PELTA – Shielding Multiparty-FHE against Malicious Adversaries

Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux

EPFL, Lausanne, Switzerland
{sylvain.chatel}@epfl.ch

Abstract. Multiparty fully homomorphic encryption (MFHE) schemes enable multiple parties to efficiently compute functions on their sensitive data while retaining confidentiality. However, existing MFHE schemes guarantee data confidentiality and the correctness of the computation result *only against honest-but-curious adversaries*. In this work, we provide the first practical construction that enables the verification of MFHE operations in zero-knowledge, protecting MFHE from malicious adversaries. Our solution relies on a combination of lattice-based commitment schemes and proof systems which we adapt to support both modern FHE schemes and their implementation optimizations. We implement our construction in Pelta. Our experimental evaluation shows that Pelta is one to two orders of magnitude faster than existing techniques in the literature.

1 Introduction

Multiparty Fully Homomorphic Encryption (MFHE) schemes [LATV12; Che+19; Ash+12; Bon+18; Par21; Kwa+21; AH19; Mou+21; MBH22] enable multiple parties to homomorphically compute joint functions, while ensuring that the decryption of the underlying data and results can only be performed collectively. MFHE offers a more flexible and efficient alternative to classic multiparty computation (MPC) protocols and (single-party) FHE, and it has been successfully employed for distributed training of machine-learning models [Che+19; Fer+21; ATP21; Fro+21a; Sav+21], medical analytics [Fro+21b; Sav+22; Che+22; Cho+22a], and financial audits [Yua+22]. However, similar to FHE, MFHE schemes are *only* secure against honest-but-curious adversaries that follow the protocol specification; this opens up new avenues for malicious parties to disrupt secure computation pipelines involving high-value data. Indeed, in MFHE, a single malicious party can compromise the correctness of the computation by generating improper keys that lead to invalid outputs (*e.g.,* decryption of *garbage*) or it can bias the decryption result by excluding the contributions of other parties. Furthermore, collusions among malicious actors and incorrect homomorphic evaluation can hinder the confidentiality of honest parties [VKH23; CT14; CGG16]. While these limitations have been known for a decade [Ash+12; MW16], the existing literature has focused mostly on the passive adversary model [LATV12;

Che+19; Par21; Mou+21; MBH22] and any proposals accounting for malicious adversaries have remained theoretical [Ash+12; MW16].

A natural approach to secure MFHE pipelines against malicious adversaries is to employ techniques based on non-interactive zero-knowledge (NIZK) proofs [Ash+12]. However, concretely instantiating a NIZK-based solution for maliciously-secure MFHE pipelines requires overcoming considerable challenges. First, the workflow of MFHE schemes involves verifying a number of constraints ranging from linear relations between polynomials with small and large coefficient bounds, to consistency of secrets across different protocols. Such verification requires tailored constructions that can not be supported by prior work without hindering the capabilities of the underlying FHE scheme to fit the constraint of the NIZK proof [Bos+20; DPLS19; BLS19; ENS20; GNSV21]. Second, the high dimensionality of the underlying polynomial structures used in modern FHE schemes (*e.g.,* BFV [FV12; HPS19], BGV [BGV14; KPZ21], and CKKS [Che+17]) makes interfacing MFHE with most NIZK proofs prohibitively expensive. In particular, as in MFHE multiple parties execute a series of online protocols, the NIZK proof needs to have short runtimes. Additionally, implementation-specific optimizations commonly employed to make FHE schemes practical, *e.g.,* non-prime specific moduli [Baj+17] and NTT-transformations [AM+16; PG12; Göt+12], drastically limit the set of NIZK proofs practically compatible with MFHE.

Contributions. In this work, we address these challenges and we provide the first practical construction that tolerates malicious adversaries in MFHE pipelines. We first systematize the MFHE variants from a security perspective and show that proving their correct execution under the malicious threat model involves verifying (a) the appropriate sampling from specific distributions (*e.g.,* ternary or Gaussian), (b) the accurate generation of cryptographic keys, and (c) the correct combination of each party's cryptographic material. Then, to verify the correctness of these MFHE operations, we propose a *commit-then-prove* approach, where each party commits to its execution and proves its correctness in zero-knowledge. To achieve compatibility between our NIZK approach and both the underlying structure and security assumptions of modern FHE schemes, we instantiate our solution using efficient lattice-based commitments [ALS20] and proof systems [ENS20; LNS20], and we design MFHE operation-specific *statements* that account for the theoretical constraints and the implementation optimizations of MFHE. We use an existing MFHE library [EPF21] and we implement our construction which we dub Pelta.[1] We experimentally evaluate Pelta and show that it induces little overhead (just a few seconds) for the MFHE parties and acceptable proof sizes (in the order of MB). To show the superior performance of our approach, we compare it to prior techniques based on malicious MPC [CP16] and proof systems [Bos+20], and show that Pelta achieves one to two orders of magnitude faster prover runtimes with 15 times smaller setup time. Our code is provided for review purposes and it will be made open source with the final version of this paper.

[1] A shield protecting MFHE pipelines against malicious adversaries.

Organization. The remainder of this paper is organised as follows. In §2, we provide some background on multiparty FHE (MFHE). In §3, we provide a systematization of the MFHE variants and the overview of our approach. In §4 and §5, we present our technical contribution for verifying the correctness of the different operations in an MFHE pipeline. We introduce our implementation and experimental evaluation in §6. Finally, we discuss how to secure MFHE pipelines against malicious adversaries end-to-end in §7 and the related work in §8, before concluding in §9.

2 Preliminaries

In this section, we introduce the necessary components of FHE and its multiparty variants.

2.1 Mathematical Notations

Let $\mathcal{R}$ and $\mathcal{R}_q$ be the rings $\mathbb{Z}[X]/(X^N+1)$ and $\mathbb{Z}_q[X]/(X^N+1)$, resp., with N a power-of-two. A boldface letter $\mathbf{p}$ denotes a polynomial element in $\mathcal{R}_q$ (whose coefficients can also be seen as a vector in $\mathbb{Z}_q$). An upper arrow $\vec{\mathbf{b}}$ (resp. $\vec{b}$) denotes a vector of polynomials (resp. of scalars). Let $|S|$ be the number of elements in the set S.

Norms and Sizes. The ℓ_1 norm of an element $w \in \mathbb{Z}_q$ is defined as $|w| = |w \bmod q|$. Similarly, ℓ_∞ of an element $\mathbf{w} \in \mathcal{R}_q$ is defined as $\|\mathbf{w}\|_\infty = \max_i |\mathbf{w}[i]|$.

Matrices and Vectors. Let $\mathbf{Id}$ be the identity matrix of size $N \times N$. We denote by $\mathrm{diag}(\vec{p})$ the diagonal matrix made from the coefficients in $\vec{p}$. We denote by $\langle \mathbf{a}, \mathbf{b} \rangle$ the inner product between two vectors $\mathbf{a}$ and $\mathbf{b}$.

2.2 Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE) is a specific type of encryption that enables operations directly on ciphertexts. The most versatile FHE schemes are based on lattice assumptions. For instance, BFV [FV12; HPS19], BGV [BGV14; KPZ21], and CKKS [Che+17], are FHE schemes whose security is based on the hardness of the ring-learning-with-errors (RLWE) problem; these schemes can support additions and a bounded number of multiplications. *Bootstrapping* [Kim+21], is an additional operation that enables to refresh a ciphertext and to evaluate an unbounded number of multiplications. Several libraries implement such RLWE schemes [IBM21; Sea; EPF21; PRR17].

In practice, RLWE schemes work over a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N+1)$ where N is a power-of-two. For efficiency, q is selected to support the number theoretic transform (NTT) that can be seen as a Fourier transform for polynomials. This enables efficient polynomial multiplication operations.

Here, we describe the BFV scheme [FV12]; we use its construction throughout the paper to describe MFHE pipelines and our approach to securing them

against malicious adversaries. However, we note that our approach is also applicable to other RLWE schemes such as BGV [BGV14; KPZ21] or CKKS that share the same arithmetic structure as BFV. The BFV plaintext space is R_t with t an NTT-friendly prime in order to enable batching. Batching is a critical FHE feature, as it enables homomorphic operations over plaintext vectors in $\mathbb{Z}_t$, hence, amortizing FHE costs. BFV requires sampling secrets from two polynomial distributions: the key distribution χ_k, *i.e.,* a ternary distribution with values in the set $\{-1, 0, 1\}$, and the error distribution χ_{err}, *i.e.,* a discretized, centered, and bounded Gaussian distribution of small variance. We now describe the key procedures of the BFV scheme:

BFV.KeyGen$(1^\lambda) \rightarrow \mathbf{sk}, \mathbf{pk}, \mathbf{evk}$: Sample a ternary secret key $\mathbf{s} \leftarrow \chi_k$ from the key distribution and set $\mathbf{sk} := (1, \mathbf{s}) \in \mathcal{R}_q^2$. Compute the public key as $\mathbf{pk} := (\mathbf{b}, \mathbf{a})$, where $\mathbf{a}$ is sampled uniformly at random from $\mathcal{R}_q$ and $\mathbf{b} := [-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e})]_q \in \mathcal{R}_q$ with $\mathbf{e} \leftarrow \chi_{\text{err}}$ sampled from the error distribution. Similarly, generate additional evaluation keys $\mathbf{evk}$, *i.e.,* relinearization and rotation keys (see [FV12]). Output $\mathbf{sk}$, $\mathbf{pk}$, and $\mathbf{evk}$.

BFV.Enc$(\mathbf{m}; \mathbf{pk}) \rightarrow \mathbf{c}$: For a plaintext message $\mathbf{m} \in \mathcal{R}_t$, sample $\mathbf{u} \leftarrow \chi_k$ and $\mathbf{e}_0', \mathbf{e}_1' \leftarrow \chi_{\text{err}}$. Output $\mathbf{c} := [\mathbf{u} \cdot \mathbf{pk} + (\mathbf{e}_0' + \Delta \cdot \mathbf{m}, \mathbf{e}_1')]_q$ where $\Delta = \lceil q/t \rfloor$.

BFV.Dec$(\mathbf{c}; \mathbf{sk}) \rightarrow \mathbf{m}$: Output $\mathbf{m} := [\lceil t/q \cdot [\langle\!\langle \mathbf{sk}, \mathbf{c} \rangle\!\rangle_q \rfloor]_t$.

Ciphertexts are composed of two polynomials. During homomorphic evaluation, any linear operations are executed element-wise on each pair of polynomials representing the input ciphertexts. The multiplication, however, performs a tensor product between the input polynomials. As such, it returns, not a pair, but a triplet of polynomials. A relinearization technique transforms the ciphertext back to only two polynomials [FV12]. This step requires arithmetic outside of $\mathcal{R}_q$. More than linear and multiplicative operations, full arithmetic in $\mathbb{Z}_t$ is possible with rotation operations that shift the components of the plaintext vector using the rotation keys. Additionally, a key-switching operation modifies the key under which a ciphertext is encrypted and a bootstrapping operation can *refresh* the ciphertext to support more operations. These operations are executed outside of the ring $\mathcal{R}_q$, *i.e.,* on larger rings or even in $\mathbb{Z}$ directly.

2.3 Multiparty FHE

In concrete secure computation scenarios, the input data is not held by a single entity but it is, instead, distributed among multiple stakeholders (*e.g.,* in medical [Jag+17; Rai+18], financial [Bog+09; BTW12; Pol+23], and law enforcement [Bog+15] application domains). Multiparty FHE (MFHE) enables the evaluation of functions over encrypted data in these scenarios, while enforcing *joint* cryptographic access-control over the underlying data. As such, MFHE enables secure multiparty computation (MPC) through a simple protocol: First, the parties encrypt their sensitive input data with the MFHE scheme, and the function is homomorphically evaluated over the ciphertexts, either by the parties themselves or by an external evaluator. Then, the parties obtain the computation output by

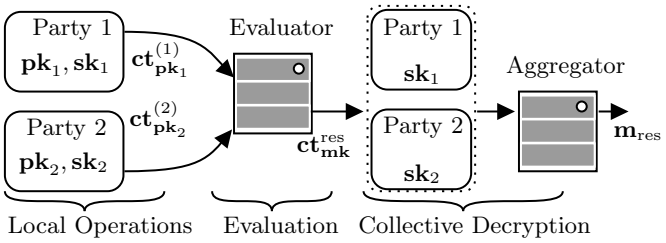

Fig. 1: An illustration of an MkHE pipeline.

engaging in a multiparty decryption protocol. Modern MFHE schemes are based on the ring-learning-with-errors (RLWE) problem [Che+19; Mou+21; Par21; Kwa+21; AH19], and recent literature has employed such schemes to build practical systems for, *e.g.,* distributed analytics [Fro+21b; Cho+22b; Yan+22], and federated machine-learning [Che+19; Sav+22; Sav+21; ATP21; Xu+22; Xu+23; Alo+19].

MFHE schemes can be divided into three families that mainly differ in whether the set of parties is pre-determined before the computation begins, or if parties can join the computation *on-the-fly*. We briefly introduce these families below, and defer the technical details to Section 3.2.

Multi-key FHE (MkHE). In multi-key FHE [LATV12; CCS19; Che+19; CZW17; PS16; BP16], each party encrypts its data with its own locally generated secret key, and each gate in the homomorphic circuit outputs a ciphertext that is encrypted under the concatenation of the parties' secret keys. As a result, the access-control of the intermediate computation values is updated *on-the-fly* with new secret-keys. The final result decryption requires the collaboration among all parties that provided an input to the circuit. An illustration of an MkHE pipeline is presented in Figure 1. MkHE schemes are highly flexible as they do not require to pre-determine the set of participants before the computation can begin; collaboration is only required for the decryption of the final result. However, all current constructions have non-compact ciphertexts (*i.e.,* that grow at best linearly with the number of parties) and induce a significant overhead compared to single-party FHE schemes [Li+19; Yas+18].

Threshold FHE (ThHE). Intuitively, parties in a ThHE scheme emulate a plain, single-party FHE scheme for which the secret key is secret-shared among them [Ash+12; Bon+18; Mou+21; Par21]. More specifically, the parties first generate, by means of a multiparty protocol, a collective encryption key. Then, they encrypt their private inputs under this collective key, and the homomorphic circuit evaluation preserves this secret-key structure throughout the computation. Finally, the parties obtain the computation result by executing a decryption protocol (according to the secret-sharing scheme). A visual illustration of a ThHE pipeline is shown in Figure 2. ThHE schemes require defining the set of parties before the computation can begin, hence they are less flexible than their MkHE counterparts. However, current ThHE constructions are compact, and the complexity of their homomorphic evaluation is independent of the number

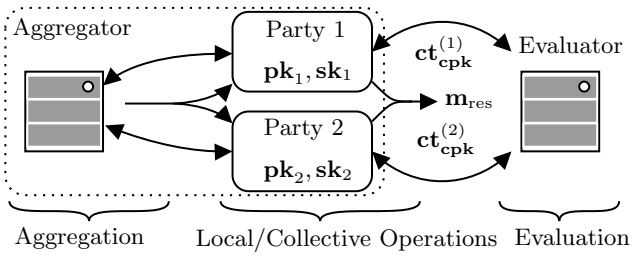

Fig. 2: An illustration of a ThHE pipeline.

of parties, making them a much more practical option for settings with a fixed set of participants.

Multi-group FHE (MgHE). Multi-group FHE [Par21; Kwa+21; AH19] is a hybrid construction with a fixed set (a *group*) of parties acting as a single party (*i.e.,* by means of a ThHE scheme) within an MkHE scheme. In this way, any parts of the homomorphic circuit that only require input from the fixed, pre-defined set of parties can be evaluated under the more efficient ThHE homomorphic arithmetic, while leaving the possibility to evaluate parts of the circuit *on-the-fly* using MkHE. For example, a machine learning model can be trained, under ThHE, among a fixed set of training data holders, then queried, under MkHE, by external parties.

3 Towards Malicious Multiparty FHE

In this section, we present the system and threat models considered in our work (§3.1). Then, we propose the first security-oriented systematization of the MFHE families (§3.2). This enables us to identify core MFHE operations that can be exploited by malicious adversaries and hence, need to be protected. Finally, we present the roadmap of our solution to achieve this goal (§3.3).

3.1 System and Threat Model

System Model. We consider a set of *parties* $\mathcal{P}$ that seek to engage in a joint computation by using multiparty homomorphic encryption [LATV12; CZW17; Ash+12; Bon+18; Mou+21], and evaluate a function $f(\cdot)$ on sensitive data. To facilitate the protocol execution, an *aggregator* combines cryptographic material, and an *evaluator* executes the homomorphic circuit on the ciphertexts (Figs. 1 and 2). These two roles can be played by any of the parties. The parties are interconnected via authenticated channels and are available during the protocol execution (see Figures 1 and 2 for interaction illustrations).

Threat Model. We consider that the parties are in an *anytrust* model [Wol+12], *i.e.,* up to all but one of them can behave maliciously and collude to break the correctness or the confidentiality of the protocol execution. We do not consider denial-of-service attacks, *i.e.,* the parties do not refuse to participate in the protocol execution. However, they can attempt to force the secure computation

protocol to output an invalid result. For the sake of abstraction, we consider a *verifier* whose task is to check the correct execution of the multiparty protocol and to abort if inconsistencies are detected. We note that this virtual entity's role can be executed by, at least, the honest party in the system.

3.2 Systematizing Multiparty FHE

To secure MFHE pipelines in the presence of malicious adversaries (§3.1), we need to identify the MFHE operations that can be tampered with and whose correctness needs to be verified. To this end, we propose a systematization of the MFHE families presented in §2.3, under a unified model. We observe that the currently implemented RLWE-based MkHE, ThHE, and MgHE schemes [Che+19; Mou+21; Par21; Kwa+21; AH19], share common functionalities in their constructions. We systematize these functionalities by identifying their common denominators; this enables us to define the building blocks of a generic method for verifying their correct execution in the presence of malicious parties. We also highlight how the various constructions of MFHE schemes differ and show that, from the correctness verification perspective, these differences only affect a single *aggregation* functionality.

MFHE schemes can be expressed in terms of their basic operations: (SecKeyGen, EncKeyGen, EvalKeyGen, Enc, Eval, Dec). In SecKeyGen, each party $\mathcal{P}_i \in \mathcal{P}$ for $i \in [1, |\mathcal{P}|]$ generates its local secret-key $\mathbf{s}_i$ for the MFHE scheme. In EncKeyGen and EvalKeyGen, the parties use their secret-keys to generate the public key material required by the Enc and Eval operations, respectively. Enc performs the encryption of the input data into ciphertexts, and Eval evaluates a homomorphic circuit on these ciphertexts. Finally, Dec decrypts MFHE ciphertexts. These MFHE basic operations can be organized into two types: *interactive* and *non-interactive* ones. Operations that require secret inputs from several parties are implemented as secure *interactive protocols*; their output should be computed while preserving the confidentiality of these secrets. EvalKeyGen and Dec are such interactive operations. Interestingly, MFHE interactive protocols have a very similar structure across the different families, which we discuss in §3.2.1. Non-interactive MFHE operations, however, do not require secret inputs from multiple parties, hence they can be executed locally by each party without any interaction. SecKeyGen, Enc, and Eval are non-interactive operations and we discuss such operations in §3.2.2. We note that EncKeyGen is a non-interactive operation in MkHE as each party generates its own encryption key, while it is an interactive one in ThHE where the parties generate a single collective encryption key.

3.2.1 Interactive Operations

These operations are at the core of MFHE schemes because they enable secret-key functionalities through interaction among the parties; verifying their correct execution is the main target of our work. For all MFHE families (§2.3), these operations are single-round protocols which unfold in two steps:

Step 1 - Local Share Generation: In this step, each party uses its local secret key (as well as additional freshly sampled secrets and error polynomials) to locally generate a share that can be publicly disclosed. We first observe that these shares have a common structure across interactive protocols and MFHE families. In particular, a share $\mathbf{b}_i$ is computed as a linear equation in $\mathcal{R}_q$ of the form:

$$\mathbf{b}_i = \mathbf{a} \cdot \mathbf{s}_i + \mathbf{X} + \mathbf{e}_i \tag{1}$$

where $\mathbf{a}$ is a publicly known polynomial, $\mathbf{s}_i$ is the secret-key of the party $\mathcal{P}_i \in \mathcal{P}$, $\mathbf{e}_i \leftarrow \chi$ is a freshly sampled error term from some distribution χ, and $\mathbf{X} \in \mathcal{R}_q$ is a *polynomial placeholder* that takes different forms in the various interactive protocols.

For EncKeyGen and EvalKeyGen, $\mathbf{a}$ is sampled from a common random string (CRS) and $\chi{=}\chi_{\text{err}}$ is the short-norm RLWE error distribution (for both ThHE and MkHE). $\mathbf{X}$ is zero for EncKeyGen, and it is a function of the secret-key (*i.e.,* its decomposition in some small-norm vector basis) for EvalKeyGen. For Dec, $\mathbf{a}$ is an element in $\mathcal{R}_q$ from the ciphertext being decrypted. In particular, for ThHE schemes, where a ciphertext $\vec{\mathbf{c}}$ encrypting $\mathbf{m}$ has the form $\vec{\mathbf{c}}{=}(\mathbf{c}_0, \mathbf{c}_1)$ such that $\text{Dec}(\mathbf{s}_1, \ldots, \mathbf{s}_{|\mathcal{P}|}, \vec{\mathbf{c}}){=}\mathbf{c}_0 + \mathbf{c}_1 \sum_{i=1}^{|\mathcal{P}|} \mathbf{s}_i{=}\mathbf{m}$, $\mathbf{a}$ is the ciphertext element $\mathbf{c}_1$. For MkHE schemes, where $\vec{\mathbf{c}}{=}(\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathcal{P}|})$ such that $\text{Dec}(\mathbf{s}_1, \ldots, \mathbf{s}_{|\mathcal{P}|}, \vec{\mathbf{c}}){=}\langle(1, \mathbf{s}_1, \ldots, \mathbf{s}_{|\mathcal{P}|}), \vec{\mathbf{c}}\rangle{=}\mathbf{m}$, $\mathbf{a}$ is the element $\mathbf{c}_i$ for the share of party $\mathcal{P}_i$. For both ThHE and MkHE families, the error term $\mathbf{e}_i$ in Dec is sampled from a higher norm distribution χ_{smdg} to ensure circuit privacy according to the *smudging* technique [Ash+12].

An erroneous execution of the local share generation by malicious parties in any MFHE protocol would lead to an incorrect decryption at the end of the computation, hence verifying the correct execution of this protocol step is crucial for MFHE pipelines. However, as the local share generation involves the parties secret keys, its correct execution needs to be verified in zero-knowledge. This implies checking that: (i) the secret-key and error polynomials are sampled from specific distributions (*e.g.,* ternary or Gaussian distributions of variable bounds), (ii) $\mathbf{X}$ and the linear relation in Eq. (1) are computed correctly, and (iii) each party uses the same secret-key across different MFHE interactive protocols.

Step 2 - Share Aggregation: After the parties complete their local share generation comes the interactive part of the protocol. Each party sends its share $\mathbf{b}_i$ to the aggregator, which aggregates the shares into a *collective value* from which the final protocol output can be computed. The aggregation function depends on the MFHE operation as well as on the scheme family: In MkHE schemes, during EvalKeyGen, the aggregator computes the public evaluation-key by simply concatenating the parties' shares. In ThHE schemes, the aggregator computes the public encryption and evaluation keys (*i.e.,* EncKeyGen and EvalKeyGen), by computing the sum of the parties' shares in $\mathcal{R}_q$. For both scheme families, the sum is used to combine the parties' shares during Dec.

Ensuring the correctness of the share aggregation operation is crucial because, without this capability, the confidentiality or the utility of the MFHE scheme can be compromised by malicious parties. For instance, during the collective public-

key generation (EncKeyGen) in ThHE schemes, a cheating aggregator could collude with malicious parties and discard the shares of honest parties. This way, the latter could be tricked into encrypting their private data with a public key for which they do not hold a secret key share, hence allowing the colluding parties to decrypt them. Furthermore, during the collective decryption (Dec) in both MkHE and ThHE schemes, a malicious aggregator could purposefully omit a specific party's polynomial in the merging operation leading to an invalid output.

Other Operations. ThHE schemes support additional interactive operations that enable functionalities such as key switching (KeySwitch) where the parties change the key under which a ciphertext is encrypted and collective bootstrapping (CollBootstrap) in which the parties refresh a ciphertext [Mou+21]. These operations are special cases of Equation 1 and they can be expressed by setting $\mathbf{X}$ accordingly. As such, they can also be decomposed following **Step 1** and **2**, described above.

3.2.2 Non-interactive Operations

MFHE non-interactive operations are executed by each party without interaction; SecKeyGen, Enc, and Eval are such examples. For all MFHE schemes, in SecKeyGen, each party generates its secret-key by sampling the key distribution χ_k. As seen in §2.2, χ_k is a polynomial distribution with ternary coefficients. Ensuring ternarity of the coefficients is paramount for the correctness of the MFHE scheme and needs to be checked in zero-knowledge to avoid any confidentiality issues. Moreover, as the secret-key is used during the local-share generation of interactive MFHE operations, its consistency throughout these needs to be ensured. The encryption operation (Enc) of MFHE schemes is the standard FHE encryption (see §2.2), *i.e.,* it is very similar to Eq.1. In MkHE schemes, each party encrypts using its own key, and in ThHE, each party encrypts using the collective public key generated by EncKeyGen. Ensuring correct encryption implies checking the correctness of the input data; this is a problem already under consideration in the literature (under standard MPC models; see §7.2) [ENS20; LN17; Bos+20; DPLS19]. Finally, during the computation phase (Eval), the evaluator homomorphically executes the public circuit on the encrypted data. Depending on the setting, this evaluation is performed over single-key ciphertexts (ThHE) or over extended ciphertexts with multiple keys (MkHE and hybrid). Wrongful operations can lead to erroneous results or even to the leakage of the parties' secret keys, as a malicious evaluator can return a tailored ciphertext on which performing partial decryption can leak the party's secret key [CT14]. Recent works have designed promising solutions to verify the correctness of homomorphic evaluation [GNSV21; Cha+22; FGP14; Boi+21; FNP20; Nat+21] (see §7.1). As these are orthogonal problems receiving independent interest, in this work, we do not focus on the correctness of Enc and Eval non-interactive operations.

3.3 Roadmap of our Solution

To secure MFHE pipelines in the presence of malicious adversaries, we propose a novel construction that enables the correctness of critical operations executed by the parties during the various MFHE phases to be proved. We focus, in particular, on **interactive operations** (§3.2.1), and we ensure that all parties and the aggregator perform the correct actions. Recall that MFHE interactive operations are executed in two steps: **Local Share Generation** and **Share Aggregation**. The local share generation involves the creation of secret cryptographic material crucial for the confidentiality and integrity of the MFHE pipeline. Therefore, to verify the correct creation of the shares, we use zero-knowledge proofs that ensure the validity of each message generated by the parties, as proposed and proved secure against malicious adversaries by Asharov *et al.* [Ash+12] but never instantiated. We employ a *commit-then-prove* approach, where each party commits to its local share and proves its correctness in zero-knowledge to the verifier. We enable such proofs by designing statements (or *relations*), the validity of which can be proven using proof systems. To account for the characteristics of RLWE-based FHE schemes and their implementation optimizations (see §2.3), we tailor our statements for efficiency and employ lattice-based commitments and proof systems [ENS20; LNS20]. Informally, parties commit to their different secrets, and a Σ-protocol is executed to ensure that these secrets validate the corresponding MFHE statements. The details are described in §4. To verify the correctness of the share aggregation, we design a novel verifiable aggregation protocol based on the polynomial identity lemma [Sch80]. This enables us to extend our statements and to prove, in one shot, both the parties' correct local operations and aggregation on committed values. This protocol is discussed in §5.

4 Verification of MFHE Local Share Generation

As discussed in §2.3 and §3.2, at the core of MFHE interactive operations is the local share generation performed by the parties (*i.e.,* **Step 1** in §3.2.1). This is crucial for the generation of each party's individual keys and for the MFHE functionalities such as collective key generation, decryption, key-switching, and bootstrapping. Malicious parties can tamper with their local share generation and can create improperly formatted keys that can lead to (i) erroneous decryption, (ii) key leakages, and (iii) broken confidentiality (see §3.2). In this section, we present the details of our *commit-then-prove* approach that forces parties to create publicly verifiable proofs that attest to the correct execution of their local share generation. We first discuss the technical challenges associated with verifying this operation (§4.1) and provide background information on lattice-based commitments (§4.2). Then, we present our techniques for assembling, in a practical manner, the statement required to verify the correct generation of RLWE samples (§4.3) and the ways this can be extended to various MFHE operations, *e.g.,* local key generation, collective decryption, public-key switching, and bootstrapping (§4.4). In this section, we take the viewpoint of a single party as the

prover. Hence, for the sake of notation, we omit the subscript i for the party's secret inputs to Eq. 1.

4.1 Challenges of Verifying MFHE Local Share Generation

Whereas several works, *e.g.,* [DPLS19; Bos+20; BLS19; ENS20; LNP22; Lib+18], propose solutions to verify variants of Eq. 1 by using proof systems such as Bulletproofs [Bün+18] and Aurora [BS+19], or commitments and proofs [Bau+18b], they do not consider their application to (M)FHE pipelines. Indeed, modern MFHE implementations introduce challenges that none of the prior works have addressed:

1. **Residue Number System (RNS):** For efficiency reasons, current implementations set the modulus $q = \prod_{j=0}^{L} q_j$ such that the ring splits as $\mathcal{R}_q = \mathcal{R}_{q_0} \times \ldots \times \mathcal{R}_{q_L}$, and perform the ring arithmetic in the decomposed domain. In the proofs, this requires the secret elements to be linked across the sub-rings.
2. **Sub-ring Compatibility:** As the secret $\mathbf{s}$ and the noise $\mathbf{e}$ need to have the same representation over the different sub-rings $\mathcal{R}_{q_j}$, committing to them requires the usage of a sub-ring agnostic commitment scheme.
3. **Number Theoretic Transform (NTT):** As the moduli $\{q_j\}_{j=0}^{L}$ are NTT friendly, *i.e.,* as the polynomial X^N+1 splits into N linear terms modulo each q_j, the proof system must be compatible with this specific composite modulus q (which is not prime, hence, practically incompatible with most generic proof systems [Bün+18; BS+19; Par+13]).
4. **Infinity Norm:** As the secret and noise bounds need to be exactly evaluated in infinity norm, the proofs cannot rely on approximate results or optimised ℓ_2 variants, as in prior work [LNP22].
5. **Large Polynomial Noises:** The noise polynomials are sampled from a Gaussian distribution with bounded infinity norm. Depending on the type of noise, this bound can be large and not easily handled by proof systems, *e.g.,* [BS+19; ENS20].
6. **Secret Consistency Across Protocols:** To guarantee the correctness of the MFHE pipeline, the same secret key(s) should be reused across different interactive operations. Consequently, the secret commitments should be persistent across several protocol executions.

Furthermore, generic proof systems, *e.g.,* [Bün+18; BS+19; Par+13], suffer from long runtime and memory requirements, are incompatible with the arithmetic structure of FHE, and rely on security assumptions different than lattice-based problems (*e.g.,* discrete logarithm and Reed-Solomon codes). To address these issues and the aforementioned challenges, in this work, we use lattice-based commitments and proofs that preserve the post-quantum security of the underlying FHE scheme. We do so as such schemes share the same arithmetic structure as FHE, and we employ recent improvements and optimizations that have made them more efficient than generic constructions for structured relations [ENS20]

to verify the correctness of statements, such as Eq. 1, in the scope of MFHE. Before describing in depth our tailored statements, we provide background information on lattice-based commitment schemes and the types of proofs that can be realized with their constructions.

4.2 Background: Lattice-Based Commitments

Lattice-based commitments, *e.g.,* [Ajt96; KTX08; Bau+18b], rely on the hardness of lattice-based problems such as the module short integer solution (MSIS) and the module learning with error (MLWE) [LS15; Duc+18] to securely commit to polynomial values. For instance, the scheme by Ajtai [Ajt96] can be used to commit to a small norm vector with a commitment size independent of the input's dimension. Committing to vectors of larger norm requires the use of BDLOP [Bau+18b] at the cost of the commitment's linear growth with the dimension of the input vector. Recently, various improvements, *e.g.,* new sampling techniques, the support for integer relations, NTT-friendly rings and NTT operations, have boosted the practicality of such constructions [ALS20; Esg+19a; LNS20; LNS21a; LNP22]. Built on such commitments, proof systems can be used to prove knowledge of committed secrets satisfying specific statements (called *relations*).

We recall a variant of the BDLOP [Bau+18b] lattice-based commitment scheme. Suppose the module ranks κ and λ ensure MSIS and MLWE security [LNP22] and that we want to commit to a message vector of ℓ polynomials $\vec{\mathbf{m}} = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)^T$. The commitment scheme works as follows:

Com.KeyGen$(\kappa, \lambda, \ell) \rightarrow \mathbf{B}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_\ell$: Sample a uniformly random matrix $\mathbf{B}_0 \xleftarrow{\$} \mathcal{R}_q^{\kappa \times (\lambda+\kappa+\ell)}$ and vectors $\vec{\mathbf{b}}_1, \ldots, \vec{\mathbf{b}}_\ell \xleftarrow{\$} \mathcal{R}_q^{\lambda+\kappa+\ell}$. Output them as public parameters.

Com.Commit$(\vec{\mathbf{m}}) \rightarrow \vec{\mathbf{t}}$: First, sample a random vector $\vec{\mathbf{r}} \xleftarrow{\$} \chi^{(\kappa+\lambda+\ell)}$ with χ the polynomial ternary distribution with coefficients in $\{-1, 0, 1\}$ such that $\Pr(0) = 6/16$ and $\Pr(-1) = \Pr(1) = 5/16$. Then, compute:

$$\vec{\mathbf{t}}_0 = \mathbf{B}_0 \vec{\mathbf{r}},$$
$$\mathbf{t}_j = \langle \vec{\mathbf{b}}_j, \vec{\mathbf{r}} \rangle + \vec{\mathbf{m}}_j \text{ for } j = 1, \ldots, \ell.$$

In this scheme, $\vec{\mathbf{t}_0}$ acts as the binding part and each polynomial $\mathbf{t}_j$ encodes one message $\mathbf{m}_j$. The commitment is the vector of polynomials $\vec{\mathbf{t}} = (\vec{\mathbf{t}_0}, \mathbf{t}_1, \ldots, \mathbf{t}_\ell)$. By construction, the commitment scheme is computationally hiding under the MLWE assumption and computationally binding under the MSIS assumption [LNP22]. Using committed values, we can instantiate Σ-protocols and create proofs of opening, *i.e.,* proofs of knowledge of the committed value, proofs of linear or quadratic relations, and bound relations [Bau+18b; ALS20; ENS20; LNS21a]. Whereas the original construction requires a specific parameterization of the polynomial rings that is incompatible with NTT-friendly prime modulus [Bau+18b], Attema *et al.* [ALS20] introduce a variant to alleviate this issue. In this work,

we employ the parameterization from Esgin *et al.* for the BDLOP commitment scheme [ENS20] and the aforementioned proofs.

4.3 Practically Verifying RLWE Samples

Here, we present the way we combine lattice-based proofs to address the challenges outlined in §4.1 and verify the correctness of Eq. 1. In this subsection, we set $\mathbf{X}$=0, *i.e.,* the local share is an RLWE sample; verifying its correct generation is the basis for all MFHE operations. In §4.4, we will show how to extend our techniques to other MFHE operations (that instantiate Eq. 1 with a non-zero polynomial $\mathbf{X}$).

Verifying the Linear Relation. Recall that, during **Step 1** of the MFHE interactive operations (§3.2.1), each party creates a public polynomial by linearly combining the secret and noise polynomials sampled from specific distributions (*i.e.,* ternary and bounded Gaussian, resp.). To prove the correctness of this linear combination, we treat it as proving the knowledge of a solution to the linear equation and rely on the lattice-based interactive proof of Esgin *et al.* [ENS20].

Verifying the Ternarity of the Secret Polynomial. We verify the ternary property of the secret key polynomial $\mathbf{s}$ by checking that its coefficients satisfy the equation $x(x-1)(x+1)=0$ following Esgin *et al.* [ENS20].

Verifying the Norm of Noise Polynomials. To verify the bound of the noise polynomials, prior works opted for bounds in ℓ_2-norm (*e.g.,* [LNP22]), approximate range proofs (*e.g.,* [Bau+18b; LNP22; LNS21a; BL17]) or simply considered ternary noise [ENS20]. However, these approaches are unsuitable for modern (M)FHE implementations that cannot tolerate a knowledge gap and that, for security, use large noises in infinity norm. Our solution is to employ a ternary decomposition of the noise as proof of shortness in order to avoid any knowledge gap that could compromise the security of the MFHE pipeline. More precisely, for a noise $\mathbf{e} \in \mathcal{R}_q$ such that $\|\mathbf{e}\|_\infty \leqslant B$, we follow the observation made by Ling *et al.* [Lin+13] that the subset sum of $b_1{=}\lceil \frac{B}{2} \rceil$, $b_2{=}\lceil \frac{B-b_1}{2} \rceil$, $b_3{=}\lceil \frac{B-b_1-b_2}{2} \rceil$, $\ldots$, $b_k{=}1$ (with $k{=}\lceil \log(B) \rceil$), covers exactly the set $\{0,\ldots,B\}$. Thus, we perform the ternary decomposition of $\mathbf{e}$ as follows:

$$\mathbf{e} = \sum_{j=1}^{k} b_j \mathbf{e}_j \quad \text{with } \mathbf{e}_j \in \{-1,0,1\}^N$$

Using the ternary proof from Esgin *et al.* [ENS20], we can prove the exact bound of the noise polynomials.

Conversion between NTT and Polynomial Domains. Recall that for efficiency reasons, (M)FHE implementations employ the NTT representation and that secret keys are stored in this form (rather than their polynomial one). However, the bound proofs need to be executed in the polynomial domain to ensure the key ternarity. To account for this, and because of the linearity of the NTT transform, we decide to incorporate the NTT transformation matrix $\mathbf{T}$ directly in the statement, *i.e.,* we make it part of the linear relation to be proven.

Statement 1: RLWE sample over $\mathcal{R}_q$

Public Inputs: $\mathbf{a}, \mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}$
Private Inputs: $\mathbf{s}, \{\mathbf{e}_j\}_{j=1}^k \in \mathcal{R}_q, \mathbf{r} \in \mathcal{R}_q^2$, and $\vec{\mathrm{k}} \in \mathbb{Z}_q^N$
Outputs: $\mathbf{p}_0, \mathbf{c}_{\mathrm{ajtai}} \in \mathcal{R}_q$

- Linear Relation:
  $\mathbf{p}_0 = \mathbf{a} \circ \mathbf{T}\mathbf{s} + \sum_{j=1}^k \mathbf{T} b_j \mathbf{e}_j$
  $\mathbf{c}_{\mathrm{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}} p$
- Ternary Checks: $\|\mathbf{s}\|_\infty, \|\mathbf{r}\|_\infty, \|\mathbf{e}_j\|_\infty \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}\|_\infty \ll q$

Linking the Secret Polynomial. We use the commitment scheme by Ajtai [Ajt96] to link the secret polynomial across the different sub-rings (of the RNS representation), and we commit to the secret polynomials over a smaller space $\mathbb{Z}_p$ with $p \leqslant q_j$ for $j \in [0, L]$. With appropriate parameterization, the commitment coefficients will never exceed q_j. As a result, there is no wrap-around modulo q_j, and any value in $\mathbb{Z}_{q_j}$ or in $\mathbb{Z}$ (hence, across the different sub-rings) is the same. In particular, the commitment of a ternary value $\mathbf{m}$ with randomness $\mathbf{r}$ has the following form:

$$\mathbf{A}_1 \cdot \mathbf{m} + \mathbf{A}_2 \cdot \mathbf{r} = \mathbf{c}_{\mathrm{ajtai}} \mod p,$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{Z}_p^{\kappa N \times 2N}$. Similarly to Del Pino *et al.* [DPLS18], we rewrite it in $\mathbb{Z}$ to ensure no wrap-around $\mod q_j$ as

$$\mathbf{A}_1 \cdot \mathbf{m} + \mathbf{A}_2 \cdot \mathbf{r} = \mathbf{c}_{\mathrm{ajtai}} + \vec{\mathrm{k}} \cdot p$$

However, contrary to their approach, ours operates directly in the NTT domain and there is no quotient modulo X^N+1, thus making the vector $\vec{\mathrm{k}}$ unique per sub-ring by Euclid's Lemma. As both $\mathbf{m}$ and $\mathbf{s}$ are ternary vectors, $\|\mathbf{c}_{\mathrm{ajtai}} + \vec{\mathrm{k}} \cdot p\|_\infty \leqslant 2Np$. As a result, ensuring that, for all $j \in [0, L]$, $q_j \gg 2Np$ guarantees no wrap-around modulo any of the q_j. This commitment entails that the witness is extended to also comprise the commitment randomness $\mathbf{r}$ and the quotient $\vec{\mathrm{k}}$. To prove that $\|\vec{\mathrm{k}}\|_\infty \ll q_j$, we rely on an approximate bound proof (see Appendix §C). Note that, in practice, both ternarity and consistency of the secret key are checked simultaneously.

Assembling the Statement. We combine the blocks above and build a statement that, if satisfied, ensures the correct generation of an RLWE sample over $\mathcal{R}_q$, as shown in Statement 1. In more detail, Statement 1 encompasses a proof of opening that checks the correctness of the commitments, a linear-relation proof that verifies the RLWE and commitment relations, a cubic proof that checks if $\mathbf{s}, \mathbf{e}_1, \ldots, \mathbf{e}_k, \mathbf{r}$ are ternary, and an approximate bound proof on $\vec{\mathrm{k}}$ that ensures there is no wrap-around in creating the Ajtai commitment. Then, we use a lattice-based proof system [ALS20; ENS20; LNS20] made non-interactive

with the Fiat-Shamir heuristic [FS86], and we obtain a non-interactive zero-knowledge (NIZK) proof for RLWE samples. We rewrite, in particular, the set of equations and constraints of Statement 1 as an unstructured linear relation $(\mathbf{p}_0, \mathbf{c}_{\text{ajtai}})^T = \mathbf{A}\mathbf{w}$, where $\mathbf{w}^T = (\mathbf{s}, \mathbf{r}, \mathbf{e}_1, \ldots, \mathbf{e}_k, \vec{\mathrm{k}})$ is the witness and $\mathbf{A}$ is a sparse matrix of the form:

$$\mathbf{A} = \begin{pmatrix} -\mathrm{diag}(\vec{\mathtt{p}}_1)\cdot\mathbf{T} & \mathbf{0} & \mathbf{T}\cdot\mathbf{Id}b_1 & \ldots & \mathbf{T}\cdot\mathbf{Id}b_k & \mathbf{0} \\ \mathtt{A}_1 & \mathtt{A}_2 & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{Id}p \end{pmatrix},$$

with $\vec{\mathtt{p}}_1$ the NTT vector of the polynomial $\mathbf{p}_1$ and $\mathtt{A}_1$ (resp., $\mathtt{A}_2$) the matrix whose product with the coefficients vector of $\mathbf{s}$ returns the NTT form of $\mathbf{A}_1 \cdot \mathbf{s}$. Note that the proof size of the non-interactive protocol is affected by the input dimension of the linear relation but also by the polynomial ring used in the commitment. By carefully crafting the statements and accounting for the NTT transformations, Statement 1 (over $\mathcal{R}_q$) can be seen as a linear relation under constraints over $\mathbb{Z}_q$. In the remainder of this paper, we denote by $\mathfrak{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ the polynomial ring of degree d used by the commitment scheme (with same modulus as $\mathcal{R}_q$). Finally, note that using the witness $\mathbf{w}$, further linear relations can be proven by appending extra rows to the matrix $\mathbf{A}$. In §4.4, we extend Statement 1 and we design tailored statements that can ensure the correctness of the local share generation in various MFHE operations, such as local key generation, collective decryption, public-key switching, and bootstrapping.

Security Analysis. The completeness and the computational honest verifier zero-knowledge of the resulting proof hold by the properties of the underlying proof systems: [ENS20, Th3.1] and [LNS21a].

4.4 Verifying MFHE Local Share Generation

We now describe how we extend Statement 1 to design tailored statements that ensure the correctness of the local share generation in various MFHE operations such as local key generation (§4.4.1), collective decryption (§4.4.2), public-key switching (§4.4.3), and bootstrapping (§4.4.4). As the majority of these operations require handling large polynomials for confidentiality purposes, we first describe how we address this issue.

Handling Large Polynomials. Some MFHE operations require using polynomials with large coefficients. For instance, the collective decryption, key switching, and bootstrapping (see §2.3), all use *smudging noise* [Ash+12] that is sampled from a distribution of very large variance to prevent private information leakage.[2] Furthermore, the collective bootstrapping requires an additional masking polynomial sampled from the plaintext space (*i.e.,* with 16-bits coefficients). Decomposing such polynomials into a binary or ternary representation similar to our approach for handling the noise polynomials (§4.3) is impractical: Indeed, such a decomposition would introduce in the proof system additional polynomial

[2] Ideally 2^λ times the variance of the ciphertext's noise, with λ the security level. In practical implementations, it is between 16 and 40 bits.

inputs (as many as the decomposition size), hence affect both its runtime and proof size. To avoid this issue, we propose a novel trick. Instead of sampling these polynomials with large coefficients, we construct them such that they are indistinguishable from uniformly sampled elements by following the RLWE assumption. For example, we construct the smudging noise $\mathcal{R}_{q_{\text{smdg}}}$ (with q_{smdg} configured by the range of the smudging noise) such that $\mathbf{e}_{\text{smdg}}=\mathbf{a}\cdot\mathbf{s}'+\mathbf{e}$ abiding by the smudging lemma [Ash+12]. Then, we use an observation stemming from the RLWE problem [LS15; Duc+18]: Given a public value $\mathbf{a}\xleftarrow{\$}\mathcal{R}_{q_{\text{smdg}}}$ and a ternary secret $\mathbf{s}'$, then $(\mathbf{a},\mathbf{a}\cdot\mathbf{s}'+\mathbf{e})$ is indistinguishable from a uniformly sampled random pair $(\mathbf{a},\mathbf{b})\in\mathcal{R}^2_{q_{\text{smdg}}}$. However, as the noise $\mathbf{e}_{\text{smdg}}$ is constructed in $\mathcal{R}_{q_{\text{smdg}}}$, we need to ensure there is no wrap-around modulo any of the q_j for $j\in[0,L]$ (*i.e.,* similarly to our approach for linking the secret polynomial across the sub-rings using the Ajtai commitment). For this reason, we commit to the quotient to enable the reconstruction from $\mathbb{Z}_{q_{\text{smdg}}}$ to $\mathbb{Z}$ and then to $\mathbb{Z}_{q_j}$, under the condition that there is no wrap-around modulo q_j (recall that $q=\prod_{j=0}^{L}q_j$). Proving the correctness of this statement requires an additional linear relation, two bound proofs (*i.e.,* ternarity of $\mathbf{s}'$ and $\mathbf{e}<B$), as well as adds $\mathbf{s}'$ and $\mathbf{e}\leftarrow\chi_{\text{err}}$ to the witness. This technique is easily extended to the plaintext mask required for the collective bootstrapping protocol.

4.4.1 Local Key Generation

The MFHE local key generation operation comprises the generation of a private/public key pair (SecKeyGen and EncKeyGen – §3.2), as well as additional evaluation keys that are useful for the homomorphic evaluation, *e.g.,* rotation and relinearization keys (EvalKeyGen). Statement 2 assembles the conditions for the correct generation of a private/public key pair. Each party samples, in particular, a secret key $\mathbf{s}$ and returns the corresponding public key $(\mathbf{p}_0,\mathbf{p}_1)$, as well as the Ajtai commitment $\mathbf{c}_{\text{ajtai}}$ of the secret key. The approximate bound proof ensures no wrap-around modulo each of the q_j in the generation of $\mathbf{c}_{\text{ajtai}}$. A similar approach can be used to ensure the correct generation of rotation keys (see [Mou+21] for details). Then, Statement 3 displays the requirements for the secure generation of the relinearization keys, for a public parameter $\vec{\mathbf{a}}\in\mathcal{R}_q^l$ [Par21]. The value l corresponds to the length of a decomposition basis used to ensure correctness of the relinearization. Each party recomputes the Ajtai commitment of the secret (*i.e.,* $\mathbf{c}_{\text{ajtai}}$) and generates the vectors of public polynomials $\vec{\mathbf{h}}_0$ and $\vec{\mathbf{h}}_1$. As we will see in §5, this statement can be combined with Statement 7 to verify the correctness of the MFHE collective key-generation protocol.

4.4.2 Collective Decryption

To decrypt the computation result, the MFHE parties engage in a collaborative decryption operation (Dec) that introduces smudging noise to prevent indirect leakage from encryption noise correlations [Ash+12]. As seen above, we rely on

Statement 2: Local Key Generation

Public Inputs: $\mathbf{p}_1 \in \mathcal{R}_q, \mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}$
Private Inputs: $\mathbf{s}, \{\mathbf{e}_j\}_{j=1}^k \in \mathcal{R}_q$ $\mathbf{r} \in \mathcal{R}_q^2$,and $\vec{\mathrm{k}} \in \mathbb{Z}_q^N$
Outputs: $\mathbf{p}_0, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$
- Linear Relation:
  $\mathbf{p}_0 = -\mathbf{p}_1 \circ \mathbf{T} \cdot \mathbf{s} + \mathbf{T} \cdot \sum_{j=1}^k b_j \mathbf{e}_j$
  $\mathbf{c}_{\text{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}} p$
- Ternary Checks: $\|\mathbf{s}\|_\infty, \|\mathbf{r}\|_\infty, \|\mathbf{e}_j\|_\infty \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}\|_\infty \ll q$

Statement 3: Relinearization Key Generation

Public Inputs: $\mathbf{cpk} = (\mathbf{a}, \mathbf{b})$, $\mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}$
Private Inputs: $\mathbf{s}, \vec{\mathbf{u}} \in \mathcal{R}_q$, $\mathbf{r} \in \mathcal{R}_q^2$ $\{\vec{\mathbf{e}_{0j}}\}_{j=1}^k, \{\vec{\mathbf{e}_{1j}}\}_{j=1}^k \in \mathcal{R}_q^l$, and $\vec{\mathrm{k}}_1 \in \mathbb{Z}_q^N$
Outputs: $\vec{\mathbf{h}}_0, \vec{\mathbf{h}}_1, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$
- Linear Relation:
  $\vec{\mathbf{h}}_0 = \mathbf{a} \circ \mathbf{T} \cdot \vec{\mathbf{u}} + \vec{\omega} \circ \mathbf{T} \cdot \mathbf{s} + \mathbf{T} \cdot \sum_{j=1}^k b_j \vec{\mathbf{e}}_{0j}$
  $\vec{\mathbf{h}}_1 = \mathbf{b} \circ \mathbf{T} \cdot \vec{\mathbf{u}} + \mathbf{T} \cdot \sum_{j=1}^k b_j \vec{\mathbf{e}}_{1j}$
  $\mathbf{c}_{\text{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}}_1 \cdot p$
- Ternary Checks: $\|\mathbf{s}\|_\infty, \|\vec{\mathbf{u}}\|_\infty, \|\mathbf{r}\|_\infty, \|\vec{\mathbf{e}}_{0j}\|_\infty, \|\vec{\mathbf{e}}_{1j}\|_\infty \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}_1\|_\infty \ll q$

an additional linear relation in a different ring $\mathcal{R}_{q_{\text{smdg}}}$. We also account for the quotient $\vec{k}_2$ and ensure that there is no wrap-around modulo any of the q_j for $j \in [0, L]$ in the reconstruction of the smudging noise. Statement 4 summarizes, during collective decryption, the conditions for proving the correctness of the parties' operations. Each party re-computes the Ajtai commitment $\mathbf{c}_{\text{ajtai}}$ of the secret key used in previous MFHE protocols, generates a smudging noise $\mathbf{e}$ and, finally, it outputs the partial decryption $\mathbf{h}$. The ternary checks ensure that the secret polynomials and noise decompositions are well-formed and the two approximate bound proofs guarantee no wrap-around of $\mathbf{e}$ and $\mathbf{c}_{\text{ajtai}}$ modulo each q_j (for $j \in [0, L]$).

4.4.3 Collective Public-key Switching

In ThHE and hybrid settings (§2.3), the MFHE parties execute the public-key switching operation in order to re-encrypt the computation result under the key of an external computation-result receiver. From high-level, this protocol corresponds to each party encrypting as $(\mathbf{h}_0, \mathbf{h}_1)$ their own partial decryption share under the receiver's public-key. Hence, the desired statement is similar

Statement 4: Collective Decryption

Public Inputs: $\mathbf{c}_1 \in \mathcal{R}_q, \mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}, \mathbf{A}_3 \in \mathcal{R}_{q_{\text{smdg}}}$
Private Inputs: $\mathbf{s}, \mathbf{s}', \{\mathbf{e}_j\}_{j=1}^k \in \mathcal{R}_q, \mathbf{r} \in \mathcal{R}_q^2$ and $\vec{\mathrm{k}}_1, \vec{\mathrm{k}}_2 \in \mathbb{Z}_q^N$
Outputs: $\mathbf{h}, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$

- Linear Relation:

$$\mathbf{h} = \mathbf{c}_1 \circ \mathbf{T} \cdot \mathbf{s} + \mathbf{e}$$
$$\mathbf{e} = \mathbf{A}_3 \circ \mathbf{T} \cdot \mathbf{s}' + \mathbf{T} \cdot \textstyle\sum_{j=1}^k b_j \mathbf{e}_j - \vec{\mathrm{k}}_2 q_{\text{smdg}}$$
$$\mathbf{c}_{\text{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}}_1 p$$

- Ternary Checks: $\|\mathbf{s}\|_\infty, \|\mathbf{s}'\|_\infty, \|\mathbf{r}\|_\infty, \|\mathbf{e}_j\|_\infty \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}_1\|_\infty, \|\vec{\mathrm{k}}_2\|_\infty \ll q$

Statement 5: Public-key Switching

Public Inputs: $\mathbf{c}_1, \mathbf{p}_0', \mathbf{p}_1' \in \mathcal{R}_q, \mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}, \mathbf{A}_3 \in \mathcal{R}_{q_{\text{smdg}}}$
Private Inputs: $\mathbf{s}, \mathbf{s}', \mathbf{u}, \{\mathbf{e}_{0j}\}_{j=1}^k, \{\mathbf{e}_{1j}\}_{j=1}^k \in \mathcal{R}_q, \mathbf{r} \in \mathcal{R}_q^2$ and $\vec{\mathrm{k}}_1, \vec{\mathrm{k}}_2 \in \mathbb{Z}_q^N$
Outputs: $\mathbf{h}_0, \mathbf{h}_1, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$

- Linear Relation:

$$\mathbf{h}_0 = \mathbf{c}_1 \circ \mathbf{T} \cdot \mathbf{s} + \mathbf{p}_0' \circ \mathbf{T} \cdot \mathbf{u} + \mathbf{e}_0$$
$$\mathbf{h}_1 = \mathbf{p}_1' \circ \mathbf{T} \cdot \mathbf{u} + \mathbf{T} \cdot \textstyle\sum_{j=1}^k b_j \mathbf{e}_{1j}$$
$$\mathbf{c}_{\text{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}}_1 p$$
$$\mathbf{e}_0 = \mathbf{A}_3 \circ \mathbf{T}\mathbf{s}' + \mathbf{T} \cdot \textstyle\sum_{j=1}^k b_j \mathbf{e}_{0j} - \vec{\mathrm{k}}_2 q_{\text{smdg}}$$

- Ternary Checks: $\mathbf{s}, \mathbf{s}', \mathbf{r}, \mathbf{u}, \mathbf{e}_{0j}, \mathbf{e}_{1j} \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}_1\|_\infty, \|\vec{\mathrm{k}}_2\|_\infty \ll q$

to the collective decryption one (*c.f.* Statement 4) with two additional linear relations to verify the encryption under the receiver key (which is considered a public input).

The resulting statement is given as Statement 5.

4.4.4 Collective Bootstrapping

When the ciphertext's capacity has been exhausted, the MFHE parties execute a collective bootstrapping protocol to refresh its noise and to enable further homomorphic computations on it. In a nutshell, during this protocol, each party re-encrypts a masked partial decryption of the ciphertext. To avoid plaintext leakage, the collective bootstrapping protocol requires a mask $\mathbf{M} \in \mathcal{R}_t$ and a smudging noise of large norm. Statement 6 summarizes the necessary conditions for ensuring, during the collective bootstrapping protocol, the correctness of the parties' local shares.

Statement 6: Collective Bootstrapping

Public Inputs: $\mathbf{c}_1, \mathbf{a} \in \mathcal{R}_q, \mathbf{A}_1 \in \mathcal{R}_p, \mathbf{A}_2 \in \mathcal{R}_p^{1\times 2}, \mathbf{A}_3 \in \mathcal{R}_{q_{\text{smdg}}}, \mathbf{A}_4 \in \mathcal{R}_{q_{pt}}$
Private Inputs: $\mathbf{s}, \mathbf{s}', \mathbf{s}'', \{\mathbf{e}_{0j}, \mathbf{e}_{1j}, \mathbf{e}_{2j}\}_{j=1}^{k} \in \mathcal{R}_q, \mathbf{r} \in \mathcal{R}_q^2$ and $\vec{\mathrm{k}}_1, \vec{\mathrm{k}}_2, \vec{k}_3 \in \mathbb{Z}_q^N$
Outputs: $\mathbf{h}_0, \mathbf{h}_1, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$

- Linear Relation:

$$\mathbf{h}_0 = \mathbf{c}_1 \circ \mathbf{T} \cdot \mathbf{s} - \Delta\mathbf{M} + \mathbf{e}_0$$
$$\mathbf{h}_1 = -\mathbf{a} \circ \mathbf{T} \cdot \mathbf{s} + \Delta\mathbf{M} + \mathbf{T} \cdot \textstyle\sum_{j=1}^{k} b_j \mathbf{e}_{1j}$$
$$\mathbf{c}_{\text{ajtai}} = \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{A}_2 \cdot \mathbf{r} - \vec{\mathrm{k}}_1 p$$
$$\mathbf{e}_0 = \mathbf{A}_3 \circ \mathbf{T} \cdot \mathbf{s}' + \mathbf{T} \cdot \textstyle\sum_{j=1}^{k} b_j \mathbf{e}_{0j} - \vec{\mathrm{k}}_2 q_{\text{smdg}}$$
$$\mathbf{M} = \mathbf{A}_4 \circ \mathbf{T} \cdot \mathbf{s}'' + \mathbf{T} \cdot \textstyle\sum_{j=1}^{k} b_j \mathbf{e}_{2j} - \vec{\mathrm{k}}_3 q_{pt}$$

- Ternary Checks: $\mathbf{s}, \mathbf{s}', \mathbf{s}'', \mathbf{r}, \mathbf{e}_{0j}, \mathbf{e}_{1j}, \mathbf{e}_{2j} \leqslant 1$ for $j \in \{1, \ldots, k\}$
- Approximate Bound Proof: $\|\vec{\mathrm{k}}_1\|_\infty, \|\vec{\mathrm{k}}_2\|_\infty, \|\vec{\mathrm{k}}_3\|_\infty \ll q$

5 Verifiable Share Aggregation for MFHE

MFHE interactive operations, such as public-key generation, decryption, public-key switching, and bootstrapping, rely on an aggregator to combine together the parties' local shares (**Step 2** – §3.2.1). A malicious aggregator can tamper with the aggregation operation and harm the confidentiality of the honest parties and/or the correctness of the MFHE pipeline (see §3.2); the same holds if MFHE parties collude with the aggregator. While ensuring the correct concatenation of the parties' shares (*e.g.,* in MkHE schemes) is trivially solved by concatenating their verified versions (§4), verifying their summation remains a challenge. Hence, in this section, we design a novel verifiable aggregation protocol that guarantees the correct combination of the parties' locally generated key material. Our protocol uses a probabilistic polynomial equality test to ensure the returned polynomial is indeed the sum of each of the parties' local polynomials. Note that, under our threat model (§3.1), existing protocols for maliciously secure aggregation based on splitting trust among multiple servers [Rat+23] or on verifiable secret sharing [Ben86], ensure confidentiality but not correctness. As the MFHE aggregation operation sums up public polynomials, it does not require confidentiality but an efficient transparent proof of correctness. Other approaches, *e.g.,* using generic proof systems for rings [GNSV21], would require a designated verifier and lead to significant overhead (setup and prover).

Overview of our Approach. As discussed in §3.2, a correct aggregation operation entails the summation of the public polynomials returned by each party's local share generation, *i.e.,* an honest aggregator returns $\mathbf{p} = \sum_i \mathbf{p}_i$, $\forall \mathcal{P}_i \in \mathcal{P}$. Hence, verifying aggregation correctness is equivalent to checking for polynomial equality. Using probabilistic polynomial checks, we build our succinct protocol: Instead of checking the above equality for all the N coefficients of each polynomial, we verify the correctness of the polynomial evaluation on a random challenge point. By the polynomial identity lemma in $\mathbb{Z}_q$ (Schwartz-Zippel) [Sch80],

the probability of collision in the aggregation is N/q. In other words, the probability that the returned aggregate (*i.e.,* $\mathbf{p}$) indeed sums up the different local shares (*i.e.,* $\mathbf{p}_i$, $\forall \mathcal{P}_i \in \mathcal{P}$) is $1-N/q$ (typically, with $\log N$ between 13 and 15 and $\log q$=54 leading to four points being sampled for soundness of at least 2^{-128}).

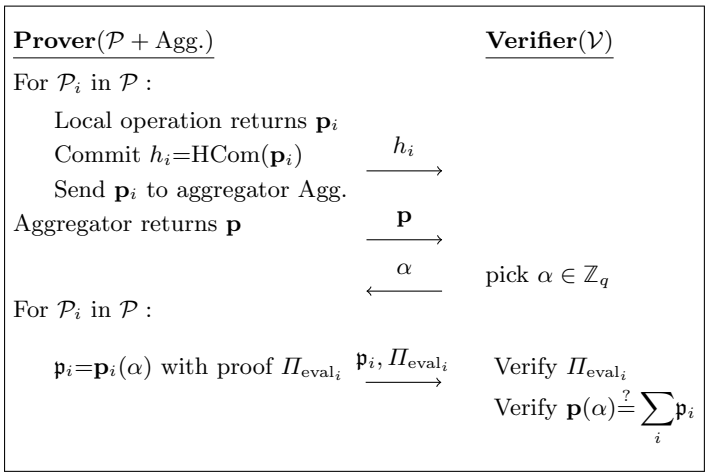

Fig. 3: An illustration of the verifiable aggregation interactive protocol. In practice, this protocol is made non-interactive using the Fiat-Shamir heuristic, i.e., the challenge α is generated in the random oracle model (ROM) from the parties' commitments h_i and the aggregate $\mathbf{p}$. Without loss of generality, we consider that each party's local share generation returns a single polynomial $\mathbf{p}_i$. The proof Π_{eval_i} ensures correct evaluation of the polynomial committed in c_i on the challenge value α.

Verifiable Aggregation Protocol. As in our threat model all but one MFHE parties are potentially malicious, we consider the parties and the aggregator as the *prover* that generates a proof that can be verified by a *verifier* with a Σ-protocol. In more detail, each party $\mathcal{P}_i$ first runs its local share generation (§4) and obtains a polynomial $\mathbf{p}_i$ that it commits to using a standard hash-based commitment scheme (HCom($\cdot$)). Then, each party publishes its commitment and sends $\mathbf{p}_i$ to the aggregator that, in turn, returns the aggregate $\mathbf{p}$. The verifier samples a random challenge point $\alpha \in \mathbb{Z}_q$ and publishes it. Each party evaluates its local polynomial $\mathbf{p}_i$ on α and publishes a proof of correct evaluation. Finally, the verifier checks the correctness of the parties' commitments and of the polynomial evaluations on the challenge point, and it verifies that their sum is indeed equal to $\mathbf{p}(\alpha)$. An illustration of our verifiable aggregation protocol can be seen in Figure 3. Note that in practice, this Σ-protocol is made non-interactive in the random oracle model (ROM) by using the Fiat-Shamir heuristic [FS86].

To realize, in a practical manner, the verifiable aggregation protocol for MFHE pipelines, we combine it with the proofs of local share generation produced by the parties (§4.4). In particular, when creating a proof of correct local

Statement 7: Verifiable Aggregation

Public Inputs: $\mathbf{p}_i, \alpha$
Outputs: $\mathfrak{p}_i$
- Linear Relation: $\mathfrak{p}_i = \mathbf{p}_i(\alpha)$

share generation, each party also appends a linear relation to the statements of §4.4 to prove the correctness of the polynomial evaluation on the challenge point for the verifiable aggregation protocol (Statement 7). Following this approach, our verifiable aggregation protocol (i) only costs one additional linear relation, (ii) works directly with our cyclotomic rings, (iii) is publicly verifiable, and (iv) does not rely on extra cryptographic assumptions. In practice, as we show in §6, the addition of this linear relation is negligible to the prover's runtime and, due to the properties of the proof system, does not affect the proof size. As a result, in the ROM, our aggregation protocol costs only one hash-based commitment digest per party.

Remark. Note that, alternatively, our verifiable aggregation protocol could be realized by employing a polynomial commitment scheme (PCS) that enables proofs of correct evaluation on committed polynomials, *e.g.,* KZG [KZG10], Hyrax [Wah+18], FRI [BS+18], or DARK [BFS20]. However, these schemes require a trusted setup, *e.g.,* [KZG10], or rely on cryptographic assumptions (*e.g.,* discrete logarithm, Reed-Solomon codes) different from those made for (M)FHE pipelines. On the contrary, our approach does not require additional assumptions and introduces minimal communication and computation overhead, *i.e.,* it comes almost for *free* in our overall construction for verifying the correctness of MFHE pipelines.

6 Implementation and Evaluation

We introduce Pelta , an implementation of our *commit-then-prove* construction for securing MFHE pipelines against malicious adversaries. We evaluate Pelta over different MFHE protocols and compare its performance with prior work.

6.1 Implementation

We implement Pelta in Golang on top of the Lattigo library [EPF21] that supports MFHE pipelines. We use Lattigo's polynomial ring package and we create a new lattice-based commitment library that implements the BDLOP [Bau+18b] and Ajtai [Ajt96] commitments. Based on this commitment library, we implement the proof systems from Esgin *et al.* [ENS20] and Lyubashevsky *et*

al. [LNS21a] to verify the correctness of the statements proposed in the previous sections (§4 and §5). Our code is available for review and will be made open source.

6.2 Experimental Setup

We focus on key MFHE protocols, i.e., key generation (§4.4.1), collective decryption (§4.4.2), public-key switching (§4.4.3), and collective bootstrapping (§4.4.4); these play a critical role in MFHE pipelines (both MkHE and ThHE – see §3.2). By default, we configure the security parameters of the lattice-based commitment scheme to achieve at least 128-bit security, as in prior work [ENS20]. We set, in particular, the MSIS and MLWE parameters to κ=1 and λ=1 for a $\mathfrak{R}_q$ of degree d=2^{13}. Unless otherwise specified throughout the experiments, we consider FHE parameters with polynomial degree of N=2^{13} ($\log q$=218). We display the results for a single sub-ring of the RNS representation because the benchmark costs are linear in the number of q_j (see Appendix A) and, because the sub-rings are independent, they are in practice parallelizable. All our experiments are conducted on a machine with two Intel Xeon E5-2680 v3 processors running at 2.5GHz over 12 cores and equipped with 256GB of RAM, and the results are averaged over 10 repetitions. We report Pelta's performance both for a single-threaded implementation (Pelta single-th.) as well as an optimised version that uses an underlying multi-threaded package for some polynomial operations (Pelta multi-th.).

Baselines. We compare Pelta with two different approaches from prior work that we extend to support the different statements presented in §4 and §5. The first is an MPC solution based on cut-and-choose (C&C) [CP16]. In more detail, the prover first commits to $\mathcal{K}$ protocol iterations. Then, the verifier picks one of the commitments and the prover reveals the secrets of the remaining $\mathcal{K}-1$ protocol executions. Finally, the verifier checks that the commitment openings and the revealed secrets lead to correct protocol executions. We set $\mathcal{K}$ to 100K to have comparable runtimes with the other approaches. However, we note that this protocol yields low security (16-bit). The second baseline is a SNARK-based approach proposed by Boschini *et al.* [Bos+20] that relies on the Aurora [BS+19] proof system which is configured to achieve 128-bit security.

Evaluation Metrics. We compare the performance of Pelta and the baselines by measuring their overhead in terms of setup, prover, and verifier runtimes, for both its single and multi-threaded versions. The setup times comprise the sampling of the input polynomials, the creation of the linear relations, and the generation of the BDLOP public parameters. We also measure their proof sizes.

6.3 Performance Analysis

We first compare the performance of Pelta and the two baselines for the local key-generation protocol (§4.4.1). Table 1 shows that Pelta outperforms both

Table 1: Performance results of the local key-generation protocol (§4.4.1) with $\log N$=13 over a single sub-ring of the RNS representation.

	Setup	Prover (s)	Verifier (s)	Proof (MB)
C&C	-	463	463	6.4
[Bos+20]	>5min	845	312	0.463
Pelta (single-th.)	12.2s	14.0	14.9	2.05
Pelta (multi-th.)	10.4s	9.5	10.5	2.05

baselines. In particular, single-threaded Pelta is ∼30× faster than the cut-and-choose protocol (while achieving much higher security) and one order of magnitude more efficient than the Aurora-based approach. Although Pelta yields a slightly larger proof than the latter, we argue that its size is reasonable considering the MFHE communication costs; a single public key is already 0.45MB and evaluation keys for practical applications can be in the order of GBs (see [Sav+21]). Due to the overhead and low security achieved by C&C, as well as to the fact that this approach can not be used to prove key consistency across MFHE protocol executions without additional extensions, in the subsequent protocol evaluations, we focus on the comparison between Pelta and [Bos+20]. We do not report results for the relinearization key-generation protocol as the Aurora-based approach required more than 256GB of RAM. We additionally observe that the multi-threading of the underlying math operations can speed up Pelta by a factor 1.5.

Table 2 shows that Pelta (single-threaded) achieves at least one order of magnitude faster prover runtimes, compared to the Aurora-based approach for the various MFHE collective protocols. Pelta's prover execution is, in particular, 87× faster for the collective decryption (Table 2(a)), and 67× and 42× more efficient for the collective public-key switching and bootstrapping protocols, resp. (Tables 2(b) and 2(c)). Whereas the verifier runtimes do not yield a similar gap due to Aurora's verification efficiency, Pelta remains at least 14× faster. Then, although [Bos+20] achieve a smaller proof size due to Aurora's succinctness, we note that Pelta's proof sizes per sub-ring are in similar ranges. Finally, Table 2 demonstrates that our approach has a considerably faster setup phase (15×).

Influence of the Aggregation. We incorporate the aggregation relation into the different statements of §4 as discussed in §5. Table 3 shows the performance results for the collective key-generation protocol (Statements 2 and 7), for [Bos+20] and Pelta. In both cases, we observe that the proof size is only slightly modified due to Aurora's succinctness and the lattice-based proof's construction (*c.f.* Table 1). Although the runtime of Pelta is almost not affected, the inclusion of the aggregation relation significantly impacts Boschini *et al.*'s approach [Bos+20]. Indeed, for Aurora, the new polynomial evaluations (used for our compression technique in §5) and the public outputs increase the number of variables and create additional constraints on the back end. For the lattice-based proof we employ, the addition of the linear constraint affects only slightly the computation runtime (*i.e.,* the operations involving the matrix $\mathbf{A}$, §4.3 and

Table 2: Performance results of various MFHE protocols with $\log N$=13 over a single sub-ring of the RNS representation.

(a) Collective decryption (§4.4.2).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[Bos+20]	>8min	1,487	401	0.496
Pelta (single-th.)	16.1s	16.9	17.9	2.37
Pelta (multi-th.)	11.6s	11.3	12.5	2.37

(b) Public-key switching (§4.4.3).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[Bos+20]	>12min	1,681	580	0.495
Pelta (single-th.)	25.0s	24.5	26.2	3.15
Pelta (multi-th.)	20.2s	16.6	18.3	3.15

(c) Collective bootstrapping (§4.4.4).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[Bos+20]	>12min	1,649	566	0.495
Pelta (single-th.)	48.2s	38.6	40.8	3.9
Pelta (multi-th.)	34.8s	25.3	27.6	3.9

Figure 4). As a result, the aggregation comes almost for *free* due to our trick of using a polynomial evaluation to verify it (§5).

Influence of the FHE Polynomial Ring $\mathcal{R}_q$. We evaluate Pelta's performance with various polynomial rings commonly used in FHE pipelines, *i.e.,* rings whose degree ranges between N=2^{11} and N=2^{15} with the corresponding moduli of the Lattigo library [EPF21]. Recall that the ring degree and the modulus depend on the security requirement and the hardness of the RLWE problem [Alb+18]. For simplicity, we consider a single modulus, as we experimentally observe a linear correlation with the number of sub-rings (see Table 5 in Appendix §A). As expected, we observe in Table 4 an exponential correlation between both computation and communication complexity (for both prover and verifier) and the FHE-ring degree.

Influence of the Commitment Polynomial Ring $\mathfrak{R}_q$. While the proof sizes of Pelta are acceptable compared to the communication overhead already induced by MFHE pipelines, Tables 1–3 show that they are larger than Boschini *et al.* [Bos+20]. Nonetheless, Pelta's proof size can be reduced and made on par with [Bos+20] by opting for a smaller commitment ring $\mathfrak{R}_q$ and by adapting the parameterization of the commitment scheme such that the MLWE and MSIS problems remain hard (see Table 6 in Appendix B). For instance, the proof size of the key-generation protocol can be reduced from 2.05MB to 1.3MB (1.5MB) by using a commitment ring degree $\log d$=7 ($\log d$=10) at the cost of slightly longer runtimes (2× to 3×) due to a higher number of costly ring operations. As such, Pelta can achieve a tradeoff between runtime and proof size.

Table 3: Performance of the collective key-generation protocol (*i.e.,* Statements 2 and 7), per sub-ring ($\log N$=13), with aggregation proof.

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[Bos+20]	>14min	1,151	606	0.464
Pelta (single-th.)	12.7s	14.9	15.8	2.05
Pelta (multi-th.)	11.2s	9.8	10.9	2.05

7 End-to-End MFHE Pipeline Security

Our construction ensures that, during the MFHE interactive operations, the parties correctly generate the key material and execute the protocol steps. However, to guarantee the validity of the computation output and the end-to-end security of the MFHE pipeline in the presence of malicious adversaries, additional techniques for verifying the correctness of non-interactive operations (§3.2.2) are required. In particular, countermeasures that ensure (i) the validity of the homomorphic evaluation (§7.1), as well as (ii) the trustworthiness of the input provided to the computation (§7.2), should be incorporated. As discussed in §3.2.2, such countermeasures are orthogonal and complementary to Pelta.

7.1 Homomorphic Evaluation Correctness

Verifying the correctness of the homomorphic evaluation (Eval– §3.2.2) in MFHE pipelines is crucial, as a malicious evaluator could tamper with the computation such that the parties decrypt an invalid result. Although several works propose techniques for proving the correctness of homomorphic evaluation, we note that several open issues hinder their practical application and that further research is required. For instance, works that rely on homomorphic message authentication codes (HMACs) to verify the operations executed on the ciphertexts are limited to quadratic functions and do not support critical FHE features such as batching [Lai+14; LWZ18; LWX22; CMP14; TPD16; Lib+13; Che+18; JY14]. Whereas a recent work by Chatel *et al.* [Cha+22] generalizes the HMACs approach to any homomorphic operation by using novel plaintext encoding schemes, their work is only suitable for settings with trusted clients. Another line of work combines homomorphic hashing techniques [FGP14] with SNARKs to prove the correctness of the homomorphic evaluation [FNP20; Boi+21]. These solutions cannot, however, cope with crucial FHE operations such as key-switching and relinearization (§2.2) due to the incompatibility of the SNARKs with the rings used by FHE schemes. The recent work by Ganesh *et al.* introduced a new SNARK for rings [GNSV21], but their approach is not directly suitable for MFHE settings, as it operates in the designated verifier model. Finally, solutions based on trusted execution environments (TEEs), *e.g.,* [Nat+21], introduce different trust assumptions and are potentially prone to integrity attacks [Fei+21].

Therefore, we argue that without novel and efficient techniques for proving the homomorphic evaluation correctness, a simple approach to secure this part

Table 4: Pelta's performance (single-th.) for EncKeyGen (§4.4.1) per sub-ring and variable FHE polynomial ring $\mathcal{R}_q$ degree N.

$\log N =$	11	12	13	14	15
Setup (s)	0.9	3.9	11.7	47.6	198.4
Prover (s)	1.3	4.4	13.9	43.9	158.5
Verifier (s)	1.6	3.1	14.8	46.2	163.3
Proof/sub-ring (MB)	1.1	1.4	2.05	3.3	5.8

of MFHE pipelines (*i.e.,* Eval – §3.2.2) is *computation replication* by each party. Note that the homomorphic evaluation process, in particular, does not involve any private information, hence each party can locally re-compute the homomorphic evaluation on the ciphertexts. With our threat model, at least one party is honest (§3.1), hence this party can detect a malicious evaluator and abort the protocol. To offer re-execution accountability, *e.g.,* to prevent a dishonest party from falsely claiming that the ciphertext returned by the evaluator is wrong, we could use a hash-based commitment (HCom$(\cdot)$) approach, as follows: (a) Each MFHE party publicly commits to the output of their local homomorphic evaluation, (b) the evaluator reveals the output of its homomorphic computation, then (c) all the parties open their commitments. If the honest party detects a mismatch between the commitment openings and the output of its own computation, it aborts the protocol.

7.2 Input Correctness

As in any MPC system, active adversaries can also tamper with the MFHE pipeline output by providing invalid or maliciously crafted inputs to the computation. As a result, ensuring correct computation output additionally requires verifying both the validity of the parties' (plaintext) data and its correct encryption (*i.e.,* the ciphertexts generated by Enc– §3.2.2). Although ensuring the correct encryption operation of a valid input can be achieved using techniques similar to §4.3 and to prior work [ENS20; LN17; Bos+20; DPLS19], verifying (plaintext) input correctness is a more challenging problem that requires specialized solutions. We could use commitments and range proofs [Gro11; Bün+18; Cou+21; ENS20] to prove that the input data originates from the plaintext space of the FHE scheme. However, we note that (i) such proofs would be highly inefficient due to the large coefficients (*e.g.,* $16-32$ bits) and (ii) malicious parties can still craft inputs that are within the valid range but mislead the computation output (*e.g.,* similar to data poisoning when training a machine-learning model [Jag+18; BNL12]). Therefore, additional techniques that can verify input correctness based on, *e.g.,* statistical tests [Che+21], proofs of correct computation on authenticated data [Bac+15], or authenticity checks on encrypted data [Cha+21], are required.

8 Related Work

We review prior work related to the primitives used in our constructions.

Lattice-Based Commitments. Our solution is based on lattice-based commitments proposed initially by Baum *et al.* [Bau+18b]. The introduction of the *Fiat-Shamir with Abort* by Lyubashevsky [Lyu09; Lyu12] paved the way for *one-shot* zero-knowledge proofs based on lattice-based assumptions [Ben+16; Bau+18b; Bau+18a]. However, these works can be used only to prove relaxed versions of statements with a knowledge gap. Then, relying on NTT and RNS packing, Esgin *et al.* [Esg+19a] introduced a more efficient exact one-shot solution for non-linear polynomial relations. Similar ideas were employed by Attema *et al.* [ALS20] and by Esgin *et al.* [ENS20] who designed NTT-friendly protocols for efficiently proving products and linear relations of ternary secrets.

Proof of Knowledge for Linear Relations. Several works investigate the problem of proving the exact knowledge of a small-norm solution $\mathbf{s}$ to the linear relation $\mathbf{As} = \mathbf{u}$. The first proposals, *i.e.,* [KTX08; Lin+13], followed the combinatorial approach of Stern [Ste93]. However, a drawback of this approach is the large soundness error that enforces many protocol repetitions for security purposes. Libert *et al.* [Lib+18] employed a similar Stern-proof for lattices, which led to prohibitive proof sizes (see [LNS20] for details). Beullens [Beu20] combined it with a cut-and-choose protocol to decrease the soundness error, but at the cost of higher runtime.

Other works rely on generic proof systems for verifying such linear relations [DPLS18; Bos+20]. For instance, Del Pino *et al.* [DPLS18] used Bulletproof [Boo+16; Bün+18] and Boschini *et al.* [Bos+20] employed Aurora [BS+19]. Although their protocols yield short proofs, they suffer from high runtime and memory requirements (see §6 for a comparison of our construction with [Bos+20]).

By relying on lattice-based commitments and their corresponding zero-knowledge proofs several protocols prove the correctness of the linear relation with low-norm secret [BLS19; Yan+19]. These initial works had prohibitive costs and required specific polynomial rings incompatible with FHE, but improvements on the commitment scheme front by Attema *et al.* [ALS20], Esgin *et al.* [ENS20], and Lyubashevsky *et al.* [LNS21b; LNS20; LNS21a; LNP22], resulted in more efficient and smaller size proofs, as well as achieved support for NTT optimizations. Similar to our construction, that of Bootle *et al.* [BLS19], Yang *et al.* [Yan+19], and Beullens [Beu20], use the BDLOP commitment scheme and its improvements, but with a specific polynomial ring structure that is not compatible with FHE implementations. As Attema *et al.* [ALS20], Esgin *et al.* [ENS20], and Lyubashevsky *et al.* [LNS20] improve [BLS19] and yield more efficient proofs, we extend them to prove statements relevant to MFHE operations. More recently, Bootle *et al.* [Boo+21] replaced the lattice-based commitment scheme with an 'encode-then-hash' *ideal linear commitment* (ILC) [Boo+17], which enables asymptotically more efficient proofs but at the cost of large additive overhead that grows logarithmically with the number of instances.

In a different paradigm, Baum and Nof used MPC-in-the-head combined with a cut-and-choose protocol in order to prove the linear relation [BN20]. Their approach leads, however, to proofs that are at least an order of magnitude larger than commitment-based approaches [Beu20]. More recently, Rotaru *et al.* [Rot+22] proposed an actively secure setup for SPDZ [Dam+12]; it supports distributed BGV key generation. Their solution introduces a significant computation and communication overhead even for the simple public keys used in SPDZ.[3] For instance, the key generation among two parties requires more than 40min ($\log N$=15 and only two RNS sub-rings). Although this may be satisfactory for the SPDZ setup phase, it is far from practical for modern MFHE pipelines.

9 Conclusion

In this work, we have introduced the first practical construction for thwarting malicious adversaries in multiparty fully homomorphic encryption (MFHE) pipelines. Built on lattice-based commitments and zero-knowledge proofs, our solution addresses the challenges introduced by the structure of modern FHE schemes and their implementation optimizations. We implemented our construction in Pelta and our experimental results showed that it achieves more than one order of magnitude faster runtimes than solutions based on generic proof systems on key MFHE operations. Our solution is a necessary first step toward building fully malicious-resistant MFHE pipelines and our implementation will be made open source.

References

[AM+16] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. "NFLlib: NTT-based fast lattice library". In: *Topics in Cryptology – CT-RSA*. https://doi.org/10.1007/978-3-319-29485-8_20. 2016.

[Ajt96] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Annual ACM Symposium on Theory of Computing (STOC)*. https://doi.org/10.1145/237814.237838. 1996, pp. 99–108.

[Alb+18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. *Homomorphic Encryption Security Standard*. Tech. rep. http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf. Toronto, Canada: HomomorphicEncryption.org, 2018.

[3] For its purposes, SPDZ instantiates an HE scheme with only two sub-rings and no homomorphic evaluation keys are required.

[ATP21] Andreea B Alexandru, Anastasios Tsiamis, and George J Pappas. "Encrypted distributed Lasso for sparse data predictive control". In: *IEEE Conference on Decision and Control (CDC)*. https://doi.org/10.1109/CDC45484.2021.9683401. IEEE. 2021, pp. 4901–4906.

[AH19] Asma Aloufi and Peizhao Hu. "Collaborative homomorphic computation on data encrypted under multiple keys". In: *International Workshop on Privacy Engineering (IWPE'19)* (2019). https://doi.org/10.48550/arXiv.1911.04101.

[Alo+19] Asma Aloufi, Peizhao Hu, Harry WH Wong, and Sherman SM Chow. "Blindfolded evaluation of random forests with multi-key homomorphic encryption". In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* (2019). https://doi.org/10.1109/TDSC.2019.2940020.

[Ash+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-642-29011-4_29. Springer. 2012, pp. 483–501. DOI: 10.1007/978-3-642-29011-4_29.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. "Practical Product Proofs for Lattice Commitments". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-030-56880-1_17. Springer. 2020, pp. 470–499.

[Bac+15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M Reischuk. "ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2015.24. IEEE. 2015, pp. 271–286.

[Baj+17] Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. "A full RNS variant of FV like somewhat homomorphic encryption schemes". In: *Selected Areas in Cryptography – SAC*. https://doi.org/10.1007/978-3-319-69453-5_23. Springer. 2017, pp. 423–442.

[Bau+18a] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafael del Pino, Jens Groth, and Vadim Lyubashevsky. "Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-319-96881-0_23. Springer. 2018, pp. 669–699. DOI: 10.1007/978-3-319-96881-0_23.

[Bau+18b] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. "More efficient commitments from structured lattice assumptions". In: *Security and Cryptography for Networks (SCN)*. https://doi.org/10.1007/978-3-319-98113-0_20. Springer. 2018, pp. 368–385.

[BL17] Carsten Baum and Vadim Lyubashevsky. "Simple amortized proofs of shortness for linear relations over polynomial rings". In: *Cryptology ePrint Archive* (2017). https://eprint.iacr.org/2017/759.

[BN20] Carsten Baum and Ariel Nof. "Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography". In: *Public-Key Cryptography – PKC*. https://doi.org/10.1007/978-3-030-45374-9_17. Springer. 2020, pp. 495–526. DOI: 10.1007/978-3-030-45374-9_17.

[BS+18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast reed-solomon interactive oracle proofs of proximity". In: *International Colloquium on Automata, Languages, and Programming (ICALP)*. http://drops.dagstuhl.de/opus/volltexte/2018/9018. 2018.

[BS+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. "Aurora: Transparent succinct arguments for R1CS". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-030-17653-2_4. Springer. 2019, pp. 103–128.

[Ben86] Josh Cohen Benaloh. "Secret Sharing Homomorphisms: Keeping Shares of A Secret Sharing". In: *Advances in Cryptolog – CRYPTO*. Vol. 263. https://doi.org/10.1007/3-540-47721-7_19. Springer. 1986, pp. 251–260. DOI: 10.1007/3-540-47721-7_19.

[Ben+16] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. "Efficient zero-knowledge proofs for commitments from learning with errors over rings". In: *Computer Security – ESORICS*. https://doi.org/10.1007/978-3-319-24174-6_16. Springer. 2016, pp. 305–325.

[Beu20] Ward Beullens. "Sigma protocols for MQ, PKP and SIS, and fishy signature schemes". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-030-45727-3_7. Springer. 2020, pp. 183–211.

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning attacks against support vector machines". In: *ICML*. https://dl.acm.org/doi/10.5555/3042573.3042761. 2012.

[Bog+15] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. "How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation". In: *International Conference on Financial Cryptography and Data Security (FC)*. https://doi.org/10.1007/978-3-662-47854-7_14. Springer. 2015, pp. 227–234.

[BTW12] Dan Bogdanov, Riivo Talviste, and Jan Willemson. "Deploying secure multi-party computation for financial data analysis". In: *International Conference on Financial Cryptography and Data Security (FC)*. https://doi.org/10.1007/978-3-642-32946-3_5. Springer. 2012, pp. 57–64.

[Bog+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. "Secure multiparty computation goes live". In: *International Conference on Financial Cryptography and Data Security (FC)*. https://doi.org/10.1007/978-3-642-03549-4_20. Springer. 2009, pp. 325–343.

[Boi+21] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. "Flexible and Efficient Verifiable Computation on Encrypted Data". In: *Public-Key Cryptography – PKC*. https://doi.org/10.1007/978-3-030-75248-4_19. Springer. 2021, pp. 528–558.

[Bon+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-319-96884-1_19. Springer. 2018, pp. 565–596. DOI: 10.1007/978-3-319-96884-1_19.

[Boo+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-662-49896-5_12. Springer. 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.

[Boo+17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-319-70700-6_12. Springer. 2017, pp. 336–365. DOI: 10.1007/978-3-319-70700-6_12.

[Boo+21] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "More efficient amortization of exact zero-knowledge proofs for LWE". In: *Computer Security – ESORICS*. https://doi.org/10.1007/978-3-030-88428-4_30. Springer. 2021, pp. 608–627.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. "Algebraic Techniques for Short(er) Exact Lattice-Based Zero-Knowledge Proofs". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-030-26948-7_7. Springer. 2019, pp. 176–202. DOI: 10.1007/978-3-030-26948-7_7.

[Bos+20] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. "Efficient post-quantum SNARKs for RSIS and RLWE and their applications to privacy". In: *International Conference on Post-Quantum Cryptography (PQCrypto)*. https://doi.org/10.1007/978-3-030-44223-1_14. Springer. 2020, pp. 247–267.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In:

ACM Transactions on Computation Theory (TOCT) 6.3 (2014). https://dl.acm.org/doi/10.1145/2090236.2090262, pp. 1–36.

[BP16] Zvika Brakerski and Renen Perlman. "Lattice-Based Fully Dynamic Multi-key FHE with Short Ciphertexts". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-662-53018-4_8. Springer. 2016, pp. 190–213. DOI: 10.1007/978-3-662-53018-4_8.

[Bün+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short proofs for confidential transactions and more". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2018.00020. IEEE. 2018, pp. 315–334.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-030-45721-1_24. 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[CMP14] Dario Catalano, Antonio Marcedone, and Orazio Puglisi. "Authenticating computation on groups: New homomorphic primitives and applications". In: *Advances in Cryptology – ASIACRYPT*. https://dx.doi.org/10.1007/978-3-662-45608-8_11. Springer. 2014, pp. 193–212.

[Cha+22] Sylvain Chatel, Christian Knabenhans, Apostolos Pyrgelis, and Jean-Pierre Hubaux. "Verifiable Encodings for Secure Homomorphic Analytics". In: *arXiv preprint arXiv:2207.14071* (2022). https://arxiv.org/abs/2207.14071.

[Cha+21] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. "Privacy and Integrity Preserving Computations with CRISP". In: *USENIX Security Symposium*. https://www.usenix.org/conference/usenixsecurity21/presentation/chatel. 2021, pp. 2111–2128.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. "Multi-Key Homomorphic Encryption from TFHE". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-030-34621-8_16. Springer. 2019, pp. 446–472. DOI: 10.1007/978-3-030-34621-8_16.

[Che+19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference". In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/3319535.3363207. 2019, pp. 395–412.

[Che+22] Jeffrey Chen, Manaswitha Edupalli, Bonnie Berger, and Hyunghoon Cho. "Secure and federated linear mixed model association tests". In: *bioRxiv* (2022). https://doi.org/10.1101/2022.05.20.492837.

[CZW17] Long Chen, Zhenfeng Zhang, and Xueqing Wang. "Batched Multi-hop Multi-key FHE from Ring-LWE with Compact Ciphertext Extension". In: *Theory of Cryptography (TCC)*. https://doi.org/10.1007/978-3-319-70503-3_20. Springer. 2017, pp. 597–627. DOI: 10.1007/978-3-319-70503-3_20.

[Che+21] Weikeng Chen, Katerina Sotiraki, Ian Chang, Murat Kantarcioglu, and Raluca Ada Popa. "HOLMES: a platform for detecting malicious inputs in secure collaborative computation". In: *Cryptology ePrint Archive* (2021). https://eprint.iacr.org/2021/1517.

[CT14] Massimo Chenal and Qiang Tang. "On key recovery attacks against existing somewhat homomorphic encryption schemes". In: *Progress in Cryptology – LATINCRYPT*. https://doi.org/10.1007/978-3-319-16295-9_13. Springer. 2014, pp. 239–258.

[Che+18] Jung Hee Cheon, Kyoohyung Han, Seong-Min Hong, Hyoun Jin Kim, Junsoo Kim, Suseong Kim, Hosung Seo, Hyungbo Shim, and Yongsoo Song. "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption". In: *IEEE access* 6 (2018). https://ieeexplore.ieee.org/abstract/document/8325268, pp. 24325–24339.

[Che+17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-319-70694-8_15. Springer. 2017, pp. 409–437. DOI: 10.1007/978-3-319-70694-8_15.

[CGG16] Ilaria Chillotti, Nicolas Gama, and Louis Goubin. "Attacking FHE-based applications by software fault injections". In: *Cryptology ePrint Archive* (2016). https://ia.cr/2016/1164.

[Cho+22a] Hyunghoon Cho, David Froelicher, Jeffrey Chen, Manaswitha Edupalli, Apostolos Pyrgelis, Juan R. Troncoso-Pastoriza, Jean-Pierre Hubaux, and Bonnie Berger. "Secure and Federated Genome-Wide Association Studies for Biobank-Scale Datasets". In: *bioRxiv* (2022). eprint: https://www.biorxiv.org/content/early/2022/12/02/2022.11.30.518537.full.pdf. URL: https://www.biorxiv.org/content/early/2022/12/02/2022.11.30.518537.

[Cho+22b] Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. *Efficient Threshold FHE with Application to Real-Time Systems*. Cryptology ePrint Archive, Paper 2022/1625. https://eprint.iacr.org/2022/1625. 2022. URL: https://eprint.iacr.org/2022/1625.

[Cou+21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. "Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-030-77883-5_9. 2021.

[CP16] Eric Crockett and Chris Peikert. "Challenges for ring-LWE". In: *Cryptology ePrint Archive* (2016). https://ia.cr/2016/782.

[Dam+12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-642-32009-5_38. Springer. 2012, pp. 643–662. DOI: 10.1007/978-3-642-32009-5_38.

[DPLS18] Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. "Lattice-based group signatures and zero-knowledge proofs of automorphism stability". In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/3243734.3243852. 2018, pp. 574–591.

[DPLS19] Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. "Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts". In: *Public-Key Cryptography – PKC*. https://doi.org/10.1007/978-3-030-17253-4_12. 2019. DOI: 10.1007/978-3-030-17253-4_12.

[Duc+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "Crystals-dilithium: A lattice-based digital signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018). https://doi.org/10.13154/tches.v2018.i1.238-268.

[EPF21] EPFL-LDS. *Lattigo v2.2.0*. Online: http://github.com/ldsec/lattigo. July 2021.

[ENS20] Muhammed F Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-030-64834-3_9. Springer. 2020, pp. 259–288. DOI: 10.1007/978-3-030-64834-3_9.

[Esg+19a] Muhammed F Esgin, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. "Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-030-26948-7_5. Springer. 2019, pp. 115–146. DOI: 10.1007/978-3-030-26948-7_5.

[Esg+19b] Muhammed F Esgin, Ron Steinfeld, Amin Sakzad, Joseph K Liu, and Dongxi Liu. "Short Lattice-Based One-out-of-Many Proofs and Applications to Ring Signatures". In: *Applied Cryptography and Network Security (ACNS)*. https://doi.org/10.1007/978-3-030-21568-2_4. Springer. 2019, pp. 67–88.

[FV12] Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." In: *IACR Cryptol. ePrint Arch.* (2012). https://eprint.iacr.org/2012/144.

[Fei+21] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. "Security Vulnerabilities of SGX and Countermeasures: A Survey". In: *ACM*

Computing Surveys (CSUR) 54.6 (2021). https://doi.org/10.1145/3456631, pp. 1–36.

[Fer+21] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. "SAFE-Learn: secure aggregation for private federated learning". In: *IEEE Security and Privacy Workshops (SPW)*. https://doi.org/10.1109/SPW53761.2021.00017. IEEE. 2021, pp. 56–62.

[FS86] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology – CRYPTO*. Vol. 263. https://doi.org/10.1007/3-540-47721-7_12. Springer. 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. "Efficiently verifiable computation on encrypted data". In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. https://dl.acm.org/doi/10.1145/2660267.2660366. 2014, pp. 844–855.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. "Boosting Verifiable Computation on Encrypted Data". In: *Public-Key Cryptography – PKC*. https://doi.org/10.1007/978-3-030-45388-6_5. Springer. 2020, pp. 124–154. DOI: 10.1007/978-3-030-45388-6_5.

[Fro+21a] David Froelicher, Juan R Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. "Scalable privacy-preserving distributed learning". In: *Proceedings on Privacy Enhancing Technologies* 2021.2 (2021). https://doi.org/10.2478/popets-2021-0030, pp. 323–347.

[Fro+21b] David Froelicher, Juan R Troncoso-Pastoriza, Jean Louis Raisaro, Michel A Cuendet, Joao Sa Sousa, Hyunghoon Cho, Bonnie Berger, Jacques Fellay, and Jean-Pierre Hubaux. "Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption". In: *Nature communications* 12.1 (2021). https://doi.org/10.1038/s41467-021-25972-y, pp. 1–10.

[GNSV21] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. "Rinocchio: SNARKs for Ring Arithmetic". In: *Cryptology ePrint Archive, Report 2021/322* (2021). https://ia.cr/2021/322.

[Göt+12] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. "On the design of hardware building blocks for modern lattice-based encryption schemes". In: *Cryptographic Hardware and Embedded Systems – CHES*. https://doi.org/10.1007/978-3-642-33027-8_30. 2012.

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: *Advances in Cryptology – ASIACRYPT*. Vol. 7073. https://doi.org/10.1007/978-3-642-

25385-0_23. Springer, 2011, pp. 431–448. DOI: 10.1007/978-3-642-25385-0_23.

[HPS19] Shai Halevi, Yuriy Polyakov, and Victor Shoup. "An improved RNS variant of the BFV homomorphic encryption scheme". In: *Topics in Cryptology–CT-RSA*. https://doi.org/10.1007/978-3-030-12612-4_5. Springer. 2019, pp. 83–105.

[IBM21] IBM. *HELib v2.2.1*. Online: https://github.com/homenc/HElib. Oct. 2021.

[Jag+17] Karthik A Jagadeesh, David J Wu, Johannes A Birgmeier, Dan Boneh, and Gill Bejerano. "Deriving genomic diagnoses without revealing patient genomes". In: *Science* (2017). https://doi.org/10.1126/science.aam9710.

[Jag+18] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2018.00057. 2018.

[JY14] Chihong Joo and Aaram Yun. "Homomorphic authenticated encryption secure against chosen-ciphertext attackHomomorphic Authenticated Encryption Secure against Chosen-Ciphertext Attack". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-662-45608-8_10. Springer. 2014, pp. 173–192. DOI: 10.1007/978-3-662-45608-8_10.

[KZG10] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-642-17373-8_11. Springer. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. "Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems". In: *Advances in Cryptology – ASIACRYPT*. Vol. 5350. https://doi.org/10.1007/978-3-540-89255-7_23. Springer. 2008, pp. 372–389. DOI: 10.1007/978-3-540-89255-7_23.

[Kim+21] Andrey Kim, Maxim Deryabin, Jieun Eom, Rakyong Choi, Yongwoo Lee, Whan Ghang, and Donghoon Yoo. "General bootstrapping approach for RLWE-based homomorphic encryption". In: *Cryptology ePrint Archive* (2021). https://eprint.iacr.org/2021/691.

[KPZ21] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. "Revisiting Homomorphic Encryption Schemes for Finite Fields". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-030-92078-4_21. Springer. 2021, pp. 608–639.

[Kwa+21] Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. "A Unified Framework of Homomorphic Encryption for Multiple

Parties with Non-Interactive Setup". In: *Cryptology ePrint Archive* (2021). https://eprint.iacr.org/2021/1412.

[Lai+14] Junzuo Lai, Robert H Deng, HweeHwa Pang, and Jian Weng. "Verifiable computation on outsourced encrypted data". In: *Computer Security – ESORICS*. https://doi.org/10.1007/978-3-319-11203-9_16. Springer. 2014, pp. 273–291.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Designs, Codes and Cryptography* 75.3 (2015). https://doi.org/10.1007/s10623-014-9938-4, pp. 565–599.

[Li+19] Ningbo Li, Tanping Zhou, Xiaoyuan Yang, Yiliang Han, Wenchao Liu, and Guangsheng Tu. "Efficient multi-key FHE with short extended ciphertexts and directed decryption protocol". In: *IEEE Access* (2019). https://doi.org/10.1109/ACCESS.2019.2913943.

[LWX22] Shimin Li, Xin Wang, and Rui Xue. "Toward Both Privacy and Efficiency of Homomorphic MACs for Polynomial Functions and Its Applications". In: *The Computer Journal* 65.4 (2022). https://doi.org/10.1093/comjnl/bxab042, pp. 1020–1028.

[LWZ18] Shimin Li, Xin Wang, and Rui Zhang. "Privacy-Preserving Homomorphic MACs with Efficient Verification". In: *Web Services–ICWS*. https://doi.org/10.1007/978-3-319-94289-6_7. Springer. 2018, pp. 100–115.

[Lib+18] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. "Lattice-Based Zero-Knowledge Arguments for Integer Relations". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-319-96881-0_24. Springer. 2018, pp. 700–732. DOI: 10.1007/978-3-319-96881-0_24.

[Lib+13] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. "Linearly homomorphic structure-preserving signatures and their applications". In: *Advances in Cryptology – CRYPTO* (2013). https://doi.org/10.1007/978-3-642-40084-1_17.

[Lin+13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. "Improved Zero-knowledge Proofs of Knowledge for the ISIS Problem, and Applications". In: *Public-Key Cryptography – PKC. Proceedings 16*. https://doi.org/10.1007/978-3-642-36362-7_8. Springer. 2013, pp. 107–124. DOI: 10.1007/978-3-642-36362-7_8.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption". In: *Annual ACM symposium on Theory of computing (STOC)*. https://doi.org/10.1145/2213977.2214086. 2012, pp. 1219–1234.

[Lyu09] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures". In: *Advances in Cryptology – ASIACRYPT*. https://doi.org/10.1007/978-3-642-

10366-7_35. Springer. 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures without Trapdoors". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-642-29011-4_43. Springer. 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43.

[LN17] Vadim Lyubashevsky and Gregory Neven. "One-Shot Verifiable Encryption from Lattices". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-319-56620-7_11. Springer. 2017, pp. 293–323. DOI: 10.1007/978-3-319-56620-7_11.

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-031-15979-4_3. Springer. 2022, pp. 71–101.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "Practical lattice-based zero-knowledge proofs for integer relations". In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/3372297.3417894. 2020, pp. 1051–1070.

[LNS21a] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "Shorter lattice-based zero-knowledge proofs via one-time commitments". In: *Public-Key Cryptography – PKC*. https://doi.org/10.1007/978-3-030-75245-3_9. Springer. 2021, pp. 215–241.

[LNS21b] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-030-84245-1_21. Springer. 2021, pp. 611–640.

[Sea] *Microsoft SEAL (release 3.0)*. http://sealcrypto.org. Microsoft Research, Redmond, WA. Oct. 2018.

[MBH22] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. "An Efficient Threshold Access-Structure for RLWE-Based Multiparty Homomorphic Encryption". In: *Cryptology ePrint Archive* (2022). https://eprint.iacr.org/2022/780.

[Mou+21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. "Multiparty homomorphic encryption from ring-learning-with-errors". In: *Proceedings on Privacy Enhancing Technologies* 2021 (2021). https://doi.org/10.2478/popets-2021-0071, pp. 291–311.

[MW16] Pratyay Mukherjee and Daniel Wichs. "Two Round Multiparty Computation via Multi-key FHE". In: *Advances in Cryptology – EUROCRYPT*. https://doi.org/10.1007/978-3-662-49896-

5_26. Springer. 2016, pp. 735–763. DOI: 10.1007/978-3-662-49896-5_26.

[Nat+21] Deepika Natarajan, Andrew Loveless, Wei Dai, and Ronald Dreslinski. "CHEX-MIX: Combining homomorphic encryption with trusted execution environments for two-party oblivious inference in the cloud". In: *Cryptology ePrint Archive* (2021). https://ia.cr/2021/1603.

[Par21] Jeongeun Park. "Homomorphic encryption for multiple users with less communications". In: *IEEE Access* 9 (2021). https://doi.org/10.1109/ACCESS.2021.3117029, pp. 135915–135926.

[Par+13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. "Pinocchio: Nearly practical verifiable computation". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2013.47. 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

[PS16] Chris Peikert and Sina Shiehian. "Multi-key FHE from LWE, Revisited". In: *Theory of Cryptography (TCC)*. https://doi.org/10.1007/978-3-662-53644-5_9. Springer. 2016, pp. 217–238. DOI: 10.1007/978-3-662-53644-5_9.

[PRR17] Yuriy Polyakov, Kurt Rohloff, and Gerard W Ryan. "Palisade lattice cryptography library user manual". In: *Cybersecurity Research Center, New Jersey Institute of Technology (NJIT), Tech. Rep* (2017). https://palisade-crypto.org/documentation.

[Pol+23] Antigoni Polychroniadou, Gilad Asharov, Benjamin Diamond, Tucker Balch, Hans Buehler, Richard Hua, Suwen Gu, Greg Gimler, and Manuela Veloso. "Prime Match: A Privacy-Preserving Inventory Matching System". In: *Cryptology ePrint Archive* (2023). https://ia.cr/2023/400.

[PG12] Thomas Pöppelmann and Tim Güneysu. "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware". In: *Progress in Cryptology – LATINCRYPT*. https://doi.org/10.1007/978-3-642-33481-8_8. Springer. 2012.

[Rai+18] Jean Louis Raisaro, Juan Troncoso-Pastoriza, Mickaël Misbach, João Sá Sousa, Sylvain Pradervand, Edoardo Missiaglia, Olivier Michielin, Bryan Ford, and Jean-Pierre Hubaux. "MedCo: Enabling Secure and Privacy-Preserving Exploration of Distributed Clinical and Genomic Data". In: *IEEE/ACM transactions on computational biology and bioinformatics* (2018). https://doi.org/10.1109/TCBB.2018.2854776.

[Rat+23] M. Rathee, C. Shen, S. Wagh, and R. Ada Popa. "ELSA: Secure Aggregation for Federated Learning with Malicious Actors". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00090. 2023, pp. 1573–1591.

[Rot+22] Dragos Rotaru, Nigel P Smart, Titouan Tanguy, Frederik Vercauteren, and Tim Wood. "Actively Secure Setup for SPDZ". In: *Journal of*

Cryptology 35.1 (2022). https://doi.org/10.1007/s00145-021-09416-w, p. 5.

[Sav+22] Sinem Sav, Jean-Philippe Bossuat, Juan R Troncoso-Pastoriza, Manfred Claassen, and Jean-Pierre Hubaux. "Privacy-preserving federated neural network learning for disease-associated cell classification". In: *Patterns* 3.5 (2022). https://doi.org/10.1016/j.patter.2022.100487, p. 100487.

[Sav+21] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. "POSEIDON: Privacy-preserving federated neural network learning". In: *Annual Network And Distributed System Security Symposium (NDSS)* (2021). http://dx.doi.org/10.14722/ndss.2021.24119.

[Sch80] Jacob T Schwartz. "Fast probabilistic algorithms for verification of polynomial identities". In: *Journal of the ACM* 27.4 (1980). https://doi.org/10.1145/322217.322225, pp. 701–717.

[Ste93] Jacques Stern. "A New Identification Scheme Based on Syndrome Decoding". In: *Advances in Cryptology – CRYPTO*. Vol. 773. https://doi.org/10.1007/3-540-48329-2_2. Springer. 1993, pp. 13–21. DOI: 10.1007/3-540-48329-2_2.

[TPD16] Ngoc Hieu Tran, HweeHwa Pang, and Robert H Deng. "Efficient verifiable computation of linear and quadratic functions over encrypted data". In: *ACM on Asia Conference on Computer and Communications Security (Asia CCS)*. https://dl.acm.org/doi/abs/10.1145/2897845.2897892. 2016, pp. 605–616.

[VKH23] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. "Verifiable Fully Homomorphic Encryption". In: *arXiv preprint arXiv:2301.07041* (2023). https://arxiv.org/abs/2301.07041.

[Wah+18] Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. "Doubly-efficient zkSNARKs without trusted setup". In: *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2018.00060. 2018, pp. 926–943.

[Wol+12] David I Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. "Scalable anonymous group communication in the anytrust model". In: *5th European Workshop on System Security*. https://apps.dtic.mil/sti/pdfs/ADA602806.pdf. 2012.

[Xu+22] Guowen Xu, Xingshuo Han, Shengmin Xu, Tianwei Zhang, Hongwei Li, Xinyi Huang, and Robert H Deng. "Hercules: Boosting the Performance of Privacy-preserving Federated Learning". In: *IEEE Transactions on Dependable and Secure Computing* (2022). https://doi.org/10.1109/TDSC.2022.3218793.

[Xu+23] Guowen Xu, Guanlin Li, Shangwei Guo, Tianwei Zhang, and Hongwei Li. "Secure Decentralized Image Classification with Multiparty Homomorphic Encryption". In: *IEEE Transactions on Circuits and*

Systems for Video Technology (2023). https://doi.org/10.1109/TCSVT.2023.3234278.

[Yan+22] Meng Yang, Chuwen Zhang, Xiaoji Wang, Xingmin Liu, Shisen Li, Jianye Huang, Zhimin Feng, Xiaohui Sun, Fang Chen, Shuang Yang, et al. "TrustGWAS: A full-process workflow for encrypted GWAS using multi-key homomorphic encryption and pseudorandom number perturbation". In: *Cell Systems* (2022). https://doi.org/10.1016/j.cels.2022.08.001.

[Yan+19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. "Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications". In: *Advances in Cryptology – CRYPTO*. https://doi.org/10.1007/978-3-030-26948-7_6. Springer. 2019, pp. 147–175. DOI: 10.1007/978-3-030-26948-7_6.

[Yas+18] Satoshi Yasuda, Yoshihiro Koseki, Ryo Hiromasa, and Yutaka Kawai. "Multi-key homomorphic proxy re-encryption". In: *International Conference on Information Security*. https://doi.org/10.1007/978-3-319-99136-8_18. 2018, pp. 328–346.

[Yua+22] Minghao Yuan, Dongdong Wang, Feng Zhang, Shenqing Wang, Shan Ji, and Yongjun Ren. "An Examination of Multi-Key Fully Homomorphic Encryption and Its Applications". In: *Mathematics* (2022). https://doi.org/10.3390/math10244678.

Appendix

A Influence of the Number of RNS Sub-rings

Table 5 shows the effect of the number of RNS sub-rings composing $\mathcal{R}_q$ (*i.e.,* the number of levels) on the performance of Pelta. We observe that Pelta's runtimes increase linearly with the number of sub-rings.

Table 5: Pelta's performance for the local key-generation protocol (§4.4.1) and variable number of $\mathcal{R}_q$ sub-rings ($\log N = 13$).

# sub-rings	Setup(s)	Prover (s)	Verifier (s)	Proof (MB)
1	12.3	14.3.8	15.4	2.05
2	22.6	28.8	30.6	4.1
3	32.2	43.3	44.5	6.15

B Parameterization

We detail the different parameters used in our construction and present, in Table 6, their values. The degree of the commitment ring $\mathfrak{R}_q$ is d. We denote by k_{rep}

the repetition rate used in the proof (see [ALS20]). T denotes the honest prover bound of the challenge randomness (*i.e.,* $\mathbf{c}\vec{\mathbf{r}}$) and δ_1 the width of the uniform distribution for sampling masking values. M is the number of expected rejections and δ_H the root Hermite factor; for security reasons, we ensure $\delta_H < 1.0043$ following security estimation done in prior works [Esg+19b; Esg+19a]. $\log q_j$ corresponds to the number of bits of the FHE sub-ring modulus. κ and λ are respectively the MSIS and MLWE dimensions in the sub-ring $\mathfrak{R}_{q_j}$.

Table 6: Parametrization for the key generation (PN13).

$\log d$	κ	λ	T	k_{rep}	$\log \delta_1$	$\log q_j$	δ_H	M
7	8	17	2^7	4	25	54	1.0038	2.9
10	2	3	2^{10}	4	29	54	1.0027	1.75
13	1	1	2^{13}	4	32	54	1.0009	2.25

C Lattice-Based Proof

Here, we describe the proof construction for satisfiability of (i) a linear relation, (ii) with ternary coefficients, (iii) and a check of the approximate bound proof. Note that this protocol is a combination between the proof of knowledge of a ternary solution to a linear relation in $\mathbb{Z}_q$ by Esgin *et al.* [ENS20] and an approximate bound proof [BL17; BN20; LNS20]. Figure 4 presents the prover's operations while the verifier's are in Figure 5.

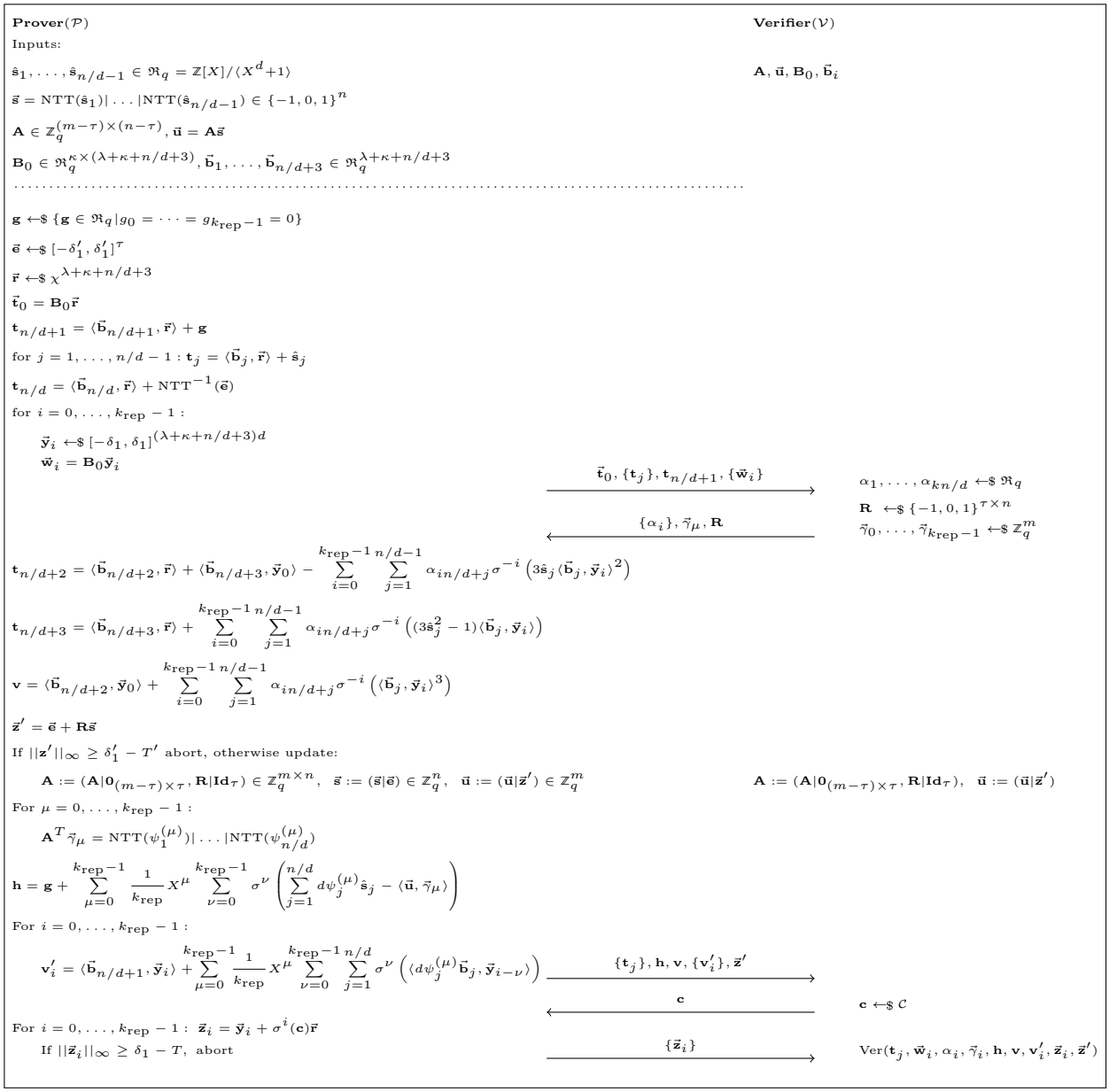

Fig. 4: Interactive proof generation of a ternary solution (of size n inputs in $\mathbb{Z}_q$) to an unstructured linear relation with additional approximate bound proof (ABP). For a polynomial ring $\mathfrak{R}_q$=$\mathbb{Z}_q[X]/\langle X^d+1\rangle$, N a power-of-two, κ and λ being respectively the MSIS and MLWE ranks, χ an error distribution in the MLWE problem, k_{rep} the repetition rate, δ_1 (resp. δ_1')the width of the distribution of the masks, T (resp. T') the bound of honest prover's $\mathbf{c}\vec{\mathbf{r}}$ of the linear proof (resp. for the ABP), and σ an automorphism of $\mathfrak{R}_q$ of order k_{rep}.

$\mathrm{Ver}(\mathbf{t}_j, \vec{\mathbf{w}}_i, \alpha_i, \vec{\gamma}_i, \mathbf{h}, \mathbf{v}, \mathbf{v}'_i, \vec{\mathbf{z}}_i, \vec{\mathbf{z}}')$

For $i = 0, \ldots, k_{\mathrm{rep}} - 1$:

1 : $\quad ||\vec{\mathbf{z}}_i||_\infty \overset{?}{<} \beta = \delta_1 - T$

2 : $\quad \mathbf{B}_0 \vec{\mathbf{z}}_i \overset{?}{=} \vec{\mathbf{w}}_i + \sigma^i(\mathbf{c})\vec{\mathbf{t}}_0$

For $i = 0, \ldots, k_{\mathrm{rep}} - 1$:

For $j = 1, \ldots, n/d$:

$\mathbf{f}_j^{(i)} = \langle \vec{\mathbf{b}}_j, \vec{\mathbf{z}}_i \rangle - \sigma^i(\mathbf{c})\mathbf{t}_j$

$\mathbf{f}_{n/d+2} = \langle \vec{\mathbf{b}}_{n/d+2}, \vec{\mathbf{z}}_0 \rangle - \mathbf{c} \cdot \mathbf{t}_{n/d+2}$

$\mathbf{f}_{n/d+3} = \langle \vec{\mathbf{b}}_{n/d+3}, \vec{\mathbf{z}}_0 \rangle - \mathbf{c} \cdot \mathbf{t}_{n/d+3}$

3 : $\quad \displaystyle\sum_{i=0}^{k_{\mathrm{rep}}-1} \sum_{j=1}^{n/d} \alpha_{in/N+j} \sigma^{-i} \left(\mathbf{f}_j^{(i)} \cdot (\mathbf{f}_j^{(i)} + \sigma^i(\mathbf{c})) \cdot (\mathbf{f}_j^{(i)} - \sigma^i(\mathbf{c})) \right) + \mathbf{f}_{n/d+2} + \mathbf{c}\mathbf{f}_{n/d+3} \overset{?}{=} \mathbf{v}$

4 : $\quad ||\mathbf{z}'||_\infty \overset{?}{<} q/2p$

For $\mu = 0, \ldots, k_{\mathrm{rep}} - 1$:

5 : $\quad h_\mu \overset{?}{=} 0$

$\mathbf{A}^T \vec{\gamma}_\mu = \mathrm{NTT}(\psi_1^{(\mu)}) | \ldots | \mathrm{NTT}(\psi_{n/d}^{(\mu)})$

$$\tau = \sum_{\mu=0}^{k_{\mathrm{rep}}-1} \frac{1}{k_{\mathrm{rep}}} X^\mu \sum_{\nu=0}^{k_{\mathrm{rep}}-1} \sigma^\nu \left(\sum_{j=1}^{n/d} d\psi_j^{(\mu)} \mathbf{t}_j - \langle \vec{\mathbf{u}}, \vec{\gamma}_\mu \rangle \right)$$

For $i = 0, \ldots, k_{\mathrm{rep}} - 1$:

6 : $\quad \displaystyle\sum_{\mu=0}^{k_{\mathrm{rep}}-1} \frac{1}{k_{\mathrm{rep}}} X^\mu \sum_{\nu=0}^{k_{\mathrm{rep}}-1} \sum_{j=1}^{n/d} \sigma^\nu \left(N\psi_j^{(\mu)} \langle \vec{\mathbf{b}}_j, \vec{\mathbf{z}}_{i-\nu \mod k_{\mathrm{rep}}} \rangle \right) + \langle \vec{\mathbf{b}}_{n/d+1}, \vec{\mathbf{z}}_i \rangle \overset{?}{=} \mathbf{v}'_i + \sigma^i(\mathbf{c})(\tau + \mathbf{t}_{n/d+1} - \mathbf{h})$

Fig. 5: Verification equations for Figure 4.