fact that s^*, e^* and f^* have not appeared anywhere else (individually) and the adversary $\mathcal{B}$ can simulate all other required variables by herself simply by sampling from proper distributions. Therefore,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_6}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_5}(\kappa)| \leq \mathsf{Adv}^{\mathsf{RLWE}}_{\mathcal{B}}(\kappa)$$

Note that here f^* and e^* need to be from the same distribution i.e., $\sigma'' = \sigma'$.

Adversary-advantage in Game $\mathbf{G}_6$. Now we show that in game $\mathbf{G}_6$ the advantage of the adversary is zero. This complete the proof. Note that,

$$\begin{aligned} u''\alpha_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}} &= u''(x_i^1 - x_i^0) + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}} \\ &= \lfloor q/K \rfloor (\lfloor q/K \rfloor^{-1} u''(x_i^1 - x_i^0) + x_i^0 1_{\mathsf{R}} + b(x_i^1 - x_i^0) 1_{\mathsf{R}}) \\ &= \lfloor q/K \rfloor ((\lfloor q/K \rfloor^{-1} u'' + b 1_{\mathsf{R}})(x_i^1 - x_i^0) + x_i^0 1_{\mathsf{R}}) \\ &= \lfloor q/K \rfloor (\hat{u}(x_i^1 - x_i^0) + x_i^0 1_{\mathsf{R}}), \end{aligned}$$

where $\lfloor q/K \rfloor^{-1}$ is the inverse of $\lfloor q/K \rfloor$ in $\mathbb{Z}_q$ and $\hat{u}$ is uniformly sampled from R_q. The last equality (which is due to the uniformity of u'') shows that in the game $\mathbf{G}_6$, the values $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_i)_i$ do not depend on the bit b and consequently the advantage of the adversary in this game is 0. □

Remark 1. Note that if one wants to encrypt a matrix $\boldsymbol{X}$ rather than a vector $\mathbf{x}$, a trivial solution is to run the encryption separately for each row of the matrix. This means that the encryption of a matrix with m rows needs $O(mT)$-computations, where $O(T)$ is the computational-complexity of one encryption-run. An interesting property of our scheme is that one can use the provided compactness in the encryption to encrypt a matrix $\boldsymbol{X}$ only by $O(T)$ computational-complexity. For this we just need to define vector 1_{R}^k for $k \in [n]$ as the polynomial of degree $k-1$ in R_q with all the coefficients zero except $(k-1)$th coefficient equals 1. Then ct_i would be as $\mathsf{ct}_i = \mathsf{pk}_i r + f_i + \lfloor q/K \rfloor \sum_{k\in[n]} x_i^k 1_{\mathsf{R}}^k$, where $\mathbf{x}^k = (x_i^k)_i$ is the kth row of $\boldsymbol{X}$ and $\boldsymbol{X}$ has ℓ columns and maximum n rows. The security proof is still working with some small editions: we define $\boldsymbol{\alpha}^k = \mathbf{x}_k^1 - \mathbf{x}_k^0$ associated with kth row of $\boldsymbol{X}$. Then in $\mathbf{G}_4$, we define the new structure of matrices $\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{F}$ w.r.t all the vectors $\boldsymbol{\alpha}^k$. More precisely, jth column of $\boldsymbol{S}$ would be replaced with $\sum_{k\in[n]} s_{j,k}^* \boldsymbol{\alpha}^k + \bar{\mathbf{s}}'_{j,k}$ where $s_{j,k}^*, \bar{\mathbf{s}}'_{j,k}$ are sampled independently for each index k.

4.2 Parameters Setting for selectively-secure IPFE

Here we overview the requirement for the parameters for our selectively-secure IPFE scheme, where κ and n are two separate security parameters (theoretically, one can consider them equal, but we aimed for the efficient implementation).
Correctness. Needs $\ell(2n\kappa\sigma_1\sigma_2 + \sqrt{\kappa}\sigma_3)B_y < \lfloor q/2K \rfloor$ and $q > K > lB_xB_y$.
Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$. Needs $\sigma_3 = \sqrt{2}\sigma_2$, $\Gamma_{\sigma_2 I_n, \sigma_2 I_n} \geq \eta_\epsilon(\mathbb{Z}^n)$ with $\epsilon = 2^{-\kappa}$ (where matrix Γ is defined in Lemma 1) and also all the parameter setting from mhe-RLWE assumption i.e., $\sigma\sqrt{1 - \frac{1}{\sigma_2^2}(\sigma n C\sqrt{\ell+2})^2} > \eta_\epsilon(\mathbb{Z}^{n+n\ell})$ where $\|s_i\|_\infty$, $\|e_i\|_\infty \leq C$ and σ is the parameter for the hardness of RLWE. By Lemma 2, one can set $C = \sqrt{\kappa}\sigma_1$.
Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$. Needs $\sigma_1 > \sqrt{\ell}2B_x\sigma'$ and $\sigma_3 \geq \sqrt{\ell}2B_x\sigma''$ for non-negatives σ' and σ'' where $\sigma_1, \sigma_3, \sigma', \sigma''$ satisfy $\Gamma_{\Sigma_j, \sigma'^2\alpha^T\alpha} \geq \eta_\epsilon(\mathbb{Z}^n)$ and $\Gamma_{\Sigma'_j, \sigma''^2\alpha^T\alpha} \geq \eta_\epsilon(\mathbb{Z}^n)$ with $\epsilon, \epsilon' = 2^{-\kappa}/n$.
Transition from $\mathbf{G}_5$ to $\mathbf{G}_6$. Needs the parameter for the hardness of RLWE where the secret and error are from the distribution $D_{\sigma' I_n}$ and $\sigma' = \sigma''$.
Hardness of RLWE. As we saw we need the parameters q, R, σ and σ' to

satisfy the conditions for the hardness of RLWE. We can use Theorem 6 from Appendix A.3, thus set $\mathsf{R} = \mathbb{Z}[x]/(x^n + 1)$, n is a power of 2, $q = 1 \mod 2n$ and $\sigma = \alpha q(n/\log n)^{1/4}$ and $\sigma' = \alpha' q(2n/\log(2n))^{1/4}$ where $\alpha \leq \sqrt{\log n/n}$, $\alpha' \leq \sqrt{\log n/n}$ and $\sqrt{\alpha q} \geq \omega(\log n)$, $\sqrt{\alpha' q} \geq \omega(\log n)$.

5 Adaptively secure IPFE based on RLWE

Here we modify the construction to lift the security to the adaptive case. The main difference from our selectively-secure construction is that here each secret key s_i and the public parameter a are vectors-of-polynomials rather than two single polynomials. Again the non-negative messages $\mathbf{x}$ and functions $\mathbf{y}$ are bounded by B_x and B_y, respectively, and let K be greater than the maximum value of the inner-product i.e., $K > \ell B_x B_y$. Though, in the security proof we discuss the required parameters, one can also check Appendix C.1 for the parameter-setting.

Construction:

– **Setup:** Let R, R_q be as before. For each $i \in [\ell]$ sample $\boldsymbol{s}_i = (s_{i1}, \ldots, s_{im}) \in \mathsf{R}^m$ where each $s_{ij} \in \mathsf{R}$ is sampled from $D_{\sigma_1 I_n}$. Sample $\boldsymbol{a} = (a_1, \ldots, a_m) \in \mathsf{R}_q^m$ uniformly at random. Check if at least one a_i is invertible in R_q; if not, refuse $\boldsymbol{a}$ and sample it again[9]. Finally, $\mathsf{msk} = \{\boldsymbol{s}_i \mid i \in [\ell]\}$ is the secret-key and the public-key is $\mathsf{mpk} = (\boldsymbol{a}, \{pk_i \mid i \in [\ell]\})$, where $\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle = \sum_j a_j s_{ij}$.

– **Encrypt:** To encrypt a vector $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}^\ell$ with $\|\mathbf{x}\|_\infty \leq B_x$ sample $r \in R_q$ from $D_{\sigma_2 I_n}$ and $\boldsymbol{f}_0 = (f_{01}, \ldots, f_{0m}) \in R_q^m$ from $D_{\sigma_2 I_{nm}}$, and $\{f_i \in \mathsf{R}_q \mid i \in [\ell]\}$ each from $D_{\sigma_3 I_n}$. Then

$$\mathbf{ct}_0 = \boldsymbol{a}r + \boldsymbol{f}_0 = (a_1 r + f_{01}, \ldots, a_m r + f_{0m}), \quad \mathsf{ct}_i = \mathsf{pk}_i r + f_i + \lfloor q/K \rfloor x_i 1_{\mathsf{R}}.$$

Check if at least one element of $\mathbf{ct}_0$ is invertible in R_q and that $\mathbf{ct}_0$ is not a multiple of $\boldsymbol{a}$ (over R_q); if this is not the case, resample $r, \boldsymbol{f}_0$ and recompute $\mathbf{ct}_0, \mathsf{ct}_i$ until the latter holds. The ciphertext is $(\mathbf{ct}_0, \{\mathsf{ct}_i\}_{i \in [\ell]})$.

– **KeyGen:** To generate the decryption key associated with $\mathbf{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}^\ell$ where $\|\mathbf{y}\|_\infty < B_y$, we calculate

$$\mathbf{sk}_y = \sum_{i=1}^{\ell} y_i \boldsymbol{s}_i = (\sum_{i=1}^{\ell} y_i s_{i1}, \ldots, \sum_{i=1}^{\ell} y_i s_{im}) \in \mathsf{R}^m$$

– **Decryption:** To decrypt the ciphertext $(\mathbf{ct}_0, \{\mathsf{ct}_i\}_{i \in [\ell]})$ by the decryption key $\mathbf{sk}_y$, compute: $d = (\sum_{i=1}^{\ell} y_i \mathsf{ct}_i) - \langle \mathbf{ct}_0, \mathbf{sk}_y \rangle$. Then d should be close to $\lfloor q/K \rfloor \langle \mathbf{x}, \mathbf{y} \rangle 1_R$ (a bit perturbed coefficients) and we can extract $\langle \mathbf{x}, \mathbf{y} \rangle$.

Correctness. Similar to the correctness proof in our sel-IPFE, one can verify that we need $\left\| \sum_i \left(y_i f_i - y_i \langle \boldsymbol{f}_0, \boldsymbol{s}_i \rangle \right) \right\|_\infty < \lfloor q/2K \rfloor$ or equivalently, $\ell B_y(\sqrt{\kappa}\sigma_3 + mn\kappa\sigma_1\sigma_2) < \lfloor q/2K \rfloor$.

[9] This step would be done efficiently, since the probability that a_i is invertible, is non-negligible.

We claim that this modified version of our IPFE scheme is adaptively-secure. For the proof we use an extended version of mhe-RLWE assumption associated with polynomially-many samples (rather-than a single sample). We also use Theorem 2 which provides us with the required variant of Ring-LHL.

The first steps of the proof are similar to the security proof of our sel-IPFE, namely, we follow a similar sequence of the games from $\mathbf{G}_0$ to $\mathbf{G}_4$. But in the next games instead of using two samples of RLWE, we use Ring-LHL. The reason for this is that the indistiguishability of proceeding games relies only on statistical arguments and so one can upgrade the security to the adaptive version by a technique similar to the complexity leveraging (CL) even for a large value $(B_x)^\ell$ (while applying CL on the computational arguments needs polynomial-size $(B_x)^\ell$). The overview of games is given in Fig. 3 in Appendix C.

Theorem 4. *Our modified IPFE scheme is adaptively-secure, for proper choice of parameters.*

The formal proof of this theorem and also the parameter settings are given in Appendix C. Similarly as in the selective case, also the adaptively secure scheme can simply be extended to allow encrypting vectors in parallel.

6 Multi-Client IPFE

In this section we present all the needed results to lift our scheme to a multi-client setting. In particular, we present a compiler built upon the compiler of multi-input IPFE (MIFE) scheme of [6], supporting corruptions [3], to transfer a IPFE to its identity-based MCFE version. First, we recall the compiler of [6]. here FE is a single-input IPFE scheme.

Compiler of [6] (MIFE-Compiler): From Single-Input to Multi-Input IPFE.

- $\mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$: it chooses $\boldsymbol{u}_i \xleftarrow{R} \mathbb{Z}_q^k$ and runs $(\mathsf{mpk}'_i, \mathsf{msk}'_i) \leftarrow \mathsf{FE.Setup}(1^\kappa, 1^k)$ for each $i \in [\ell]$. It outputs $\mathsf{msk}_i = (\mathsf{msk}'_i, \boldsymbol{u}_i)$ as the secret key of user i, $\mathsf{msk} = (\mathsf{msk}_i)_i$ as the master key and $\mathsf{pp} = (\mathsf{mpk}'_i)_i$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$: for $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ where $\mathbf{y}_i \in \mathbb{Z}_q^k$, it runs $\mathsf{sk}_{i,y} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}'_i, \mathbf{y}_i)$ for $i \in [\ell]$ and sets $\mathsf{sk}'_y = \sum_i \boldsymbol{u}_i \mathbf{y}_i$. Then it outputs $\mathsf{sk}_y = ((\mathsf{sk}_{i,y})_i, \mathsf{sk}'_y)$.
- $\mathsf{Enc}(\mathsf{msk}_i, \mathbf{x}_i)$: for $\mathbf{x}_i \in \mathbb{Z}_q^k$, it runs $\mathsf{ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{msk}'_i, \mathbf{x}_i + \boldsymbol{u}_i)$ and outputs ct_i.
- $\mathsf{Dec}((\mathsf{ct}_i)_i, \mathsf{sk}_y)$: it runs $D_i \leftarrow \mathsf{FE.Dec}_1(\mathsf{ct}_i, \mathsf{sk}_{i,y})$ for $i \in [\ell]$. Then it outputs $\mathsf{FE.Dec}_2(\sum_i D_i + \mathcal{E}(-\mathsf{sk}'_y, 0))$.

The compiler can be used on any IPFE scheme with the following properties:

Property 1 (2-step decryption with a linear encoding). The decryption algorithm of IPFE is a 2-step decryption (i.e., $\mathsf{Dec}(\mathsf{ct}, \mathsf{sk}) = \mathsf{Dec}_2(\mathsf{Dec}_1(\mathsf{ct}, \mathsf{sk}))$, where $\mathsf{Dec}_1(\mathsf{ct}, \mathsf{sk}) = \mathcal{E}(\langle x, y\rangle, \text{noise})$). That is, the first step outputs an encoding of inner-product and in the second step it extracts the inner-product from the mentioned encoding. Additionally, the encoding also has a linear property.[10]

Property 2 (linear encryption). Let Enc be the encryption algorithm of IPFE scheme. Then there exists a deterministic algorithm Add, such that the two following distributions of $\mathsf{Enc}(\mathsf{msk}, \mathbf{x}_1 + \mathbf{x}_2)$ and $\mathsf{Add}(\mathsf{Enc}(\mathsf{msk}, \mathbf{x}_1), \mathbf{x}_2)$ are identical. Informally, given the message $\mathbf{x}_2$ and the encryption of $\mathbf{x}_1$, one can compute the encryption of $\mathbf{x}_1 + \mathbf{x}_2$:

In our RLWE-based IPFE scheme, Dec_1 outputs the inner-product added by a noise term, then Dec_2 removes the noise. Encoding is defined as adding the noise which is linear. It is easy to see that the encryption is linear. We formalize this in Appendix D.2 .

We now present our compiler to build an identity-based MCFE (from MIFE allowing corruptions). In the following construction $H : (U, \mathsf{Labels}) \to \mathbb{Z}_q^k$ is a hash function (later modeled as a random oracle).

Our Compiler (MCFE-Compiler): From Multi-Input to identity-based Multi-Client IPFE.

- $\mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$: it chooses $\boldsymbol{u}'_i \xleftarrow{R} U$ and runs $(\mathsf{mpk}'_i, \mathsf{msk}'_i)_{i \in [\ell]} \leftarrow \mathsf{MIFE.Setup}(1^\kappa, 1^\ell, 1^k)$. It outputs $\mathsf{msk}_i = (\mathsf{msk}'_i, \boldsymbol{u}'_i)$ as the secret key of user i, $\mathsf{msk} = (\mathsf{msk}_i)_i$ as the master key and $\mathsf{pp} = (\mathsf{mpk}'_i)_i$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}, \gamma)$: it runs $\mathsf{sk}_y = ((\mathsf{sk}_{i,y})_i, \mathsf{sk}'_y) \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{msk}, \mathbf{y})$ and sets $\mathsf{sk}''_{y,\gamma} = \mathsf{sk}'_y + \sum_i H(\boldsymbol{u}'_i, \gamma)\mathbf{y}_i$. Then it outputs $\mathsf{sk}_{y,\gamma} = ((\mathsf{sk}_{i,y})_i, \mathsf{sk}''_{y,\gamma})$.
- $\mathsf{Enc}(\mathsf{msk}_i, \mathbf{x}_i, \gamma)$: it runs $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}_i + H(\boldsymbol{u}'_i, \gamma))$ and outputs $\mathsf{ct}_{i,\gamma}$.
- $\mathsf{Dec}((\mathsf{ct}_{i,\gamma})_i, \mathsf{sk}_{y,\gamma})$: it runs $D_\gamma \leftarrow \mathsf{MIFE.Dec}((\mathsf{ct}_{i,\gamma})_i, (\{\mathsf{sk}_{i,y}\}_i, \mathsf{sk}''_{y,\gamma}))$ and outputs D_γ

In the security proof of the above compiler, we use Property 2, used also for the compiler of [6].

Theorem 5. *In the above compiler (from MIFE with corruptions to identity-based MCFE), if MIFE is secure, then our construction is a secure MCFE against static corruptions.*[11]

The formal proof is given in Appendix D.2, here we sketch the approach. The proof proceeds through a sequence of games defined w.r.t to the labels issued by the adversary. For a fixed label γ we change the messages $\mathbf{x}^0_{i,\gamma}$ encrypted under the label γ to $\mathbf{x}^1_{i,\gamma}$ for all i. To ensure that such changes are indistinguishable, we

[10] For the sake of simplicity, here we gave an informal description of this property. An interested reader can see [6] for the formal one. The formal description guarantees the correctness of the MIFE scheme w.r.t the general IPFE, and is not used in the proof of security.

[11] Note that we are specifically using MIFE scheme of [6] and it is not any possible MIFE scheme in RO model.

rely on the security of MIFE. For encryption queries w.r.t γ the simulator answers by relaying them to the MIFE-challenger, and it programs the random oracle queries as $H(\boldsymbol{u}_i', \gamma') = \boldsymbol{r}_{i,\gamma'} - \boldsymbol{u}_i$, for $\gamma' \neq \gamma$, while $\boldsymbol{r}_{i,\gamma'}$ is randomly chosen. This allows to remove the term $\boldsymbol{u}_i$ from the encryption, which is the only unknown part to the simulator, and simulate the queries correctly.

Finally, we argue that our RLWE based scheme can be used in the above compilers.

Proposition 1. *The MIFE-compiler and the MCFE-compiler applied on our IPFE schemes in Section 4 or Section 5 result in a secure and correct MCFE scheme.*

In Appendix D.4, we further extend our identity-based MCFE scheme to its decentralized version, where we use the compiler of [3], but we modify the compiler and the security proof for the case that the secret key is involved with the label as well (which is the case in our scheme). We also show that our IPFE scheme has the required properties to be used in this compiler.

Batching in (D)MCFE: As stated in Remark 1, our RLWE scheme supports encrypting multiple messages in parallel. This property is preserved with (D)MCFE compilers described in this section. To be precise, each encrypted row needs to be masked (as described above) separately. Furthermore, identity-based (D)MCFE allows us to derive functional keys depending on a chosen label. If one encrypts multiple rows in parallel with different labels, a functional key will decrypt only the ones with the matching label. This allows fine-grained control on a batch of messages.

7 Practical instantiation

In this section, we demonstrate the efficiency and practicality of our scheme with concrete instantiations. We provide different parameter sets with different levels of security and strategies for a very efficient implementation. Finally, we apply our scheme for a privacy preserving machine learning application of identifying digits from encrypted images. The implementation is publicly available at `https://github.com/fentec-project/IPFE-RLWE`.

7.1 Implementation

Similar to other RLWE based schemes, the two major components of our scheme are polynomial multiplication and noise sampling. However, from the computational point of view the most challenging task here is to efficiently implement multiple polynomial multiplications and multiple sampling of secret and error polynomials which grow linearly with ℓ. Here, we describe our approach for efficient implementation of these components, all running in constant-time.

Discrete Gaussian sampling: Our scheme uses discrete Gaussian distribution to sample error and secret vectors. A non-constant-time sampler leaks sensitive information about these secret vectors that can break the cryptosystem. There are three choices for constant time sampling i) linear-searching of CDT (Cumulative Distribution Table) table [12], ii) bit-sliced sampler [23], and iii) constant-time binary sampling [41]. The first two methods are very efficient for smaller (< 10) standard deviations but do not scale very well for larger standard deviations. Moreover, they need different tables or minimized Boolean expressions for different samplers. One can use convolutions to first sample from smaller distributions and then combine them to generate a sample from a distribution with larger standard deviation [34]. However, this method is less efficient compared to the constant-time binary sampling described by Zhao et al. [41]. In this method, to generate a sample from D_σ, first a sample from a base distribution $x \xleftarrow{R} D^+_{\sigma_0}$ is generated. Next, an integer k is fixed such that $\sigma = k\sigma_0$ and a integer y is sampled uniformly from $[0, \cdots .k-1]$. Finally, a rejection sampling on $z = kx + y$ with the acceptance probability $p = \exp(\dfrac{-y(y+2kx)}{2\sigma^2})$ is performed. It can be easily shown that the samples generated in this way are *statistically close* to discrete Gaussian distribution with Gaussian parameter σ. To generate a sample from D_σ a randomly generated sign bit is applied on z. The rejection sampling is performed using a Bernoulli sampler. If the base sampling algorithm $D^+_{\sigma_0}$ and the Bernoulli sampler are constant-time this method runs in constant-time. In our implementation to generate samples from $\sigma_1 = k_1\sigma_0$, $\sigma_2 = k_2\sigma_0$, and $\sigma_3 = k_3\sigma_0$, we use the constant-time Bernoulli sampler proposed by Zhao et al. [41] for different values of k and σ. The uniform sampler has also been updated for different values of k. Finally, a linear-search based CDT sampling algorithm has been used for the constant-time base sampler. Using the bit-sliced algorithm to instantiate the base sampler might improve the efficiency to some extent but we leave this as future work.

CRT representation: Due to the correctness and security constraints of our scheme, the modulus q required in all variants of our scheme is quite large (≥ 64 bits). Similar to homomorphic encryption implementations [37] we adapted the residual number system based polynomial arithmetic using Chinese remainder theorem to avoid the naive and relatively slow multi-precision arithmetic. We choose a chain of moduli $q_0, q_1, \ldots, q_{n_p-1}$ such that $q = q_0 \cdot q_1 \cdots q_{n_p-1}$. All the inputs, outputs, and intermediate values are stored as elements in rings R_{q_i} instead of R_q. As all the q_i are less than 32 bits long this replaces the expensive multi-precision polynomial arithmetic with simple and efficient single-precision arithmetic. We only need to revert to R_q while extracting the value d at the end of decryption operation. We use Garner's algorithm shown in Alg. 1 in Appendix E and GNU multi-precision library to accomplish this.

Polynomial arithmetic: We use Number theoretic transform (NTT) based polynomial multiplication in our scheme since it is an in-place algorithm and runs in $O(n \log n)$ time complexity where n is the length of the polynomial. Specifically, we used the NTT with *negative wrapped convolution* [27] which produces the result of the multiplication reduced by $1 + x^n$ without any extra memory.

For a power-of-two n and prime modulus q_i, such that $q_i \equiv 1 \mod 2n$, the multiplication of two polynomials $a, b \in \mathsf{R}_{q_i}$ can be calculated as $NTT^{-1}(NTT(a) \circ NTT(b))$ where NTT and NTT^{-1} are forward and inverse NTT transformations respectively and $\circ$ denotes the component-wise multiplication of two vectors. Computationally, the forward and the inverse NTT transformation are the prevalent components of the whole $O(n \log n)$ time multiplication. We observe that one of the multiplicands, i.e. a in **Setup** and r in **Encrypt** stays same for all the $\ell + 1$ multiplications, Hence we precompute and store NTT(a) and NTT(r). This saves ℓ NTT transformations in each case. Also, the public polynomial a is random in R_{q_i}. As NTT transformation of a random vector is also random, we can assume the a is already in the NTT domain.

NTT or NTT^{-1} transformation algorithms require applying bit-reversal permutations before or after each transformation. As our polynomials are quite large and the number of multiplications is linear in ℓ, this requirement induces a significant overhead. To overcome this problem we followed the same strategy as Pöppelman et al. [35]. We used the *decimation-in-time* NTT based on Cooley-Tukey [16] butterfly as shown in Alg. 2 which requires input in normal ordering but produces output in bit-reversed ordering. For the inverse transformation we switch to *decimation-in-frequency* NTT based on Gentleman-Sande [19] butterfly as shown in Alg. 3 in Appendix E, which accepts the input in bit-reversed ordering and produces the output in normal ordering. Hence, applying these transformations in conjunction eliminates the need for bit-reversal step.

Other: There are two common strategies to generate pseudo-random numbers in cryptographic implementations: using extended output function like Keccak [10] or using block ciphers in counter mode. Since our target platform is equipped with AES-NI (Advanced Encryption Standard New Instructions), we decided to use AES in CTR mode for fast generation of cryptographically secure pseudo-random numbers. Further, we have chosen our NTT friendly primes $q_i, i \in [0, n_p - 1]$ of the form $2^i - 2^j + 1$. Due to their special structure it is possible to perform fast modular reduction similar to Mersenne primes with these primes.

7.2 Parameters and performance

We propose three sets of parameters in Table. 1 depending with different values of ℓ, B_x, and B_y. Here we have considered the selectively secure scheme described in Section 4.2. We calculate the concrete security of our scheme based on the underlying hardness of a RLWE instance. That is, we deduce our functional encryption with parameters $(n, q, \sigma_1, \sigma_2, \sigma_3, \ell, B_x, B_y)$ scheme offers $\mathcal{S}$ bits of security if the the underlying RLWE instance with (n, q, σ) offers $\mathcal{S}$ bits of security. Here, the parameters $(n, q, \sigma_1, \sigma_2, \sigma_3, \ell, B_x, B_y)$ and (n, q, σ) are related to satisfy the security constraints delineated in Section 4.2.

Performance: Table. 1 also lists the performance of different operations of our scheme. We benchmarked on a single core of an Intel i9-9880H processor running at maximum 4.8GHz frequency. The code has been compiled using

GCC-9.3 with optimization flags `-O3 -fomit-frame-pointer -march=native` on Ubuntu 18.04.

Security level	PQ Security	FE Bounds	Gaussian Parameters	Ring Parameters	CRT moduli	Time (ms)
		$B_x : 2$	$\sigma_1 : 33$	$n : 2048$	$q_1 : 2^{14} - 2^{12} + 1$	`Setup`:26
Low	76.3	$B_y : 2$	$\sigma_2 : 59473921$		$q_2 : 2^{23} - 2^{17} + 1$	`Enc`:16 `KG`:0.27
		$\ell : 64$	$\sigma_3 : 118947840$	$\lceil \log q \rceil : 66$	$q_3 : 2^{29} - 2^{18} + 1$	`Dec`:1
		$B_x : 4$	$\sigma_1 : 225.14$	$n : 4096$	$q_1 : 2^{24} - 2^{14} + 1$	`Setup`:589
Medium	119.2	$B_y : 16$	$\sigma_2 : 258376412.19$		$q_2 : 2^{31} - 2^{17} + 1$	`Enc`:381 `KG`:22
		$\ell : 785$	$\sigma_3 : 516752822.39$	$\lceil \log q \rceil : 86$	$q_3 : 2^{31} - 2^{24} + 1$	`Dec`:17
		$B_x : 32$	$\sigma_1 : 2049$	$n : 8192$	$q_1 : 2^{17} - 2^{14} + 1$	`Setup`:1743
High	246.2	$B_y : 32$	$\sigma_2 : 5371330561$		$q_2 : 2^{20} - 2^{14} + 1$	`Enc`:1388
					$q_3 : 2^{32} - 2^{20} + 1$	`KG`:70
		$\ell : 1024$	$\sigma_3 : 10742661120$	$\lceil \log q \rceil : 101$	$q_4 : 2^{32} - 2^{30} + 1$	`Dec`:45

Table 1: Parameters and performance of the RLWE based FE scheme. The security has been calculated using the LWE estimator tool [9].

7.3 Machine learning on encrypted data and other use cases

To demonstrate the efficiency of our scheme, we use it in a real world application of FE. We perform a task of classification with a simple machine learning model, but on encrypted data using our IPFE. In particular, we evaluate logistic regression on MNIST dataset, recognizing handwritten digits in images. This task involves computing 10 linear functions on a 785-dimensional vectors, where the complexity of computation is bounded with $B_x = 4$ and $B_y = 16$. See Appendix E.1 for more in depth description of the MNIST dataset.

Parameters in Table. 1 for medium level of security (129 bit of PQ Security) were chosen to fit this use-case. Hence it takes approx. 381ms to encrypt an image (vector representation) of this size and only 170ms to evaluate the model, i.e. we need to perform 10 decryptions to properly classify an image. In fact, as explained in Remark 1, one can encrypt with one encryption-run multiple images simultaneously, in our case up to 4096 images. Evaluating the model would classify all the images at once, without a major change in the complexity.

Other: We would like to additionally highlight possible practical scenarios where our scheme excels over other known schemes. On one hand, single-input public key RLWE based IPFE is particularly useful when multiple data from the same source is processed with FE, due to its batching property. This could be, for example, streams of data (e.g. a video, see [1] where a single-input public key scheme was used) processed in some fixed intervals, or learning a ML model [40] where IPFE can be used on an encrypted dataset, usually evaluating the same function on batches. On the other hand, the data itself might be structured as a matrix. In [30], a DMCFE scheme was proposed for a privacy preserving location tracking. Users in a decentralized way, for each possible location, encrypt 0 or 1 indicating their presence. Using IPFE, averages (heatmaps) can be computed, where RLWE batching can be used to cover, say, 4096 locations with one ciphertext, outperforming known FE schemes.

8 Acknowledgements

The authors were supported by EU's Horizon 2020 FENTEC project. Angshuman Karmakar is funded by FWO (Research Foundation – Flanders) as junior post-doctoral fellow (contract number 203056 / 1241722N LV). Tilen Marc is supported by ARRS projects P1-0297, N1-0095, J1-1693, N1-0218 and Horizon 2020 KRAKEN project. Azam Soleimanian benefited from the support of the Chair « Blockchain & B2B Platforms », led by l'X – Ecole polytechnique and the Fondation de l'Ecole polytechnique, sponsored by Capgemini, Nomadics Lab and Caisse des dépôts.

References

1. https://fentec.eu/content/motion-detection-and-local-decision-making
2. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_19
3. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_5
4. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_33
5. Abdalla, M., Bourse, F., Marival, H., Pointcheval, D., Soleimanian, A., Waldner, H.: Multi-client inner-product functional encryption in the random-oracle model. Cryptology ePrint Archive, Report 2020/788 (2020), https://eprint.iacr.org/2020/788
6. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_20

7. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_12
8. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC. pp. 99–108. ACM Press (May 1996). https://doi.org/10.1145/237814.237838
9. Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the {LWE, NTRU} schemes! In: Security and Cryptography for Networks - 11th International Conference, SCN 2018. vol. 11035, pp. 351–367. Springer (2018). https://doi.org/10.1007/978-3-319-98113-0_19, https://doi.org/10.1007/978-3-319-98113-0_19
10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_19
11. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16
12. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy. pp. 553–570. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.40
13. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_29
14. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_24
15. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018), https://eprint.iacr.org/2018/1021
16. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Mathematics of Computation **19**(90), 297–301 (1965), http://www.jstor.org/stable/2003354
17. Dufour-Sans, E., Gay, R., Pointcheval, D.: Reading in the dark: Classifying encrypted digits with functional encryption. IACR Cryptol. ePrint Arch. **2018**, 206 (2018), http://eprint.iacr.org/2018/206
18. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.13
19. Gentleman, W.M., Sande, G.: Fast fourier transforms: for fun and profit. AFIPS Conference Proceedings, vol. 29, pp. 563–578. AFIPS / ACM / Spartan Books, Washington D.C. (1966). https://doi.org/10.1145/1464291.1464352, https://doi.org/10.1145/1464291.1464352
20. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407

21. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488678
22. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_11
23. Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.: Constant-time discrete gaussian sampling. IEEE Trans. Computers **67**(11), 1561–1571 (2018). https://doi.org/10.1109/TC.2018.2814587, https://doi.org/10.1109/TC.2018.2814587
24. Katsumata, S., Yamada, S.: Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 682–712. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_23
25. Liu, F.H., Wang, Z.: Rounding in the rings. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 296–326. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_11
26. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
27. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (Feb 2008). https://doi.org/10.1007/978-3-540-71039-4_4
28. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1
29. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_3
30. Marc, T., Stopar, M., Hartman, J., Bizjak, M., Modic, J.: Privacy-enhanced machine learning with functional encryption. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019, Part I. LNCS, vol. 11735, pp. 3–21. Springer, Heidelberg (Sep 2019). https://doi.org/10.1007/978-3-030-29959-0_1
31. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010), https://eprint.iacr.org/2010/556
32. Peikert, C.: Limits on the hardness of lattice problems in ell _p norms. In: 22nd Annual IEEE Conference on Computational Complexity (CCC 2007). pp. 333–346. IEEE Computer Society (2007). https://doi.org/10.1109/CCC.2007.12, https://doi.org/10.1109/CCC.2007.12
33. Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939 (2015), https://eprint.iacr.org/2015/939
34. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 353–370. Springer, Heidelberg (Sep 2014). https://doi.org/10.1007/978-3-662-44709-3_20
35. Pöppelmann, T., Oder, T., Güneysu, T.: High-performance ideal lattice-based cryptography on 8-bit atxmega microcontrollers. In: Lauter, K.E., Rodríguez-Henríquez, F. (eds.) Progress in Cryptology - LATINCRYPT 2015. vol. 9230, pp.

346–365. Springer (2015). https://doi.org/10.1007/978-3-319-22174-8_19, https://doi.org/10.1007/978-3-319-22174-8_19
36. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
37. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL (Oct 2019), microsoft Research, Redmond, WA.
38. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_4
39. Wang, Z., Fan, X., Liu, F.H.: FE for inner products and its application to decentralized ABE. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 97–127. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_4
40. Xu, R., Joshi, J.B., Li, C.: Cryptonn: Training neural networks over encrypted data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). pp. 1199–1209 (2019). https://doi.org/10.1109/ICDCS.2019.00121
41. Zhao, R.K., Steinfeld, R., Sakzad, A.: FACCT: fast, compact, and constant-time discrete gaussian sampler over integers. IEEE Trans. Computers **69**(1), 126–137 (2020). https://doi.org/10.1109/TC.2019.2940949, https://doi.org/10.1109/TC.2019.2940949

Supplementary Materials

A Background

Here we present some additional materials.

A.1 Lattices

A lattice is a discrete subset of $\mathbb{R}^n$ which can be generated by a integer linear combination of some vectors known as the basis. It is formally defined as follows.

Definition 6 ([33]).
– **Lattice.** *A lattice $\mathcal{L}$ is a subset of $\mathbb{R}^n$ that it is both:*

1. *An additive subgroup: $\mathbf{0} \in \mathcal{L}$ and $-\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{y} \in \mathcal{L}$, for every $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$.*
2. *Discrete: every $\boldsymbol{x} \in \mathcal{L}$ has a neighborhood in $\mathbb{R}^n$ which $\boldsymbol{x}$ is the the only lattice point.*

– **Basis and dimension.** *Equivalently, a k-dimensional lattice $\mathcal{L}$ can be defined by its basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k)$ for $\boldsymbol{b}_i \in \mathbb{R}^n$ as:*

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}) = \left\{ \sum_{i=1}^{k} c_i \boldsymbol{b}_i \ : \ c_i \in \mathbb{Z} \right\}$$

– **Minimum distance.** *The minimum distance of a lattice $\mathcal{L}$ is the length of a shortest nonzero lattice vector: $\lambda_1 := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} ||\boldsymbol{v}||$. If we use the infinity norm* *in the expression, we denote it with* λ_1^∞.

– **Dual-lattice.** *The dual of a lattice* $\mathcal{L} \subset \mathbb{R}^n$ *is defined as:* $\widehat{\mathcal{L}} := \{\boldsymbol{w} \ : \ \langle \boldsymbol{w}, \mathcal{L} \rangle \subset \mathbb{Z}\}$ *i.e., the set of points whose inner products with the vectors in* $\mathcal{L}$ *are all integers. It is straightforward to verify that* $\widehat{\mathcal{L}}$ *is a lattice.*

A.2 Discrete Gaussian Distribution

In this section we gather the results needed to prove the properties of discrete Gaussian distribution used in the paper

We have the following useful fact showing that values from a discrete Gaussian distribution can be bounded.

Lemma 2 ([26]). *For any* $k > 0$, $\Pr_{x \leftarrow D_\sigma}[|x| > \sqrt{k}\sigma] \leq 2e^{-k/2}$. *(one dimension Gaussian)*

The following lemma explains that sampling from lattice $\mathbb{Z}^n$ is efficiently doable, which is the core of our construction.

Lemma 3 ([20]). *Let* Σ *be positive definite. There exists a polynomial-time algorithm for sampling from a distribution whose statistical distance to* $D_{\mathbb{Z}^n, \sqrt{\Sigma}}$ *is negligible, as long as* $\sqrt{\Sigma} \geq \omega(\log(n))$.

Note that in [20, Theorem 4.1] the statement is a bit different saying that for arbitrary lattice L with basis $\boldsymbol{B}$ and diagonal positive definite matrix $\sigma^2 I_n$ there exists a polynomial-time algorithm for sampling from $D_{L,\sigma}$, as long as $\sigma \geq \omega(\log(n))||\boldsymbol{B}||$, where $||B|| = \max_i(||\boldsymbol{b_i}||)$. The statements of the above lemma follows directly from the following reasons. Assume that $\sqrt{\Sigma} > \omega(\log(n))$ and define the basis matrix $\boldsymbol{B} = \sigma \sqrt{\Sigma}^{-1}$. Then $\omega(\log(n))||\boldsymbol{B}|| < \sigma$, and by the original statement we can sample from $D_{L,\sigma}$, where L is defined by the basis $\boldsymbol{B}$. Now let z be sampled from $D_{L,\sigma}$:

$$\Pr[z = x_1\boldsymbol{b_1} + \ldots + x_n\boldsymbol{b_n}] \propto \exp(-\frac{||x_1\boldsymbol{b_1} + \ldots + x_n\boldsymbol{b_n}||^2}{2\sigma^2}).$$

On the other hand for w sampled from $D_{\mathbb{Z}^n, \sqrt{\Sigma}}$ we have

$$\begin{aligned}\Pr[w = (x_1, \ldots, x_n)] &\propto \exp(-\frac{(x_1, \ldots, x_n)^T \Sigma^{-1}(x_1, \ldots, x_n)}{2}) \\ &= \exp(-\frac{||x_1\boldsymbol{b_1} + \ldots + x_n\boldsymbol{b_n}||^2}{2\sigma^2}),\end{aligned}$$

where the last equality follows from the definition of $\boldsymbol{B}$. Hence samples from $D_{\mathbb{Z}^n, \sqrt{\Sigma}}$ can be extracted as coefficients of samples from $D_{L,\sigma}$.

The following is in the proof of Lemma 1. For any lattice $\mathcal{L}$ and positive real $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(\mathcal{L})$ is the smallest real $s > 0$ such that $\rho_{s^{-1}I}(\widehat{\mathcal{L}} \setminus \{0\}) \leq \epsilon$.

Lemma 4 ([4,20]). *Let Σ be a positive semi-definite matrix. For every $\boldsymbol{c} \in \mathbb{R}^n$ in the span of Σ it holds*

$$\rho_{\sqrt{\Sigma}}(\boldsymbol{c} + \mathbb{Z}^n) = \rho_{\sqrt{\Sigma}}(\mathbb{Z}^n)\mu_c,$$

for some $\mu_c \in [\frac{1-\epsilon}{1+\epsilon}, 1]$, as long as $\sqrt{\Sigma} \geq \eta_\epsilon(\mathbb{Z}^n)$.

A.3 Hardness of RLWE

The following theorem discusses hardness of RLWE and required parameters for the reduction from SIVP to RLWE.

Theorem 6 ([28] Theorem 3.6). *Let $\mathsf{R} = \mathbb{Z}[x]/(x^n + 1)$ where n is a power of 2, $\alpha = \alpha(n) \leq \sqrt{\log n/n}$, and $q = 1 \mod 2n$ which is a $\operatorname{poly}(n)$-bounded prime such that $\sqrt{\alpha q} \geq \omega(\log n)$. Then there exists a $\operatorname{poly}(n)$-time quantum reduction from $\tilde{O}(\sqrt{n/\alpha})$-approximate SIVP (Short Independent Vectors Problem) on ideal lattices[12] in the ring R to solving $\mathsf{RLWE}_{q,\chi}$ with $l \geq 1$ samples. where $\chi = D_{\mathbb{Z}^n,\sigma}$ is the discrete Gaussian distribution with parameter $\sigma = \alpha q \cdot (nl/\log(nl))^{1/4}$.*

The following lemma is an immediate extension of our mhe-RLWE assumption from one sample to m samples. The proof works in a similar way, except that instead of discussing based on a single sample, matrices Δ', A' and B' we would have m versions of these matrices.

Lemma 5 (mhe-RLWE with (polynomially) many samples). *One can further extend mhe-RLWE to include m samples $(a_j r + f_j)_{j\in[m]}$. More precisely, two following distributions are indistinguishable.*

$$\Big((a_j)_{j\in[m]}, (a_j r + f_j)_{j\in[m]}, \big(e_i, (s_{i,j})_{j\in[m]}, e_i r + g_i, (s_{i,j} f_j + h_{i,j})_{j\in[m]}\big)_{i\in[l]}\Big)$$

and

$$\Big((a_j)_{j\in[m]}, (u_j)_{j\in[m]}, \big(e_i, (s_{i,j})_{j\in[m]}, e_i r + g_i, (s_{i,j} f_j + h_{i,j})_{j\in[m]}\big)_{i\in[l]}\Big).$$

where $a_j, u_j \in R_q$ are sampled uniformly, $||e_i||_\infty, ||s_{i,j}||_\infty \leq C$, and r, f_j, g_i, $h_{i,j}$ sampled from $D_{\delta I_n}$, for $i \in [l], j \in [m]$, as long as $\sigma\sqrt{1 - \frac{1}{\delta^2}(\sigma n C\sqrt{l+2})^2} \geq \eta_\epsilon(\mathbb{Z}^{n+nl})$, where σ is such that RLWE is hard with m given samples.

B Leftover Hash Lemma in rings

In this section we discuss our Leftover Hash Lemma presented in Section 3.2. We start with some additional notation.

[12] Here the aim is just to show that RLWE is hard. So, we avoid to recall the definition of ideal lattices. The interested reader can see [28] for the definition of ideal lattices.

For a matrix $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$, we shall write $\boldsymbol{a}_i \in \mathsf{R}_q^m$ for the i-th row of $\boldsymbol{A}$ and $a_{i,j} \in \mathsf{R}_q$ for the entry in i-th row and j-th column. We denote with $(\mathsf{R}_q^{k\times m})^*$ the set of all matrices $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ for which the mapping $f_A : \mathsf{R}_q^m \to \mathsf{R}_q^k$ defined as matrix multiplication $f_A(x) = \boldsymbol{A}\boldsymbol{x}$ is surjective.
Let $\boldsymbol{a}_i = (a_{i,1}, \ldots, a_{i,m}) \in \mathsf{R}_q^m$. Then denote:

$$\boldsymbol{a}_i^{\perp} = \{(t_1, \ldots, t_m) \in \mathsf{R}^m \mid \sum_{i=j}^{m} a_{i,j} t_j = 0 \bmod q\}$$
$$L(\boldsymbol{a}_i) = \{(t_1, \ldots, t_m) \in \mathsf{R}^m \mid \exists s \in \mathsf{R}_q, \forall j \in [m] : t_j = a_{i.j} s \bmod q\}$$

For a matrix $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ we shall write:

$$\begin{aligned} L(\boldsymbol{A}) &:= L(\boldsymbol{a}_1) + \ldots + L(\boldsymbol{a}_k) \\ &= \{(t_1, \ldots, t_m) \in \mathsf{R}^m \mid \exists (s_1, \ldots, s_k) \in \mathsf{R}_q^k, \forall j \in [m] : t_j = \sum_{i=1}^{k} a_{i,j} s_i \bmod q\} \end{aligned}$$

For $\boldsymbol{a}_i \in \mathsf{R}_q^m$ define $\boldsymbol{a}_i^x = (a_{i,1}^x, \ldots, a_{i,m}^x)$, where $a_{i,j}^x := a_{i,j}(x^{-1})$. Then $\boldsymbol{A}^x$ is defined to have i-th row $\boldsymbol{a}_i^x$.

B.1 Proof of Theorem 2

For clarity, we first prove our main theorem, while all the required lemmas are given latter. All the notation is defined above and in Appendix A.1.

Proof. Let Δ_A denote the distance to the uniformity of $\boldsymbol{A}\boldsymbol{t}$ for fixed $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})^*$. The mapping $\boldsymbol{t} \mapsto \boldsymbol{A}\boldsymbol{t}$ from $\mathbb{Z}^{nm}$ to R_q^k is surjective by the definition of $(\mathsf{R}_q^{k\times m})^*$. Its kernel is $\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}$, where $\boldsymbol{a}_i$ are rows of $\boldsymbol{A}$. Hence the mapping induces an isomorphism between $\mathbb{Z}^{nm}/(\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp})$ and R_q^k.

This implies that the statistical distance Δ_A is equal to the uniformity of $\boldsymbol{t} \bmod \boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}$. By Lemma 6, it holds $\Delta_A \le 2\delta$, if σ is greater than the smoothing parameter $\eta_\delta(\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp})$. Then Lemma 7 bounds $\eta_\delta(\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}) \le \sqrt{\ln(2mn(1+1/\delta)/\pi} / \lambda_1^{\infty}(\widehat{\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}})$.

To bound $\lambda_1^{\infty}(\widehat{\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}})$ we note that $\widehat{L_1 \cap L_2} = \widehat{L_1} + \widehat{L_2}$ where plus denotes linear combinations of the lattices. By Lemma 8, $\widehat{\boldsymbol{a}_i^{\perp}} = \frac{1}{q} L(\boldsymbol{a}_i^x)$, thus $\widehat{\boldsymbol{a}_1^{\perp} \cap \ldots \cap \boldsymbol{a}_k^{\perp}} = \frac{1}{q} L(\boldsymbol{A}^x)$.

Lemma 9 bounds $\lambda_1^{\infty}(L(\boldsymbol{A}^x)) \ge \frac{1}{\sqrt{n}} q^{1-\frac{k}{m}-\frac{\epsilon}{k}}$, thus $\lambda_1^{\infty}(\frac{1}{q} L(\boldsymbol{A}^x)) \ge \frac{1}{\sqrt{n}} q^{-\frac{k}{m}-\frac{\epsilon}{k}}$, except with probability $2^n \frac{1}{q^{\epsilon n}}$ over the choice of $\boldsymbol{A}^x \in \mathsf{R}_q^{k\times m}$. Since $\boldsymbol{A} \mapsto \boldsymbol{A}^x$ is a bijection, the latter holds over the choice of $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})$. Therefore, we have $\lambda_1^{\infty}(\frac{1}{q} L(\boldsymbol{A}^x)) \ge \frac{1}{\sqrt{n}} q^{-\frac{k}{m}-\frac{\epsilon}{k}}$, except with probability at most $2^n \frac{1}{q^{\epsilon n}} (\frac{|\mathsf{R}_q^{k\times m}|}{|(\mathsf{R}_q^{k\times m})^*|})$ over the choice of $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})^*$.

On the other hand, it holds $\frac{|\mathsf{R}_q^{k\times m}|}{|(\mathsf{R}_q^{k\times m})^*|} = (\frac{q^{mk}}{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})})^n$, which is obtained using the fact that $\mathsf{R}_q \cong \mathbb{Z}_q^n$. In fact, $\mathsf{R}_q \cong \mathbb{Z}_q^n$ implies that counting all possible $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})^*$ is equivalent to counting the number of n-tuples of matrices $\boldsymbol{A}_i \in \mathbb{Z}_q^{k\times m}$, each with linearly independent rows over $\mathbb{Z}_q$. Thus the equation follows.

Summing up, if $\sigma \geq \sqrt{n\ln(2mn(1+1/\delta))/\pi}\cdot q^{\frac{k}{m}+\frac{\epsilon}{k}}$ then $\Delta_A \leq 2\delta$, except for a fraction of $2^n q^{-\epsilon n}(\frac{q^{mk}}{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})})^n$ of $\boldsymbol{A} \in (\mathsf{R}_q^{k\times m})^*$. $\square$

To determine practical values q, n, k, m for which the theorem can be applied, one needs to choose n, m, q big enough that $2\delta + 2^n q^{-\epsilon n}(\frac{q^{mk}}{(q^m-1)(q^m-q)\cdots(q^m-q^{k-1})})^n$ is close to zero, while having $\sigma = \sqrt{n\ln(2mn(1+1/\delta))/\pi}\, q^{\frac{1+k}{m}+\frac{\epsilon}{k}}$ as small as possible.

We now state all the result used in the proof of Theorem 2.

Lemma 6 ([20], Cor. 2.8). *Let $L' \subseteq L \subseteq \mathbb{R}^n$. For every $c \in \mathsf{R}^n, \delta \in (0,1/2), \sigma \geq \eta_\delta(L')$ we have $\Delta(D_{L,\sigma,c} \bmod L'; U(L/L')) \leq 2\delta$.*

Lemma 7 ([32] Lemma 3.5). *For any lattice $L \subseteq \mathsf{R}^n$ and $\delta \in (0,1)$, we have $\eta_\delta(L) \leq \sqrt{\ln(2n(1+1/\delta))/\pi}/\lambda_1^\infty(\widehat{L})$.*

Particular case for $S = \emptyset$ of [38, Lemma 7] gives:

Lemma 8 ([38]). *Let $\boldsymbol{a} \in \mathsf{R}_q^m$, then $\widehat{\boldsymbol{a}^\perp} = \frac{1}{q}L(\boldsymbol{a}^x)$.*

The following lemma is a generalization of [38, Lemma 8], where it was proved for the case $k = 1$. The proof directly follows the proof of [38, Lemma 8]. The main idea is to bound the probability of the opposite event by a negligible value. Thus, we define p as the probability that: for a randomly chosen $\boldsymbol{A}$, $L(\boldsymbol{A})$ contains a non-zero vector $\boldsymbol{t}$ with $\|\boldsymbol{t}\|_\infty < \frac{1}{\sqrt{n}}q^\beta$. To bound p, one needs to count the number of possible solutions a_{ij} which satisfy the relation $t_j = \sum_{i=1}^k a_{i,j}s_i \bmod q$.

Lemma 9. *Let $n \geq 4$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime q and $\mathsf{R}_q = \mathbb{Z}_q[x]/\langle\Phi\rangle$. Then*

$$\lambda_1^\infty(L(\boldsymbol{A})) \geq \frac{1}{\sqrt{n}}q^\beta, \quad \text{where} \qquad \beta = 1 - \frac{k}{2m} - \frac{\sqrt{k^2+4m\epsilon}}{2m} \geq 1 - \frac{k}{m} - \frac{\epsilon}{k}$$

except for a fraction of at most $2^n\frac{1}{q^{\epsilon n}}$ of all $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$.

Proof. Let $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ be chosen uniformly random. Let p denotes the probability that $L(\boldsymbol{A})$ contains a non-zero vector $\boldsymbol{t}$ with $||\boldsymbol{t}||_\infty < B = \frac{1}{\sqrt{n}}q^\beta$. This means that $\boldsymbol{A}$ was chosen such that there exists $\boldsymbol{t} = (t_1,\ldots,t_m) \in \mathsf{R}^m$ with $||t_j||_\infty < B$, $s_1,\ldots,s_k \in \mathsf{R}_q$, such that $t_j = \sum_{i=1}^k a_{i,j}s_i$ for every $j \in [m]$. Then p is exactly the fraction of all possible $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ for which the asserted inequality does not hold.

We bound p by summing over all possible $\boldsymbol{t}(\neq 0), s_1,\ldots,s_k$ the probabilities $p_{\boldsymbol{t},s_1,\ldots,s_k} = \Pr[\forall j \in [m], t_j = \sum_{i=1}^k a_{i,j}s_i]$ over the random choice of $\boldsymbol{A}$. Recall that if $\Phi = \prod_i \phi_i$ where ϕ_i are linear factors, then by CTR, we have $\mathsf{R}_q \cong \prod_i \mathsf{R}_q/\langle\phi\rangle \cong \mathbb{Z}_q^n$ with isomorphism $r \mapsto (r \mod \langle\phi_i\rangle)_{1\leq i\leq n}$.

We will use this to sum up probabilities:

$$p \leq \sum_{0\leq d<n} \sum_{\substack{h=\prod_{i\in S}\phi_i \\ S\subseteq\{1,\ldots,n\} \\ |S|=d}} \sum_{\substack{(s_1,\ldots,s_k)\in\mathsf{R}_q^k \\ \gcd(s_1,\ldots,s_k,\Phi)=h}} \sum_{\substack{\boldsymbol{t}\in R^m \\ 0<||t_j||_\infty<B \\ \forall j, h|t_j}} p_{\boldsymbol{t},s_1,\ldots,s_k}$$

Note that in the last sum we sum only over all $\boldsymbol{t} \in \mathsf{R}^m$ with $h|t_j$, since if $t_j = \sum_{i=1}^k a_{i,j}s_i$ for some $\boldsymbol{A} \in \mathsf{R}_q^{k\times m}$ and $(s_1,\ldots,s_k) \in \mathsf{R}_q^k$ with $\gcd(s_1,\ldots,s_k,\Phi) = h$, then also $h|t_j$.

We now bound $p_{\boldsymbol{t},s_1,\ldots,s_k}$ for any given $\boldsymbol{t}, s_1,\ldots,s_k$ with $\gcd(t_j,s_1,\ldots,s_k,\Phi) = h, \forall j$ where $h = \prod_{i\in S}\phi_i$ of degree d. It holds $p_{\boldsymbol{t},s_1,\ldots,s_k} = \prod_{i=1}^m p_{t_j,s_1,\ldots,s_k}$ where $p_{t_j,s_1,\ldots,s_k} = \Pr[t_j = \sum_{i=1}^k a_{i,j}s_i]$, since $\boldsymbol{A}$ is sampled uniformly random, thus its columns are sampled independently. Since $\mathsf{R}_q \cong \mathbb{Z}_q^n$, equation $t_j = \sum_{i=1}^k a_{i,j}s_i$ can be seen as n equations over $\mathbb{Z}_q$, each of the form $t_j = \sum_{i=1}^k a_{i,j}s_i \mod \phi_\ell$, where $\Phi = \prod_{\ell=1}^n \phi_\ell$. To determine the probability that such equations hold, we need to determine how many possible solutions these equations have (with a_{ij} as unknown). For $\ell \notin S$, there is a s_i such that $s_i \mod \phi_\ell \neq 0$, so equation $t_j = \sum_{i=1}^k a_{i,j}s_i \mod \phi_\ell$, $\ell \notin S$, has precisely q^{k-1} solutions in $(\mathsf{R}_q/\langle\phi_i\rangle)^k$, since one of the variables can be expressed w.r.t others. On the other hand, for $\ell \in S$, we have $s_i \mod \phi_\ell = 0$ for all $i \in [k]$ and consequently $t_j = \sum_{i=1}^k a_{i,j}s_i \mod \phi_\ell = 0$ meaning that every $a_{1,j},\ldots,a_{k,j}$ is a solution. Thus in this case there are q^k solutions. Now putting together and by the fact that $|S| = d$, we have $q^{(k-1)(n-d)+kd}$ solutions for $t_j = \sum_{i=1}^k a_{i,j}s_i \mod \Phi$.

$$p_{\boldsymbol{t},s_1,\ldots,s_k} = \prod_{i=1}^m p_{t_i,s_1,\ldots,s_k} \leq \prod_{i=1}^m \left(\frac{q^{(k-1)(n-d)+kd}}{q^{kn}}\right) = \frac{1}{q^{m(n-d)}}$$

It was proved in [38, Lemma 8] that if $d \geq \beta n$, there is no t_j divisible by h of degree d (as defined above) such that $||t_j||_\infty \leq B = \frac{1}{\sqrt{n}}q^\beta$, and that for $d < \beta n$ there are at most $(2B)^{n-d}$ possible t_j.

For $h = \prod_{i\in S}\phi_i, |S| = d$, there are q^{n-d} possible $s_j \in \mathsf{R}_q$ such that $h|s_j$. Clearly the number of all possible $S \subseteq \{1,\ldots,n\}$ is 2^n. Thus we can bound:

$$p \leq 2^n \max_{d\leq\beta n} \frac{q^{k(n-d)}(2B)^{m(n-d)}}{q^{m(n-d)}}$$

Since $n \geq 4$, it follows $2B = 2\frac{1}{\sqrt{n}}q^\beta \leq q^\beta$. This implies

$$p \leq 2^n \max_{d\leq\beta n} \frac{1}{q^{(m-k-\beta m)(n-d)}} = 2^n \frac{1}{q^{(m-k-\beta m)(n-\beta n)}},$$

where the last equation follows since $m - k - \beta m > 0$, which can be easily checked using $\beta = 1 - \frac{k}{2m} - \frac{\sqrt{k^2+4m\epsilon}}{2m}$. Moreover, putting the value of β in the last bound, we get the claimed result. □

C Proof Of Theorem 4

Game	Description	justification
$\mathbf{G}_0$	$\boldsymbol{s}_i \xleftarrow{R} D_{\sigma_1}$, $\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle$, $\mathsf{sk} = \sum_i y_i \boldsymbol{s}_i$; $\mathbf{ct}_0 = \boldsymbol{a}r + \boldsymbol{f}_0$, $\mathsf{ct}_i = \mathsf{pk}_i r + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Real Game
$\mathbf{G}_1$	$\boldsymbol{s}_i \xleftarrow{R} D_{\sigma_1}$, $\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle$, $\mathsf{sk} = \sum_i y_i \boldsymbol{s}_i$; $\mathbf{ct}_0 = \boldsymbol{a}r + \boldsymbol{f}_0$, $\mathsf{ct}_i = \langle \mathbf{ct}_0, \boldsymbol{s}_i \rangle - \langle \boldsymbol{f}_0, \boldsymbol{s}_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Identical
$\mathbf{G}_2$	$\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle$, $\mathsf{sk} = \sum y_i \boldsymbol{s}_i$; $\boxed{\mathbf{ct}_0 = \boldsymbol{u} + \boldsymbol{a}r + \boldsymbol{f}_0}$ where $\boldsymbol{u} = (u_1, \ldots, u_m) \leftarrow (\mathsf{R}_q^*)^m$, $\mathsf{ct}_i = \langle \mathbf{ct}_0, \boldsymbol{s}_i \rangle - \langle \boldsymbol{f}_0, \boldsymbol{s}_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	mhe-RLWE
$\mathbf{G}_3$	$\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}_i \rangle$, $\mathsf{sk} = \sum y_i \boldsymbol{s}_i$; $\mathbf{ct}_0 = \boldsymbol{u} + \boldsymbol{a}r + \boldsymbol{f}_0$, $\mathsf{ct}_i = \mathsf{pk}_i r + \langle \boldsymbol{u}, \boldsymbol{s}_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	Identical
$\mathbf{G}_4$	$\boxed{\boldsymbol{s}_i = \boldsymbol{s}^* \alpha_i + \boldsymbol{s}'_i}$, $\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}^* \rangle \alpha_i + \langle \boldsymbol{a}, \boldsymbol{s}'_i \rangle$, $\mathsf{sk} = \sum y_i \boldsymbol{s}'_i$; $\alpha_i = (x_i^1 - x_i^0)$, $\mathbf{ct}_0 = \boldsymbol{u} + \boldsymbol{a}r + \boldsymbol{f}_0$, $\mathsf{ct}_i = \mathsf{pk}_i r + \langle \boldsymbol{u}, \boldsymbol{s}^* \rangle \alpha_i + \langle \boldsymbol{u}, \boldsymbol{s}'_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	sta.arg.
$\mathbf{G}_5$	$\mathsf{pk}_i = \boxed{u'} \alpha_i + \langle \boldsymbol{a}, \boldsymbol{s}'_i \rangle$, $\mathsf{sk} = \sum y_i \boldsymbol{s}'_i$; $\mathbf{ct}_0 = \boldsymbol{u} + \boldsymbol{a}r + \boldsymbol{f}_0$, $\mathsf{ct}_i = \mathsf{pk}_i r + \boxed{u''} \alpha_i + \langle \boldsymbol{u}, \boldsymbol{s}'_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$	LHL

Fig. 3: Overview of games for adaptively security IPFE.

Proof. We start with game $\mathbf{G}_0$ which is the real game associated with a chosen bit b, while the last game is independent of bit b.

$\boxed{\mathbf{G}_0}$: is the real game associated with bit b.

$\boxed{\mathbf{G}_1}$: is the same as the previous game when ct_i is rewritten based on $\mathbf{ct}_0$ (replacing pk_i value) i.e.,

$$\mathsf{ct}_i = \langle \mathbf{ct}_0, \boldsymbol{s}_i \rangle - \langle \boldsymbol{f}_0, \boldsymbol{s}_i \rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$$

Clearly, $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_1}(\kappa) = \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_0}(\kappa)$.

$\boxed{\mathbf{G}_2}$: is similar to the game $\mathbf{G}_1$, except that, $\mathbf{ct}_0$ is replaced with $\boldsymbol{u} + \boldsymbol{a}r + \boldsymbol{f}_0$ where $\boldsymbol{u}$ is a m-dimensional vector-of-polynomials uniformly sampled from R_q^m with property that at least one element of $\boldsymbol{u}$ is invertible (recall that this holds also for the original $\mathbf{ct}_0$).

The proof of indistinguishability of $\mathbf{G}_1$ and $\mathbf{G}_2$ is similar to the counterpart transition for selective-case in Theorem 3, except that here $\boldsymbol{u}$ is a vector-of-polynomials. Hence one needs to use mhe-RLWE assumption with m samples (see Lemma 5). Moreover, the probability p that the uniformly-sampled vector fits the invertability condition is non-negligible (in fact, closer to 1). Thus, distinguishing $\mathbf{G}_1$ from $\mathbf{G}_2$ breaks the version of mhe-RLWE, in which only the samples with this property are given to the adversary. Or equivalently, mhe-RLWE can be solved with non-negligible probability. Thus,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_2}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_1}(\kappa)| \leq \frac{1}{p}\mathsf{Adv}^{\mathsf{mheRLWE}}_{\mathcal{B},m}(\kappa) + 2\epsilon$$

where $\sigma_3 = \sqrt{m}\delta$ and $\epsilon = 2^{-\kappa}$ satisfy the condition $\Gamma_{\delta I_n, \delta I_n} \geq \eta_\epsilon(\mathbb{Z}^n)$.

$\boxed{\mathbf{G}_3}$: is the same as the game $\mathbf{G}_2$ when ct_i is rewritten based on pk_i (replacing $\mathbf{ct}_0$), i.e., $\mathsf{ct}_i = \mathsf{pk}_i r + \langle \boldsymbol{u}, \boldsymbol{s}_i\rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$.
Since the games are identical to the adversary, $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_3}(\kappa) = \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_2}(\kappa)$.

From this point, the games are defined in the selective-setting while the adjacent games are statistically-indistinguishable. An elegant property of statistical-indistinguishability is that, the selective and adaptive security can be equivalent for the proper choice of parameters. More precisely, if the selective version of two adjacent games are statistically-indistinguishable one can lift the security to adaptive by a proper choice of parameters via a complicity leveraging mechanism and without losing any factor of security.

To proceed to the next game, at first define $\boldsymbol{S} = (\boldsymbol{S}_1, \ldots, \boldsymbol{S}_m)$ as the array of matrices associated with the master secret-key where the i-th row in matrix $\boldsymbol{S}_j$ is $\boldsymbol{s}_{ij}$ for $j \in [m]$ and each $\boldsymbol{s}_{ij}$ is the vector representation of the corresponding polynomial s_{ij} (i.e., $\boldsymbol{S}_j$ has dimension $\ell \times n$).

$\boxed{\mathbf{G}_4^*}$: Let $\mathbf{G}_3^*$ be the selective version of $\mathbf{G}_3$. Then $\mathbf{G}_4^*$ is similar to the game $\mathbf{G}_3^*$, except that, in matrix $\boldsymbol{S}_j$ the k-th column is replaced with $\bar{\boldsymbol{s}}_k^j + (s_k^j)^*\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\mathbf{x}^1 - \mathbf{x}^0)$, scalar $(s_k^j)^*$ is sampled from $D_{\sigma'}$ and vector $\bar{\boldsymbol{s}}_k^j$ is sampled from D_Σ with $\Sigma = \sigma_1^2 I_\ell - \sigma' \boldsymbol{\alpha}^T\boldsymbol{\alpha}$. We call $\boldsymbol{S}'$ the new representation of the master-secret key (after applying the mentioned changes on $\boldsymbol{S}$). The transition from $\mathbf{G}_3^*$ to $\mathbf{G}_4^*$ is similar to the transition for the counterpart games in the security proof of our selective-secure construction. Thus we have,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_4^*}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_3^*}(\kappa)| \leq 2nm\epsilon$$

where $\epsilon = 2^{-\kappa}/nm$ and Σ, σ' satisfy the condition $\Gamma_{\Sigma, \sigma'^2\boldsymbol{\alpha}^T\boldsymbol{\alpha}} \geq \eta_\epsilon(\mathbb{Z}^n)$.

Changing $\boldsymbol{S}$ to $\boldsymbol{S}'$ as above, will consequently change $\boldsymbol{s}_i$, as the i-th row of $\boldsymbol{S}$, to the form $\boldsymbol{s}_i = \boldsymbol{s}^*\alpha_i + \boldsymbol{s}'_i = (s_1^*\alpha_i + s'_{i1}, \ldots, s_m^*\alpha_i + s'_{im})$ in $\boldsymbol{S}'$, where $\boldsymbol{s}'_i$ is a nm-dimensional vector and $s_j^* = ((s_1^j)^*, \ldots, (s_n^j)^*)$ (where $(s_k^j)^*$ is defined above). Thus one can rewrite pk_i, ct_i and $\mathbf{sk}_y$ w.r.t the new form of $\boldsymbol{s}_i$ (with its representation as the vector-of-polynomials). Meaning that,

$$\mathsf{pk}_i = \langle \boldsymbol{a}, \boldsymbol{s}^*\rangle\alpha_i + \langle \boldsymbol{a}, \boldsymbol{s}'_i\rangle, \quad \mathbf{sk}_y = \sum_i y_i s'_i$$

$$\mathsf{ct}_i = \mathsf{pk}_i r + \langle \boldsymbol{u}, \boldsymbol{s}^*\rangle\alpha_i + \langle \boldsymbol{u}, \boldsymbol{s}'_i\rangle + f_i + \lfloor q/K \rfloor x_i^b 1_{\mathsf{R}}$$

Note that $\mathbf{sk}_y$ can be computed without the terms $\boldsymbol{s}^*$, since the adversary is only allowed to query for $\mathbf{y}$, such that $\langle \mathbf{y}, \boldsymbol{\alpha}\rangle = 0$.

$\boxed{\mathbf{G}_5^*}$: is similar to the game $\mathbf{G}_4^*$, except that, the product $\langle \boldsymbol{u}, \boldsymbol{s}^* \rangle$ is replaced with a uniform polynomial u'. To prove the indistinguishability of $\mathbf{G}_4^*$ and $\mathbf{G}_5^*$, we use the ring version of LHL from Theorem 2. In particular, if the mapping $f_{a,u}(x) = \left(\begin{smallmatrix} \boldsymbol{a} \\ \boldsymbol{u} \end{smallmatrix}\right)$ is surjective, then for properly selected parameters with $k = 2$, the theorem provides that values $\langle \boldsymbol{a}, \boldsymbol{s}^* \rangle$ and $\langle \boldsymbol{u}, \boldsymbol{s}^* \rangle$ are statistically indistinguishable from uniformly random (u', u''). We have,

$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_5^*}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_4^*}(\kappa)| \leq \Delta[\left(\begin{smallmatrix} \boldsymbol{a} \\ \boldsymbol{u} \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{a} \\ \boldsymbol{u} \end{smallmatrix}\right) s^*; U((\mathsf{R}_q^{2\times m})^*, \mathsf{R}_q^2)] \cdot (1 - p^*) + p^*,$$

where p^* is the probability that $f_{a,u}$ is not surjective and Δ can be set through Theorem 2 for $k = 2$ and $m \geq 3$. Note that $f_{a,u}$ is not surjective only if $\boldsymbol{u} = s\boldsymbol{a}$ for a scalar $s \in \mathbb{Z}_q$, due to the fact that both $\boldsymbol{u}$ and $\boldsymbol{a}$ have invertible elements. Thus $p^* = \text{negl}(\kappa)$.

Advantage of the adversary in game $\mathbf{G}_5^*$: similar to the discussion in the last game for sel-IPFE (Theorem 3), one can see game $\mathbf{G}_5^*$ is independent of bit b and the advantage of the adversary in this game is zero.

Indistinguishability among adaptive versions $\mathbf{G}_3^*, \mathbf{G}_4^*$ and $\mathbf{G}_5^*$: The complexity leveraging (CL) technique is a common way to lift the security from selective to the adaptive by guessing the challenge in advance, though this would be possible with the cost of losing a factor of security depending on the size of message space. Thus, when CL is used alongside a computational assumption, it can be used only for a small message space. In our security proof we use CL only for games $\mathbf{G}_4^*, \mathbf{G}_5^*$ where no computational assumption is used. More precisely, thanks to the statistical argument in these game one can set the statistical distance (by proper choice of parameters) such that the effect of message-space size in CL can not decrees the security-amount.

Now, let $\mathbf{G}_i$ and $\mathbf{G}_i^*$ stand for the adaptive and selective versions of the corresponding games, respectively, where $i = 3, 4, 5$.

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_i}(\kappa) \leq B_x^{2\ell} \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}^*,\mathbf{G}_i^*}(\kappa) \qquad \text{by CL} \;\Rightarrow$$
$$|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_{i+1}}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_i}(\kappa)| \leq B_x^{2\ell} |\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}^*,\mathbf{G}_{i+1}^*}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}^*,\mathbf{G}_i^*}(\kappa)| \quad i = 3, 4$$

Note that one can control the statistical distance via setting the parameters such that $|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}^*,\mathbf{G}_{i+1}^*}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A}^*,\mathbf{G}_i^*}(\kappa)| \leq \epsilon (2B_x)^{-2\ell}$. Then we would have, $|\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_{i+1}}(\kappa) - \mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathbf{G}_i}(\kappa)| \leq \epsilon$.

Putting together, as the concerned games are statistically secure, one can achieve that any two games are indistinguishable by setting the parameters in such a way that, after applying CL (which multiplies the probability of distinguishing in a big factor) the adaptive versions of the games are still indistinguishable. While this methodology results in the adaptive security even for a big message space, it comes with the cost of bigger parameters. For example in the change from game $\mathbf{G}_3$ to $\mathbf{G}_4$, we need to have $\epsilon = \epsilon'/(2B_x)^{2\ell}$ as the probability of distinguishing the selective games (where ϵ' is the probability of distinguishing their adaptive versions), then this increase the value of $\eta_\epsilon(\mathbb{Z}^n)$ and consequently related parameters (see the parameter setting). $\square$

C.1 Parameters Setting for adaptively-secure IPFE

Here we overview the requirement for the parameters in our adaptively-secure IPFE scheme.
Correctness. Requires $lB_y(\sqrt{\kappa}\sigma_3+mn\kappa\sigma_1\sigma_2) < \lfloor q/2K \rfloor$ and $q >> K > lB_xB_y$.
Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$. Needs $\sigma_3 = \sqrt{m}\sigma_2$, $\mathit{\Gamma}_{\sigma_2 I_n, \sigma_2 I_n} \geq \eta_\epsilon(\mathbb{Z}^n)$ with $\epsilon = 2^{-\kappa}$ and also all the parameter setting from mhe-RLWE assumption i.e., $\sigma\sqrt{1-\frac{1}{\sigma_2^2}(\sigma nC\sqrt{l+2})^2} > \eta_\epsilon(\mathbb{Z}^{n+n\ell})$ where $\|\boldsymbol{s}_i\|_\infty$, $\|e_i\|_\infty \leq C$ and σ is the parameter for the hardness of RLWE with m samples. By Lemma 2 in Appendix A.2, one can set $C = \sqrt{\kappa}\sigma_1$.
Transition from $\mathbf{G}_3$ to $\mathbf{G}_4$. Needs $\sigma_1 \geq \sqrt{\ell}2B_x\sigma'$ for non-negatives σ' where σ_1, σ', satisfy $\mathit{\Gamma}_{\Sigma, \sigma'^2 \boldsymbol{\alpha}^T\boldsymbol{\alpha}} \geq \eta_\epsilon(\mathbb{Z}^n)$ with $\epsilon' = 2^{-\kappa}/nm$ and $\epsilon = \epsilon'/(2B_x)^{2\ell}$.
Transition from $\mathbf{G}_4$ to $\mathbf{G}_5$. Needs the parameter setting from LHL: n be a power of 2 such that $\mathit{\Phi} = x^n + 1$ splits into n linear factors modulo prime q, $m \geq 3$, $\delta \in (0, 1/2)$, $\epsilon > 0$, $\sigma' \geq \sqrt{n\ln(2mn(1+1/\delta)/\pi}q^{\frac{2}{m}+\frac{\epsilon}{2}}$ such that $(2\delta + 2^n\frac{1}{(q)^{\epsilon n}}(\frac{q^{2m}}{(q^m-1)(q^m-q)})^n) \cdot (2B_x)^{2\ell}$ is negligible.

D MCFE and DMCFE

D.1 Security Definition

In a MCFE scheme each client i has a different secret key ek_i, which can be individually corrupted. In the following, we formally define the security notion of a MCFE scheme.

Definition 7 (Security of MCFE). *Let* MCFE *be an MCFE scheme and* Labels *a label set. For $b \in \{0,1\}$, we define the experiment $\mathsf{IND}_{\mathcal{A}}^{b,\mathsf{MCFE}}$ as follows:*

Initialize: *The challenger runs* $(\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in[\ell]}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$ *and sends* pp *to* $\mathcal{A}$.
Corruption queries: *The adversary $\mathcal{A}$ can choose to corrupt a client i meaning that it would receive the encryption key ek_i. We denote by $\mathcal{CS}$ the set of corrupted clients at the end of the experiment.*
Challenge queries: *The adversary $\mathcal{A}$ can ask for multiple queries, where on a query $(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \gamma)$, $\mathcal{A}$ would receive $\mathsf{ct}_{i,\gamma} = \mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i^\beta, \gamma)$.*
Key Generation queries: *on a query $\mathbf{y}$, it would receive $\mathsf{sk}_y = \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$.*

The challenger aborts if the following Condition (*) *is not satisfied:*

- *If $i \in \mathcal{CS}$: for any challenge query $\mathbf{x}_i^0, \mathbf{x}_i^1$, $\mathbf{x}_i^0 = \mathbf{x}_i^1$.*
- *Denote with $\mathbf{x}_{i,q_i,\gamma}$, the message associated with the q_i-th challenge on the label γ for client i. For any label $\gamma \in$* Labels*, for any family of queries $\{\mathbf{x}_{i,q_i,\gamma}^0, \mathbf{x}_{i,q_i,\gamma}^1\}_{i\in[\ell]}$, for any query $\mathbf{y}$, we require that: $f_{\mathbf{y}}(\mathbf{x}_{1,q_1,\gamma}^1, \ldots, \mathbf{x}_{\ell,q_\ell,\gamma}^1) = f_{\mathbf{y}}(\mathbf{x}_{1,q_1,\gamma}^0, \ldots, \mathbf{x}_{\ell,q_\ell,\gamma}^0)$.*

We say that the MCFE scheme is secure if for any PPT adversary $\mathcal{A}$ there is a negligible function negl such that,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MCFE}}(1^{\kappa}) = |\Pr[\mathsf{IND}_{\mathcal{A}}^{1,\mathsf{MCFE}}(1^{\kappa}) = 1] - \Pr[\mathsf{IND}_{\mathcal{A}}^{0,\mathsf{MCFE}}(1^{\kappa}) = 1]| \leq \text{negl}(\kappa)|$$

In the above definition, we have assumed that the adversary asks queries for all the clients. If the algorithm KeyGen receives the label γ as input (this implies that also in the condition (*) we have $(\mathbf{y}, \gamma)$, rather than $\mathbf{y}$), we call the scheme an *identity-based MCFE scheme*.

If all the corruption queries are issued at the very beginning of the game, we call the scheme secure against *static corruptions*. The security of MIFE supporting corruptions can be defined similar to the security of MCFE when there is only one label (and so one can ignore to write it in the security notion).

D.2 Proof of Theorem 5

We define a sequence of the games w.r.t labels $\gamma \in \mathsf{Labels} = \{1, \ldots, L\}$ such that each time for a fixed label, we replace $\mathbf{x}_{i,\gamma}^{0}$ with $\mathbf{x}_{i,\gamma}^{1}$.

$\mathbf{G}_0$: is the real game associated with $b = 0$.
$\mathbf{G}_{0,\gamma}$: is defined recursively as similar to $\mathbf{G}_{1,\gamma-1}$, except that, for $\gamma' \neq \gamma$ the random oracle output for $(\boldsymbol{u}_i', \gamma')$ is set as $H(\boldsymbol{u}_i', \gamma') = \boldsymbol{r}_{i,\gamma'} - \boldsymbol{u}_i$, where $\boldsymbol{u}_i$ is the secret key obtained from the MIFE setup (run in $\mathsf{Setup}(1^{\kappa}, 1^{\ell}, 1^{k})$) and $\boldsymbol{r}_{i,\gamma'}$ is chosen randomly. Moreover, $H(\boldsymbol{u}_i', \gamma)$ is set to a random value. We define $\mathbf{G}_{1,0} = \mathbf{G}_0$.
$\mathbf{G}_{1,\gamma}$: is defined recursively as similar to $\mathbf{G}_{0,\gamma}$, except that, $\mathbf{x}_{i,\gamma}^{0}$ is replaced with $\mathbf{x}_{i,\gamma}^{1}$ for all $i \in [\ell]$.
$\mathbf{G}_1$: is the real game associated with $b = 1$ and equal to $\mathbf{G}_{1,L}$.
We go through these games in the following order:

$$\mathbf{G}_0 \to \mathbf{G}_{0,1} \to \mathbf{G}_{1,1} \to \mathbf{G}_{0,2} \to \mathbf{G}_{1,2} \ldots \mathbf{G}_{1,L-1} \to \mathbf{G}_{0,L} \to \mathbf{G}_1$$

Transition from $\mathbf{G}_{1,\gamma-1}$ to $\mathbf{G}_{0,\gamma}$. These two games are perfectly indistinguishable by the definition of RO. Let $\mathcal{A}$ be the attacker trying to distinguish these two games. The simulator $\mathcal{B}$ simulates all the queries by the real algorithms except for the RO queries. By the execution of real algorithms, $\mathcal{B}$ obtains values $\boldsymbol{u}_i', \boldsymbol{u}_i$ for all $i \in [\ell]$. It simulates all the RO queries as follows:

- if it is a new RO query for $H(\boldsymbol{u}_i', \gamma')$ where a label $\gamma' \neq \gamma$, $\mathcal{B}$ chooses a fresh randomness $\boldsymbol{r}_{i,\gamma'}$ and answers by $H(\boldsymbol{u}_i', \gamma') = \boldsymbol{r}_{i,\gamma'} - \boldsymbol{u}_i$ and adds it to a RO-list which is empty at the beginning of the simulation. If $\mathcal{A}$ queries for $H(\boldsymbol{u}_i', \gamma)$ or any other value not listed above, then it just answers by a random value. If it is not a new RO query, it answers by the existing values in the RO-list.

Since values $\boldsymbol{r}_{i,\gamma'}$ are uniformly random, these two games are perfectly indistinguishable.

Transition from $\mathbf{G}_{0,\gamma}$ to $\mathbf{G}_{1,\gamma}$. Again let $\mathcal{A}$ be the adversary trying to distinguish between these two games. We will show that the simulator $\mathcal{B}$ can use $\mathcal{A}$ to break the security of MIFE scheme. Hence assume that $\mathcal{B}$ is also an adversary to the security of MIFE. We explain how $\mathcal{B}$ simulates the challenge for the adversary $\mathcal{A}$:

- Corruption Queries: upon receiving the set of clients that $\mathcal{A}$ wants to corrupt, it forwards them to MIFE-challenger to receive the secret keys of MIFE. Then it includes $\boldsymbol{u}'_i$ for each corrupted i and sends them to $\mathcal{A}$ as the secret keys of corrupted parties in MCFE.
- RO queries: when $\mathcal{A}$ sends a RO query, it answers by the programmed values (as in the previous game), unless if the query is for $H(u'_i, \gamma')$ where $\gamma' \neq \gamma$ and i is uncorrupted. In that case $\mathcal{B}$ simply aborts since it does not know u_i needed to answer. We argue below that this event happens with a negligible probability.
- Encryption queries: when it receives an encryption query on the challenges $\mathbf{x}_i^0, \mathbf{x}_i^1$ w.r.t the label $\gamma' \neq \gamma$, it sends $\mathsf{ct}_{i,\gamma'} = \mathsf{Add}(\mathsf{FE.Enc}(\mathsf{mpk}'_i, \mathbf{x}_i^0), \boldsymbol{r}_{i,\gamma'})$ to $\mathcal{A}$ for $\gamma' > \gamma$, and for $\gamma' < \gamma$ it sends $\mathsf{ct}_{i,\gamma'} = \mathsf{Add}(\mathsf{FE.Enc}(\mathsf{mpk}'_i, \mathbf{x}_i^1), \boldsymbol{r}_{i,\gamma'})$. Here we are using Property 2 of the MIFE scheme which is based on single-input FE scheme FE, and also the fact that FE is a public-key encryption. To compute an encryption for the label γ, it forwards the queries to the MIFE challenger, which responds with ct_i^b. Then $\mathcal{B}$ sets $\mathsf{ct}_{i,\gamma} = \mathsf{ct}_i^b + H(\boldsymbol{u}'_i, \gamma)$ and sends it to $\mathcal{A}$.
- Key Generation queries: $\mathcal{B}$ simulates these queries by sending a key generation query to its challenger. After receiving $\mathsf{sk}'_y = \sum_i \boldsymbol{u}_i \mathbf{y}_i$, it computes $\mathsf{sk}''_{i,y,\gamma} = \sum_i \mathbf{y}_i H(\boldsymbol{u}'_i, \gamma) + \mathsf{sk}'_y$, and for each $\gamma' \neq \gamma$ it sets $\mathsf{sk}''_{i,y,\gamma'} = \sum_i \mathbf{y}_i \boldsymbol{r}_{i,\gamma'}$ and sends them to $\mathcal{A}$.

Clearly, if $\mathcal{B}$ does not abort and $\mathcal{A}$ succeeds in its game with a probability ϵ, then $\mathcal{B}$ breaks the security of MIFE with the probability ϵ. The probability that $\mathcal{B}$ aborts is negligible, this is due to the fact that the keys $\boldsymbol{u}'_i$ are chosen uniformly at random from arbitrary big U and so the probability that $\mathcal{A}$ sends a RO-query on $\boldsymbol{u}'_i$ that was chosen for a uncorrupted i is negligible.

D.3 Proof of Proposition 1

To argue the security of the compilers from Section 6 used on our IPFE scheme based on RLWE assumption we shall prove that Property 2 holds. On the other hand, Property 1 is needed for the correctness. To avoid the rigorous definition of Property 1 we instead directly prove the correctness of MCFE instantiated by our IPFE schemes.

1. Correctness of MIFE based on RLWE: In the decryption of our single-input IPFE scheme, we define $\mathsf{FE.Dec}_1(\mathsf{ct}, \mathsf{sk}_y)$ to be value d in the definition of decryption in Section 4 and Section 5 and encoding $\mathcal{E}(z, \mathsf{noise}) = \lfloor q/K \rfloor z 1_{\mathsf{R}} + \mathsf{noise}$. Then $\mathsf{FE.Dec}_2(d)$ is the process of extracting the nearest multiple of $\lfloor q/K \rfloor$ in the

Game	Description	justification
$\mathbf{G}_0$	$\mathsf{ct}_{i,\gamma'} \leftarrow \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}_i + H(\boldsymbol{u}'_i, \gamma'))$ $\mathsf{sk}'_y \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{msk}_i, \mathbf{y})$ $\mathsf{sk}''_{i,y,\gamma'} = \sum_i H(\boldsymbol{u}'_i, \gamma')\mathbf{y}_i + \mathsf{sk}'_y$	Real Game
$\mathbf{G}_{0,\gamma}$	$\boxed{H(\boldsymbol{u}'_i, \gamma') = \boldsymbol{r}_{i,\gamma'} - \boldsymbol{u}_i,\ \gamma' \neq \gamma}$. $\mathsf{ct}_{i,\gamma'} = \begin{cases} \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}^1_i + H(\boldsymbol{u}'_i, \gamma')) & \gamma' < \gamma \\ \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}^0_i + H(\boldsymbol{u}'_i, \gamma')) & \gamma' = \gamma \\ \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}^0_i + H(\boldsymbol{u}'_i, \gamma')) & \gamma' < \gamma \end{cases}$ $\mathsf{sk}'_y \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{msk}_i, \mathbf{y})$ $\mathsf{sk}''_{i,y,\gamma'} = \sum_i H(\boldsymbol{u}'_i, \gamma')\mathbf{y}_i + \mathsf{sk}'_y$	
$\mathbf{G}_{1,\gamma}$	$\mathsf{ct}_{i,\gamma'} = \begin{cases} \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}^1_i + H(\boldsymbol{u}'_i, \gamma')) & \gamma' < \gamma \\ \mathsf{MIFE.Enc}(\mathsf{msk}_i, \boxed{\mathbf{x}^1_i} + H(\boldsymbol{u}'_i, \gamma')) & \gamma' = \gamma \\ \mathsf{MIFE.Enc}(\mathsf{msk}_i, \mathbf{x}^0_i + H(\boldsymbol{u}'_i, \gamma')) & \gamma' < \gamma \end{cases}$ $\mathsf{sk}'_y \leftarrow \mathsf{MIFE.KeyGen}(\mathsf{msk}_i, \mathbf{y})$ $\mathsf{sk}''_{i,y,\gamma'} = \sum_i H(\boldsymbol{u}'_i, \gamma')\mathbf{y}_i + \mathsf{sk}'_y$	MIFE

Fig. 4: Overview of games for MCFE-compiler

coefficients of d. Then the algorithm $\mathsf{MIFE.Dec}((\mathsf{ct}_{i,\gamma})_i, (\{\mathsf{sk}_{i,y}\}_i, \mathsf{sk}''_{y,\gamma}))$ computes values D_i and outputs $\mathsf{FE.Dec}_2(d)$, where d is

$$\begin{aligned}
&\sum_i D_i - \lfloor q/K \rfloor (\sum_i \boldsymbol{u}_i \mathbf{y}_i + \sum_i H(\boldsymbol{u}'_i, \gamma)\mathbf{y}_i) 1_{\mathsf{R}} \\
=&\sum_i \mathsf{noise}_i + \lfloor q/K \rfloor (\langle \mathbf{x}_i + \boldsymbol{u}_i + H(\boldsymbol{u}'_i, \gamma), \mathbf{y}_i \rangle - \langle \boldsymbol{u}_i + H(\boldsymbol{u}'_i, \gamma), \mathbf{y}_i \rangle) 1_{\mathsf{R}} \\
=&\sum_i \mathsf{noise}_i + \lfloor q/K \rfloor \sum_i \langle \mathbf{x}_i, \mathbf{y}_i \rangle 1_{\mathsf{R}}
\end{aligned}$$

Now for the correctness we need $\|\sum_i \mathsf{noise}_i\|_\infty < \lfloor q/2K \rfloor$. To be clear, we apply the MIFE compiler over our IPFE scheme, where the ciphertexts of our IPFE are still defined w.r.t K but we choose K properly. For all the indices i, w.l.g we assume that the maximum value of inner-product $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$ is kB_xB_y where $\|\mathbf{x}_i\|_\infty \leq B_x$ and $\|\mathbf{y}_i\|_\infty \leq B_y$. So we have $\sum_i \langle \mathbf{x}_i, \mathbf{y}_i \rangle \leq \ell k B_x B_y$ and we define K to be any value greater than $\ell k B_x B_y$, where ℓ is the number of clients. Then

$$\left\| \sum_i \mathsf{noise}_i \right\|_\infty \leq \sum_i \|\mathsf{noise}_i\|_\infty \leq \ell \|\mathsf{noise}_i\|_\infty \leq \ell(\lfloor q/2kB_xB_y \rfloor) \leq \lfloor q/2K \rfloor,$$

hence the output is correct.
2. Linear Encryption: Let Enc be the encryption algorithm of our IPFE scheme. We define $\mathsf{Add}(\mathsf{Enc}(\mathbf{x}_1), \mathbf{x}_2) = \mathsf{Enc}(\mathbf{x}_1) + \lfloor q/K \rfloor \mathbf{x}_2 1_{\mathsf{R}} \mod \mathsf{R}_q$. It is easy to check that its distribution is identical to $\mathsf{Enc}(\mathbf{x}_1 + \mathbf{x}_2)$.

D.4 Decentralized MCFE

In this section we present a compiler to transfer our identity-based MCFE to its decentralized version (identity-based DMCFE). The syntax of a DMCFE scheme is defined similar to the syntax of MCFE, with the only difference that the setup and key generation algorithms are decentralized, hence each client generates its share of functional key via the algorithm $\mathsf{KeyGen}(\mathsf{msk}_i, \mathbf{y}, i, \gamma)$[13], and then a public algorithm $\mathsf{KeyCombin}$ can aggregate the shares. The security notion is also similar to the security notion of MCFE except that key generation queries are asked w.r.t each client separately.

In [3] authors presented a general compiler to transfer a MCFE (or MIFE) scheme to a decentralized MCFE (or MIFE) scheme. Slightly more in details, they prove that if the functional key for the underlying scheme satisfies a special property, then it can be used in their compiler.

Property 3 (special functional key). Let the secret key of client i be composed of two main parts sk_i^1 and sk_i^2. Then the functional key should be composed of two main parts $\mathsf{sk}_y = (\mathsf{sk}_y^1, \mathsf{sk}_y^2)$ as well, such that sk_y^1 is a function of first part of the master secret key $\mathsf{msk}^1 = (\mathsf{sk}_1^1, \ldots, \mathsf{sk}_n^1)$ (and vector $\mathbf{y}$, but not the second part of secret keys) and sk_y^2 is the inner-product of vector[14] $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and the second parts of the master secret key $\mathsf{msk}^2 = (\mathsf{sk}_1^2, \ldots, \mathsf{sk}_n^2)$. Namely, $\mathsf{sk}_y^2 = \langle \mathsf{msk}^2, \mathbf{y} \rangle$.

Our identity based MCFE scheme also has this property, but the fact that the secret key $H(\boldsymbol{u}_i', \gamma)$ depends on the label γ affects the compiler slightly. Nevertheless, the compiler of [3] can be adjusted for our case via a slight change.

Let $\{H_i : \mathsf{Labels} \to \mathbb{Z}_q^{\ell k}\}_{i \in [\ell]}$ be a set of hash functions with the property that $\sum_{i \in [\ell]} H_i(\gamma) = 0$ for every $\gamma \in \mathsf{Labels}$. We postpone the discussion on how to generate such functions, and directly present the modified compiler. In the following compiler MCFE is our identity-based MCFE scheme described in Section 6.

DMCFE-Compiler: From MCFE to DMCFE

- $\mathsf{Setup}(1^\kappa, 1^\ell, 1^k)$: for each $i \in [\ell]$ it generates H_i such that $\sum_{i \in [\ell]} H_i(\gamma) = 0$ holds for every $\gamma \in \mathsf{Labels}$, and it runs $(\mathsf{mpk}_i', \mathsf{msk}_i')_i \leftarrow \mathsf{MCFE.Setup}(1^\kappa, 1^\ell, 1^k)$. It outputs $\mathsf{msk}_i = (\mathsf{msk}_i', H_i)$ as the secret key of user i and $\mathsf{pp} = (\mathsf{mpk}_i')_i$, where $\mathsf{msk}_i' = (\boldsymbol{u}_i, \boldsymbol{u}_i')$.
- $\mathsf{KeyGen}(\mathsf{msk}_i, \mathbf{y}, \gamma, i)$: it runs $(\mathsf{sk}_{i,y,\gamma}, \perp) \leftarrow \mathsf{MCFE.KeyGen}(\mathsf{msk}_i', \mathbf{y})$[15] and sets $\mathsf{sk}_{i,y,\gamma}' = \langle \boldsymbol{u}_i + H(\boldsymbol{u}_i', \gamma), \mathbf{y}_i \rangle + \langle H_i(\gamma), \mathbf{y} \rangle$. Then it outputs $(\mathsf{sk}_{i,y,\gamma}, \mathsf{sk}_{i,y,\gamma}')$.
- $\mathsf{KeyCombin}(\{\mathsf{sk}_{i,y,\gamma}, \mathsf{sk}_{i,y,\gamma}'\}_i)$: it sets $\mathsf{sk}_{y,\ell,\gamma}' = \sum_i \mathsf{sk}_{i,y,\gamma}'$ and outputs $\mathsf{sk}_{y,\gamma} = (\{\mathsf{sk}_{i,y,\gamma}\}_i, \mathsf{sk}_{y,\gamma}')$.
- $\mathsf{Enc}(\mathsf{msk}_i, \mathbf{x}_i, \gamma)$: it runs $\mathsf{ct}_{i,\gamma} \leftarrow \mathsf{MCFE.Enc}(\mathsf{msk}_i', \mathbf{x}_i, \gamma)$ and outputs $\mathsf{ct}_{i,\gamma}$.
- $\mathsf{Dec}((\mathsf{ct}_{i,\gamma})_i, \mathsf{sk}_{y,\gamma})$: it runs $D_\gamma \leftarrow \mathsf{MCFE.Dec}(\{\mathsf{ct}_{i,\gamma}\}_i, (\{\mathsf{sk}_{i,y,\gamma}\}_i, \mathsf{sk}_{y,\gamma}'))$ and outputs the result.

The compiler is identical to the one in [3], apart from the terms that use functions H_i. In the original compiler this functions are replaced with random

[13] So, the only difference with the centralized MCFE is that, KeyGen algorithm receives the master secret key of the client i, rather than all of the secret keys.

[14] Or several concatenation of $\mathbf{y}$

[15] Note that running this algorithm only over the secret key msk_i' can not generate the other parts of the output. We have used $\perp$ to show this fact.

values $\boldsymbol{v}_i$, that are generated in Setup and have the property $\sum_{i\in[\ell]} \boldsymbol{v}_i = 0$. That is, the values are independent of the label. We now explain how functions $\{H_i\}_{i\in[\ell]}$ can be generated based on general assumptions and later prove the security of the above compiler.

A Generalized protocol for Decentralized-Sum In [15] the authors presented a protocol for decentralized sum (DSum). A DSum protocol is a special case of standard MCFE for inner-product where there is only one function $\mathbf{y} = (1, \ldots, 1)$, such that the output of MCFE (for inner-product) is the sum of decentralized data. The DSum protocol of [15] is based on CDH assumption, here we extend their scheme for a more general case where as far as there exists a secure key-exchange protocol (KEX) which generates shared keys $v_{i,j}$ for two parties, DSum protocol is secure. This allows us to get a DSum protocol based on more general assumptions including the lattice-based assumptions.

A key exchange (KEX) protocol is a protocol between two clients such that the final output $v_{i,j} \leftarrow \mathsf{KEX}(1^\kappa, i, j)$ presents the exchanged shared key between the clients i and j. In the security analysis of our DSum scheme, we do not need a precise security notion of KEX. In fact, all we need is that if one can compute the shared key of two parties from their public values, then KEX is not secure. This is a basic condition for the security of any KEX scheme. We will call this condition the *weak security of KEX* (wsKEX).

Let $H' : (\{0,1\}^\kappa, \mathsf{Labels}) \to \mathbb{Z}_q^{\ell k}$ be a hash function, and $t_{i,j} = (\min\{i,j\}, \max\{i,j\})$. Now our generalized DSum protocol would be as follows.

Generalized DSum protocol (GDSum).

– $\mathsf{Setup}(1^\kappa)$: it runs $v_{i,j} \leftarrow \mathsf{KEX}(1^\kappa, i, j)$ for $i < j$. And sets $v_{j,i} = v_{i,j}$. The secret key of user i is $(v_{i,j})_{j\neq i, j\in[\ell]}$.
– $\mathsf{Enc}((v_{i,j})_j, m_i, i, \gamma)$: it outputs

$$\mathsf{ct}_{i,\gamma} = m_i + \sum_{j<i} H'(t_{i,j}, v_{i,j}, \gamma) - \sum_{j>i} H'(t_{i,j}, v_{i,j}, \gamma)$$

– $\mathsf{Dec}((\mathsf{ct}_{i,\gamma})_{i,\gamma})$: it outputs $\sum_i \mathsf{ct}_{i,\gamma}$.

Note that for this DSum protocol (as a special case of MCFE), since there is only one functional key ($\mathsf{sk}_f = \mathbf{0}$), we do not need to mention or simulate it in the scheme or the security proof. The correctness is the same as the DSum protocol of [15].

Theorem 7. *Let the KEX scheme be weakly secure, then our GDSum protocol is statically secure.*

Proof. We proceed through the following sequence of games:

$\mathbf{G}_0$: is the real game associated with the bit $b = 0$. W.l.g we assume the first $h \geq 2$ users are honest[16].

[16] Note that this is always the case about MCFE scheme, that at least there should be two honest users

$\mathbf{G}_1$: is similar to the previous game except that, the shared key $v_{i,j} \leftarrow \mathsf{KEX}(1^\kappa, i, j)$ is replaced with a random value. Here we reduce the indistinguishably of $\mathbf{G}_0$ and $\mathbf{G}_1$ to the weak security of KEX. Let $\mathcal{A}$ be the attacker trying to distinguish two games $\mathbf{G}_0$ and $\mathbf{G}_1$, and $\mathcal{B}$ be the attacker to the weak security of KEX.

Since the only difference between two games is in the value $v_{i,j}$, the adversary $\mathcal{A}$ can distinguish between these two games if it has asked a RO query on the real value $v_{i,j}$. In this case $\mathcal{B}$ would look for a RO query of the form $(t_{i,j}, v_{i,j})$ allowing it to find the right shared key associated with the concerned indices i, j. Finally, it outputs $v_{i,j}$ as its output for wsKEX.
The argument is used for each pair $\{i, j\} \subset [h]$, thus we have

$$|\mathsf{Adv}_{\mathcal{A},\mathbf{G}_0}(\kappa) - \mathsf{Adv}_{\mathcal{A},\mathbf{G}_1}(\kappa)| \leq \frac{d^2}{2} \mathsf{Adv}_{\mathcal{B}}^{\text{wsKEX}}(\kappa).$$

$\mathbf{G}_2$: Let i_γ be the index of the honest client involved in the last query of the adversary with respect to each label γ. The simulator in this game answers $\mathsf{ct}_{i_\gamma,\gamma} = \sum_{j\in[h]} m_j + \sum_{j<i_\gamma} H'(v_{i_\gamma,j}, \gamma) - \sum_{j>i_\gamma} H'(v_{i_\gamma,j}, \gamma)$, and $\mathsf{ct}_{i,\gamma} = \sum_{j<i} H'(v_{i,j}, \gamma) - \sum_{j>i} H'(v_{i,j}, \gamma)$ for every honest $i \neq i_\gamma$. This game can be obtained from the previous one by replacing $H'(v_{i,i_\gamma}, \gamma)$ with $H'(v_{i,i_\gamma}, \gamma) + (-1)^{i>i_\gamma} m_i$ for each honest client $i, i \neq i_\gamma$. Thus we replace uniformly random values with other uniformly random ones. Thus these two games are perfectly indistinguishable.
$\mathbf{G}_3$: Finally the simulator answers $\mathsf{ct}_{i,\gamma} = r_{i,\gamma} - \sum_{j>h} H'(v_{i,j}, \gamma)$ for every honest $i \neq i_\gamma$, where $r_{i,\gamma}$ is uniformly random, and $\mathsf{ct}_{i_\gamma,\gamma} = \sum_{j\in[h]} m_j - \sum_{j\in[h]-\{i_\gamma\}} r_{i,\gamma} - \sum_{j>h} H'(v_{i_\gamma,j}, \gamma)$. Similarly as before, this game can be obtained from the previous one by replacing $H'(v_{i,i_\gamma}, \gamma)$, for each $i \in [h] - \{i_\gamma\}$, with

$$(-1)^{i>i_\gamma}(r_{i,\gamma} - \sum_{j<i, j\neq i_\gamma} H'(v_{i,j}, \gamma) + \sum_{i<j\leq h, j\neq i_\gamma} H'(v_{i,j}, \gamma)),$$

which are distributed as uniformly random values.
$\mathbf{G}_4$: is the real game for $b = 1$. By the condition $\sum_{i\in[h]} m_i^0 = \sum_{i\in[h]} m_i^1$ imposed by the game, one can replace m_i^0 with m_i^1 for $i \in [h]$ in the previous game and take the backward steps similarly from $\mathbf{G}_3$ to $\mathbf{G}_0$, to get the real game associated with $b = 1$. □

Security proof of DMCFE Now we are ready to present the formal security proof of our DMCFE compiler. We define functions $\{H_i\}$ as $H_i(\gamma) = \mathsf{GDSum.Enc}((v_{i,j})_j, 0, i, \gamma)$ i.e., each client encrypts the messange $\mathbf{0}$ in the GDSum and thus we have $\sum_i H_i(\gamma) = \sum_i \mathbf{0} = \mathbf{0}$ as needed.

Theorem 8. *If GDSum and MCFE scheme are statically secure, then our compiled DMCFE scheme is statically secure.*

Proof. The proof proceeds through the following sequence of the games.

$\mathbf{G}_0$: is the real game associated with $b = 0$.
$\mathbf{G}_1$: is similar to the previous game except that, we replace the values $H_i(\gamma)$ with the random values $\boldsymbol{v}_{i,\gamma}$ such that $\sum \boldsymbol{v}_{i,\gamma} = 0$ and $\boldsymbol{v}_{i,\gamma} = H_i(\gamma)$ for the corrupted indices i.
$\mathbf{G}_2$: is similar to the previous game except that we replace the keys $\mathsf{sk}'_{i,y,\gamma}$ with the following values:

$$\mathsf{sk}'_{i,y,\gamma} = \begin{cases} \langle s_{i,\gamma}, \mathbf{y}_i \rangle & i \neq i^* \\ \left(\mathsf{sk}'_{y,\gamma} - \sum_i \langle s_{i,\gamma}, \mathbf{y}_i \rangle\right) & i = i^* \end{cases} \tag{1}$$

where i^* is the last honest user, $s_{i,\gamma} = \boldsymbol{u}_i + H(\boldsymbol{u}'_i, \gamma)$ for the corrupted indices and it is random for uncorrupted indices.
$\mathbf{G}_3$: is the real game for $b = 1$.

Transition from $\mathbf{G}_0$ to $\mathbf{G}_1$: For this transition we rely on the security of GDSum. Let $\mathcal{A}$ be the adversary trying to distinguish these two games and $\mathcal{B}$ be the adversary to the security of GDSum, then $\mathcal{B}$ simulates the games for $\mathcal{A}$ by running the real algorithms except the ones related to the GDSum as follows,

- Setup: the adversary $\mathcal{B}$ forwards the public keys of GDSum to $\mathcal{A}$.
- Corruption queries: when $\mathcal{A}$ sends a corruption query, $\mathcal{B}$ accordingly corrupt the client in its game.
- key generation queries: when $\mathcal{A}$ sends a query $\mathbf{y}$ on (i, γ), the adversary $\mathcal{B}$ sends a challenge encryption-query on $(m_i^0, m_i^1) = (0, r_{i,\gamma})$ where $r_{i,\gamma}$ is chosen randomly for uncorrupted i and $r_{i,\gamma} = 0$ for corrupted indices such that $\sum_{i \in [\ell]} r_{i,\gamma} = 0$. It receives the ciphertext $\mathsf{ct}^b_{i,\gamma}$ from its challenger. Then it simulates $\mathsf{sk}'_{i,y,\gamma}$ as $\langle \boldsymbol{u}_i, H(\boldsymbol{u}'_i, \gamma) \rangle + \langle \mathsf{ct}^b_i, \mathbf{y} \rangle$.
- finalizing: when $\mathcal{A}$ sends a bit b, the adversary $\mathcal{B}$ outputs b.

Clearly, if $\mathcal{A}$ wins its game with probability ϵ, then $\mathcal{B}$ also wins with probability ϵ. Now note that changing message 0 to $r_{i,\gamma}$ as above, is equivalent with changing $H_i(\gamma)$ to $\boldsymbol{v}_{i,\gamma}$ with the same property (mentioned in the game $\mathbf{G}_1$).

Transition from $\mathbf{G}_1$ to $\mathbf{G}_2$: This transition can be done similar to a counterpart game for the compiler of [3], with the only difference where we apply the change, used in their key generation, for each label. More precisely, for the fixed label γ, as the values of $\boldsymbol{v}_{i,\gamma}$ are random (with the mentioned property), then $\langle \boldsymbol{v}_{i,\gamma}, \mathbf{y} \rangle$ for $kn - 1$ linearly independent vectors $\mathbf{y}$ (where kn is the length of $\mathbf{y}$), hides the values $\langle \boldsymbol{u}_i + H(\boldsymbol{u}'_i, \gamma), \mathbf{y}_i \rangle$ (up to a secret-sharing level). This means the values $\mathsf{sk}'_{i,y,\gamma}$ can be seen as the secret sharing of the $\mathsf{sk}'_{y,\gamma} = \sum \mathsf{sk}'_{i,y,\gamma}$. And thus can be replaced with the values in Eq. (1) (for a more detailed discussion we refer the reader to [3]).
Transition form $\mathbf{G}_2$ to $\mathbf{G}_3$: For this transition we rely on the security of MCFE scheme. Let $\mathcal{A}$ be the attacker trying to distinguish between this two games, and $\mathcal{B}$ be the attacker to the security of MCFE. The adversary $\mathcal{B}$ simulates the games for $\mathcal{A}$ by the real algorithms, except for key generation queries. After receiving the functional key $\mathsf{sk}'_{y,\gamma} = \sum_i \langle, \boldsymbol{u}_i + H(u'_i, \gamma) \rangle$ from the challenger of MCFE, it

generates the shares $\mathsf{sk}_{i,y,\gamma}$ as Eq. (1), and sends them to $\mathcal{A}$. Note that corruption and encryption queries are simulated just by relaying them to the challenger of MCFE. Then, taking backward steps similarly from $\mathbf{G}_2$ to $\mathbf{G}_0$ results in the real game associated with $b = 1$ □

E Algorithms for CRT and NTT based multiplication

We describe the algorithms for our CRT and NTT based multiplication. Alg. 1 describes the general Garner's or inverse CRT algorithm. Note that when the q_i's are constant as in our implementation, the calculation of CRT constants C_i's from line 2-5 can be precomputed.

Algorithm 1: Garner's algorithm for Chinese remainder theorem

```
input  : A positive integer $q = \prod_{i=1}^{t} q_i > 1$, with $\texttt{gcd}(q_i, q_j) = 1$ for all $i \neq j$ and
         $v(x) = (v_1, v_2, \cdots, v_t)$ such that $X \equiv v_i \mod q_i$ for all $i$.
output : $x$ such that $X = x \mod q$
1  for $(i = 2; i \leq t; i++)$ do
2      $C_i = 1$;
3      for $j = 1; j \leq i-1; j++$ do
4          $u = q_j^{-1} \mod q_i$;
5          $C_i = u \cdot C_i \mod q_i$;
6      $u = v_1$; $x = u$;
7      for $(i = 2; i \leq t; i++)$ do
8          $u = (v_i - x) \cdot C_i \mod q_i$;
9          $x = x + u \cdot \prod_{j=1}^{i-1} q_j$
10 return $x$
```

Forward and inverse NTT : Algorithm 2 and 3 describe two algorithms we have used in our implementation for forward and reverse NTT transformations.

E.1 Use-case description

One of the main tasks of machine learning (ML) is the classification of data based on their feature vectors describing the instances. FE can allow to use ML functionality while preserving the privacy of the data. In particular, it allows to evaluate ML models on encrypted data, revealing only the end result of the classification.

A well known ML example is a dataset named MNIST, consisting of images of handwritten digits that needs to be recognized. Each image is a 28×28 pixel array, where each pixel is represented by its gray level, and the task is to classify it into one of 10 possible classes (digits). We chose this dataset since it has been

Algorithm 2: Forward NTT transformation using Cooley-Tukey method

```
   input  : A vector $a = (a[0], a[1], \cdots, a[n-1]) \in \mathbb{Z}_n^{q'}$ in standard ordering, where
            $q'$ is a prime such that $q \equiv 1 \mod 2n$ and $n$ is a power of two. A
            precomputed table $\psi_{rev} \in \mathbb{Z}_{q'}^n$ storing powers of $\psi$ in a bit-reversed order
   output : $a \leftarrow$ NTT($a$)
 1 $t = n$;
 2 for ($m = 1; m < n; m = 2m$) do
 3     $t = t/2$;
 4     for ($i = 0; i < m; i++$) do
 5         $j_1 = 2 \cdot i \cdot t$;
 6         $j_2 = j_1 + t - 1$;
 7         $S = \psi_{rev}[m + i]$;
 8         for ($j = j_1; j \leq j_2; j++$) do
 9             $U = a[j]$;
10             $V = a[j + t] \cdot S$;
11             $a[j] = U + V \mod q'$;
12             $a[j + t] = U - V \mod q'$;
13 return $a$
```

Algorithm 3: Inverse NTT transformation using Gentleman-Sade method

```
   input  : A vector $a = (a[0], a[1], \cdots, a[n-1]) \in \mathbb{Z}_n^{q'}$ in bit-reversed ordering,
            where $q'$ is a prime such that $q \equiv 1 \mod 2n$ and $n$ is a power of two. A
            precomputed table $\psi_{rev}^{-1} \in \mathbb{Z}_{q'}^n$ storing powers of $\psi^{-1}$ in bit-reversed
            order
   output : $a \leftarrow$ INTT($a$)
 1 $t = 1$;
 2 for ($m = n; m > 1; m = m/2$) do
 3     $j_1 = 0$;
 4     $h = m/2$;
 5     for ($i = 0; i < h; i++$) do
 6         $j_2 = j_1 + t - 1$;
 7         $S = \psi_{rev}^{-1}[h + i]$;
 8         for ($j = j_1; j \leq j_2; j++$) do
 9             $U = a[j]$;
10             $V = a[j + t] \cdot S$;
11             $a[j] = U + V \mod q'$;
12             $a[j + t] = (U - V) \cdot S \mod q'$;
13         $j_1 = j_1 + 2t$;
14     $t = 2t$;
15 for ($j = 0; j < n; j++$) do
16     $a[j] = a[j] \cdot n^{-1} \mod q$;
17 return $a$
```

used before in the context of FE [17, 30]. While state of the art ML models can provide even 99% accuracy on this task, FE allows computing only limited functions. In our case the prediction has to be done through linear functions which is known as the logistic regression. Such a model can achieve up to 92% accuracy.

To be more precise, 10 linear functions, whose coefficients need to be learned in advance, are evaluated on $\ell = 785$ dimensional vectors (one dimension is added for the bias of the model). Each function is indicating how likely the corresponding digit is in the image. Since the inputs have to be integers, the data and the model have to have discrete values and not floats. Having inputs of the vectors (grayscale) from interval $[0, 4]$ (bound $B_x = 4$) and coefficients from $[0, 16]$ (bound $B_y = 16$) suffices that the accuracy of the model does not significantly change.