EPFL

Thèse n° 9845

Thwarting Malicious Adversaries in Homomorphic Encryption Pipelines

Présentée le 3 novembre 2023

Faculté informatique et communications
Laboratoire d'ingénierie de sécurité et privacy
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Sylvain CHATEL

Acceptée sur proposition du jury

Prof. C. Pit-Claudel, président du jury
Prof. C. González Troncoso, Prof. J.-P. Hubaux, directeurs de thèse
Dr J. Bootle, rapporteur
Dr P. Paillier, rapporteur
Prof. B. Ford, rapporteur

École polytechnique fédérale de Lausanne

2023

All we have to decide is what to do
with the time that is given us.
— J.R.R. Tolkien

To my family and my friends...

Acknowledgements

The existence of this manuscript has been made possible thanks to the support, counsel, guidance, and contributions of many collaborators, my friends, and my family. I would like to take the opportunity to attempt to convey my total appreciation for several people who have helped me throughout this journey.

First, I would like to thank my advisors Prof. Carmela Troncoso and Prof. Jean-Pierre Hubaux. I am extremely grateful to Jean-Pierre for having given me the opportunity to start my thesis in the Laboratory for Data Security, and for his guidance, advice, humor, and patience. His vision and experience helped me during my PhD research and I now have insight for my future endeavours. I would also like to immensely thank Carmela for her continuous support and for the key advice provided throughout this PhD experience. Her advice on research, career, and life will help me for years to come. I am particularly grateful for her welcoming us into the Security and Privacy Engineering Lab, in the last years of my PhD work. The friendly and supportive environment made those years such a special part of my life.

I would like to thank my committee members: Dr. Pascal Paillier, Dr. Jonathan Bootle, Prof. Bryan Ford, and Prof. Clément Pit-Claudel for finding the time to review my thesis and for their thoughtful discussions during the defense.

I also thank Angela Devenoge, Patricia Hjelt, and Isabelle Coke for their prompt and continuous help with administrative tasks. A special thanks go to Holly Cogliati-Bauereis for her editorial help in making our writing better and her eye-opening vision about hyphens.

I had the amazing opportunity to work with extraordinary collaborators. I would like to thank Christian Knabenhans and Ali Utkan Sahin for their input and help with the implementation of some of our work. I am also grateful to Dr. Juan R. Troncoso Pastoriza for his help and guidance through lattice-base cryptography.

A particular word goes out to my collaborator and friend Dr. Apostolos Pyrgelis. His advice, guidance, visions, support, patience, and fun were keys to the success of this thesis. In a few words, Σας ευχαριστώ πολύ! Δεν θα ήταν δυνατόν χωρίς εσάς. Η καθημερινή σας υποστήριξη, το μακροπρόθεσμο όραμα και οι συμβουλές σας με βοήθησαν να γίνω αυτό που είμαι και θα το θυμάμαι αυτό για πολλά χρόνια.

I am also extremely lucky to have been able to work alongside Sinem Sav and Christian Mouchet. Their constant support, friendship, and advice are beyond appreciated. I will deeply miss our coffee breaks, discussions, trips, and feasts. I hope we will still be able to

share many more of those.
I am also thankful for the members of LCA1/LDS, Spring, DEDIS, and LASEC whom I had the pleasure to meet throughout the years for making our floor such a vibrant environment. In particular, I would like to thank Prof. Wouter Lueks for his advice and passionate discussions about cryptography, food, and bread recipes; Theresa Stadler for the amazing discussions about research and life; Jean-Philippe Bossuat for his always prompt and insightful advice with Lattigo and views about FHE; Sandra Siby for showing us how to pack multiple days into one; Kasra EdalatNejad for his ever-lasting good mood; Bogdan kulynych for his humor; Simone Colombo for our fruitful coffee sessions; and Daniel Collins for being such a good mate.
This journey would have not been the same without some good friends made along the way. Dinners, outings, chilling, lake swimming, and endless discussions are just a small part of what made those five years even more enjoyable. A special thanks to the following people for having been such a big part of my life: Charlotte, Benoît, Juliette, Apo, Sinem, Ercu, Christian, Theresa, and all my friends from Chemin des Côtes. Our foody club took a noteworthy part in this journey as well. Thanks to Henry, Ludovic, and Kirill for being such good partners in debriefing about good food around the world. A special thanks goes to Quentin and Clara for their amazing friendship and continuous support throughout the years. Another special thanks goes to my friends and flatmates with whom I shared such a big part of my life: Etienne and Arnout. It was so fun to share this with you.
I would also like to thank my friends back home for their never-failing support.
Finally, I would like to thank my amazing parents Claire and Laurent; my extraordinary brother Rémy; and my exceptional partner Maria. Your unconditional love and support made all this possible.

Lausanne, October 2, 2023 S. C.

Abstract

Homomorphic Encryption (HE) enables computations to be executed directly on encrypted data. As such, it is an auspicious solution for protecting the confidentiality of sensitive data without impeding its usability. However, HE does not provide any guarantees that the cryptographic material used has been honestly generated and that the computation was executed correctly on the encrypted data. Thus, even though many practical systems rely on HE to achieve strong privacy guarantees, they consider only an honest-but-curious threat model in their constructions.

Although several efforts have been conducted to analyze and improve the security of HE-based systems against stronger threat models, these works have remained mostly theoretical and are still insufficient to be applicable to practical HE pipelines and real-life scenarios. Therefore, in our work, we propose and build solutions to protect HE pipelines against malicious adversaries and evaluate their performance over a wide range of use cases.

We first propose VERITAS, an efficient solution that proves the correctness of homomorphic computations, without compromising the expressiveness of the HE scheme. Then, we introduce PELTA, a set of building blocks that secure HE pipelines in the multiparty setting. Our constructions can be used to verify, in a practical manner, the correctness of distributed operations without any compromise on the HE scheme. Finally, we propose CRISP to secure input verification and to prove correct encryption in settings where the client who encrypts the data is untrusted.

All our constructions are a first step for evaluating the impact of the change of threat model in HE pipelines with real-life implementation constraints.

Keywords: Homomorphic Encryption, Secure Multiparty Computation, Malicious Adversary, Proof Systems.

Résumé

Le chiffrement homomorphe (HE) permet d'exécuter des calculs sur des données chiffrées sans avoir à les déchiffrer. Cette technique est donc prometteuse afin d'assurer la confidentialité des données tout en permettant leur potentielle utilisation lors d'analyses. Cependant, l'encryption homomorphe ne fournit aucune garantie que le matériel cryptographique utilisé ait été généré honnêtement ni que les calculs aient été exécutés correctement sur les données chiffrées. Ainsi, bien que de nombreuses constructions s'appuient sur ce type d'encryptions afin d'obtenir de solides garanties de confidentialité, elles considèrent uniquement un modèle de menace *honnête mais curieux*.
Les premiers efforts pour étudier et améliorer les garanties de sécurité sont restés principalement théoriques et sont encore insuffisants pour être applicables aux pipelines HE.
Ainsi, dans notre travail, nous cherchons à réduire cet écart en considérant des adversaires malveillants et nous construisons différents systèmes qui protègent le pipeline homomorphe contre de tels adversaires.
Nous proposons d'abord VERITAS, une solution efficace pour prouver l'exactitude des opérations homomorphes effectuées par un serveur sans compromettre l'expressivité du chiffrement homomorphe.
Ensuite, nous introduisons PELTA, un ensemble d'outils pour sécuriser les pipelines HE dans le cadre multipartite. Nos constructions peuvent être utilisées pour vérifier de manière efficace l'exactitude des opérations distribuées sans aucun compromis sur le schéma HE.
Enfin, nous proposons CRISP pour sécuriser la vérification des données d'entrée et le chiffrement correct dans des contextes où le client qui chiffre ses données n'est pas fiable.
Toutes nos constructions fournissent des éléments de base afin d'évaluer l'impact du changement de modèle de menace dans les pipelines homomorphes avec des contraintes d'implémentation réelles.

Mots-clés : Chiffrement Homomorphe, Calculs Multipartites Sécurisées, Adversaire Malveillant, Systèmes de Preuve.

Contents

Introduction

Homomorphic encryption (HE) schemes enable computations to be executed directly on ciphertexts, without decryption. They are an auspicious solution to protecting confidentiality during data processing. Contrary to their classic encryption counterparts, such as AES [DBN+01], HE schemes can protect data not only *in transit* and *at rest* but also during *computation*.

Since their introduction in the 80s, HE schemes have undergone several breakthroughs increasing their practicality. While the initial HE schemes supported only a single type of operations [ElG85, RSA78, Pai99], in 2009 Gentry introduced a new HE scheme supporting the evaluation of arbitrary circuits [Gen09b, Gen09a]. Such HE schemes that support the evaluation of *any* arithmetic circuit are called *fully homomorphic encryption* (FHE) schemes. Following Gentry's idea, numerous works have developed novel constructions and made HE schemes practical by supporting a wide range of operations and input data [CGGI20, CKKS17, BGV14, CHK+18b].

Nowadays, HE schemes have become functional and their implementation into several open-source libraries [EPF21, PRR17, SEA18], as well as the development of tools such as compilers [VJH21, Via23], facilitate their adoption. Overall, HE is an opportune solution for a plethora of applications where confidentiality of the data is paramount: *e.g.,* confidential computing and privacy-preserving analytics [Ama21, Goo20, Goo18, IBM21b, Mic21]. For instance, in federated learning [MMR+17, KMRR16], where a set of parties interact to build a collective machine learning model, HE enables the parties to execute the learning process fully under encryption thus avoiding any data leakage during the training [SPTP+21, SBTP+22, FTPP+21, LKS17, ZPGS19, HTGW18]. Similarly, machine learning inference can be conducted under encryption thus protecting the confidentiality of the client's data [GLN12, BGBE19]. In addition to machine learning, HE schemes find wide applicability in medical research [CJLL17, KL15, LYS15, RTPM+18], smart grids [LLL10, BCIV17, CLL22], and database queries [BGH+13, MW22, CGHK22].

In practice, HE enables a computing server to perform operations on encrypted data offloaded by one [CGBH+18, CJLL17, KL15] or more clients [MTPBH21, CZW17, CDKS19]. Using state-of-the-art HE schemes, an HE pipeline can be instantiated as a

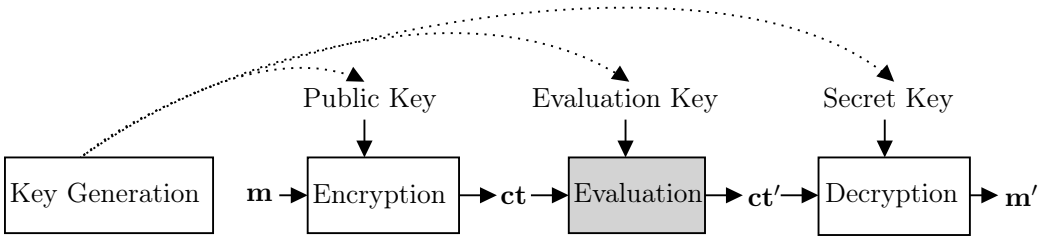


Figure 1: Illustration of an HE pipeline

cryptographic system operating in several steps as depicted in Figure 1. The first step consists in generating the cryptographic keys for each of the involved parties: secret keys for the data holders, public keys for data offloading, and public evaluation keys for the computing server. In the second step, the data holder encrypts the data. The third step is the evaluation, by a computing server, of a function over the encrypted data. While the computing server (represented in grey on the figure) typically performs the computation on its own, it can also be assisted by the parties using interactive protocols [MTPBH21, MBH22]. The final step is the decryption of the evaluation result using the secret key.

From a system model perspective, HE trivially supports computation delegation to a computing server but also enables efficient multiparty computation (MPC). In HE-based MPC, several parties collaborate to generate collective keys and to decrypt the final result. The encryption and evaluation can be executed by a single entity. Overall, HE-based MPC unites the benefits of both MPC and HE techniques (see Chapter 4).

Malicious Adversaries in HE Pipelines

Most of the practical applications of HE so far assume settings with honest-but-curious adversaries: *i.e.,* legitimate participants that do not deviate from the protocol but attempt to infer as much as possible from the information derived from the execution. However, as HE becomes practical enough to be deployed to real-life applications, the potential for attacks increases and this threat model assumption becomes less realistic. Firstly, HE schemes do not provide any guarantees about the *correctness* of the result. A malicious computing server can return a false result that destroys the utility of the process and can engage in key-recovery attacks that compromise confidentiality [CT14, CGG16] without being detected [VKH23]. Secondly, when considering HE-based multiparty computations, malicious clients can generate improperly formatted key material, create incorrect encryption of the data, and/or even alter the plaintext data, thus rendering unsafe and useless the result of the homomorphic computation. For example, in cloud-assisted machine-learning settings, a malicious cloud could insert a backdoor, when

training a model for a security-critical application, *e.g.,* malware detection, or return a wrongful prediction that could lead to a misdiagnosis in a medical application. Conversely, a malicious client could create improper cryptographic keys that would destroy the utility of the whole HE pipeline. Furthermore, a malicious client can disrupt the HE analytics pipeline by creating wrongly formatted encryption or by tampering with the data. This would leave useless the result of the overall pipeline.

Contribution of this thesis

Due to the aforementioned threats, it is necessary to create new constructions that can ensure the protection, in all the steps, of HE pipelines against malicious adversaries. Whilst numerous works have addressed some aspects of these problems, as discussed in Chapter 2, they lack the practicality, compatibility, and expressiveness to be entirely satisfactory. Therefore, in this dissertation, we provide the following technical contributions:

- **Chapter 3 – Verifiable Encodings for Secure Homomorphic Analytics.** In this chapter, we propose two error-detection encodings for homomorphic encryption messages; these encodings enable practical client-verification of cloud-based homomorphic computations without reducing the functionalities of the encryption scheme. We implement our solution in VERITAS and demonstrate its versatility and low overhead, on several use cases.

- **Chapter 4 – Securing HE-based Multiparty Computation against Malicious Adversaries.** In this chapter, we introduce PELTA, the first practical building blocks that enable the correctness verification of homomorphic multiparty operations under the malicious threat model. Our solution relies on a combination of lattice-based proof systems [ENS20, LNS20] adapted to support the peculiarities of lattice-based homomorphic encryption schemes and their implementations.

- **Chapter 5 – Verifiable Encryption and Trustworthy Data Release.** In this chapter, we present a generic approach to enable the authenticity verification of encrypted data offloaded to service providers for evaluation. By relying on lattice-based commitments and multi-party-computation in-the-head [IKOS09, GMO16, CDG$^+$17], our solution preserves the utility, security, and privacy that HE provides. We implement our solution, named CRISP , and demonstrate its performance via three use-cases.

Combined, the contributions of this thesis protect all steps of the HE pipelines: during the key generation, the encryption, the evaluation, and the decryption.

Summary

Overall, the high-level contributions presented in this dissertation are as follows:

- We systematize homomorphic encryption pipelines and identify the weak components vulnerable to a shift in the threat model from honest-but-curious participants to malicious adversaries (Chapter 1).
- We propose a practical solution for verifying, without compromising on the HE functionalities, the correctness of the HE computation (Chapter 3).
- We design an efficient construction for verifying the correctness of distributed operations in multiparty HE pipelines (Chapter 4).
- We introduce a solution for verifying the correctness, with respect to some authenticated inputs, of the HE encryption (Chapter 5).

Software. Our three technical constructions preserve the functionality of the HE scheme and do not compromise its parameterization. In addition, to increase the potential adoption of our contributions, we provide open-source prototype implementations of our solutions together with thorough benchmark evaluations. VERITAS is available at [LDS21]. The implementation of PELTA is available at [LDS23]. The code for CRISP can be found at [LDS20].

Bibliographic Notes

The contents of this thesis are based on the following publications:

- **Chapter 3:** S. Chatel, C. Knabenhans, A. Pyrgelis, C. Troncoso, and J.P. Hubaux, "Verifiable Encodings for Secure Homomorphic Analytics", *under submission*, arXiv preprint arXiv:2207.14071, 2023.
- **Chapter 4:** S. Chatel, C. Mouchet, A. U. Sahin, A. Pyrgelis, C. Troncoso, and J.P. Hubaux, "Multiparty-FHE against Malicious Adversaries", *to appear at CCS 2023*, Cryptology ePrint Archive, Paper 2023/642, 2023.
- **Chapter 5:** S. Chatel, A. Pyrgelis, J.R. Troncoso, and J.P. Hubaux, "Privacy and Integrity Preserving Computations with CRISP". In USENIX Security Symposium, 2021.

Other Contributions. In addition to the contributions outlined in this dissertation, we have also worked on other problems related to homomorphic encryption and privacy-preserving technologies.

Helium [MCP+23] Another contribution to the front of HE-based multiparty computation is Helium. This system provides the first end-to-end implementation of an HE-based multiparty computation among participants with limited computing and communication resources. By leveraging on an honest-but-curious *helper*, Helium supports multiparty computation with cost linear in the number of parties without non-collusion assumption. As this work was spear-headed by one of our co-authors, it is not discussed in this thesis.

SoK: Privacy-Preserving Collaborative Tree-based Model Learning [CPTPH21b] HE has been acting as the key cryptographic block in many scenarios and in particular in machine learning [SPTP+21, FTPP+21]. Among the wide range of machine learning approaches, tree-based models are one of the most efficient techniques for data mining nowadays due to their accuracy, interpretability, and simplicity. As such, we surveyed the literature on collaborative tree-based model learning from a privacy perspective considering starting from HE-based solutions, but eventually extending to additional privacy-enhancing techniques (multiparty computation, differential privacy, perturbation, and hardware solutions). We consolidated this survey by creating a systematization of knowledge based on four axes: the learning algorithm, the collaborative model, the protection mechanism, and the threat model. We use this to identify the strengths and limitations of these works and provide a framework for analyzing the information leakage occurring in distributed tree-based model learning. We find that tensions arise as the learning, distributed environment, and privacy protections introduce new constraints. Elegant and efficient solutions exist but often at the cost of some information leakage, and the few end-to-end protected solutions are not amenable to all scenarios. Therefore, we also provide a framework that identifies the information leakage occurring during the collaborative training of tree-based models. Our systematization enables us to identify limitations such as relaxed threat models and the lack of end-to-end confidentiality, and overall highlights avenues for future work.

Thesis Outline

In Chapter 1, we introduce the notations and the background on the different components used in this thesis. Specifically, we briefly recall the definition of homomorphic encryption, the different systems and threat models of HE pipelines, and the challenges HE creates. In Chapter 2, we present the relevant existing literature introducing protection mechanisms against malicious adversaries for HE pipelines. In Chapter 3, we show how to verify the correct execution of specified computation on homomorphically encrypted data. In Chapter 4, we introduce techniques that enable the correctness verification of homomorphic multiparty operations, under the malicious threat model. In Chapter 5, we sketch a solution for guaranteeing the correct encryption of authenticated data offloaded to a service provider. We conclude in Chapter 6.

Chapter 1

Background

In this chapter, we introduce the main building blocks used throughout the remaining chapters. We first present the background on homomorphic encryption (HE). Then, we expand on the different systems and threat models used with HE.

1.1 Notations

We define here the useful notation used throughout this dissertation.

Set, Integers, and Other Functions. Denote by $\mathbb{Z}_q$ the set of integers modulo q. Let $[n]$ refer to the set of integers $\{k; 0<k\leqslant n\}$ and $[0{:}n]$ the set $\{0\}\cup[n]$. For a prime t, denote $\mathbb{Z}_t$ its finite field and $\mathbb{Z}_t^*$ its non-zero elements. Let $\mathbb{R}$ be the field of the reals and $\mathbb{C}$ the one of complex numbers. For a complex number $z \in \mathbb{C}$, we denote by $\bar{z}$ its conjugate. For a set S, we note by $|S|$ its size. In the remainder of this thesis, we denote by log the logarithm base 2.

Vectors and Polynomials. A vector (resp. polynomial) be denoted by a boldface letter, *e.g.,* $\mathbf{x}$, with $\mathbf{x}[i]$ its i-th element (resp. coefficient) and $\mathbf{x}^T$ its transpose. Moreover, let $\|$ denote the concatenation operation. For two polynomials $\mathbf{a}$ and $\mathbf{b}$ we denote by $\mathbf{a} \cdot \mathbf{b}$ the polynomial multiplication. For two vectors $\mathbf{m}$ and $\mathbf{m'}$ we denote by $\mathbf{m} \circ \mathbf{m'}$ the Hadamard (*i.e.,* component-wise) multiplication. In some rare cases, a vector of scalars or polynomials will be denoted with an arrow to distinguish it from other polynomials.

Matrices. Let $\mathbf{I}_n$ be the identity matrix of size n, and $\mathbf{0}_k$ a vector of k zeros. We denote by $\text{diag}(\vec{p})$ the diagonal matrix made from the coefficients in $\vec{p}$.

Distributions and Probabilities. Let $a \leftarrow \chi$ denote that a is sampled from a

distribution χ. Unless otherwise explicitly stated, we consider the sampling uniform. We denote by $\Pr[X = x]$ the probability that the random variable X has value x.

Boolean Logic. In the remainder of this thesis, we consider that the value "*true*" is represented by the bit 1 and the value "*false*" by the bit 0.

1.2 Lattices

A full-rank n-dimensional *lattice* is defined as a discrete subgroup of $\mathbb{R}^n$. An element in the lattice can be represented from a basis of independent vectors $B = (\vec{b}_1, \ldots, \vec{b}_n) \in \mathbb{R}^n$ as a linear combination. As such,

$$L = \mathcal{L}(B) = \left\{ \sum_{i=1}^{n} c_i \vec{b}_i ; c_1, \ldots, c_n \in \mathbb{Z} \right\}.$$

In this dissertation, we work with ideal lattices: For $f : X \mapsto X^N + 1$ an integer monic irreducible polynomial of degree a large power-of-two N, we define by $\mathcal{R}$ the ring of integer polynomials modulo f (*i.e.,* $\mathcal{R} = \mathbb{Z}[X]/\langle X^N + 1 \rangle$). The ring $\mathcal{R}$ comprises integer polynomials elements of degree at most $N - 1$ whose coefficients can be seen as integer vectors in $\mathbb{Z}^N$. Let I be a subset of $\mathcal{R}$ closed under addition and multiplication in $\mathcal{R}$ – *i.e.,* an ideal of $\mathcal{R}$. Because I is additively closed in $\mathcal{R}$, the vectors corresponding to its polynomials' coefficients form a lattice. The term *ideal lattice* is employed to describe I, as it is both an *algebraic ideal* and a lattice [GH11]. The product of two ideals I and J is defined as the additive closure of the set $\{\mathbf{v} \cdot \mathbf{w} : \mathbf{v} \in I, \mathbf{w} \in J\}$. The ideal lattice generated by the polynomial $\mathbf{p} \in \mathcal{R}$ is $(\mathbf{p}) = \{\mathbf{p}_i = \mathbf{p} \cdot x^i \mod X^N + 1; i \in [0{:}n-1]\}$. We refer the reader to the work of Gentry for a more detailed introduction to lattices [GH11, Gen09a].

1.3 Fully Homomorphic Encryption

Homomorphic Encryption (HE) is a specific type of encryption that enables operations to be executed on ciphertexts and be reflected on the plaintexts directly. Examples of such schemes include RSA [RSA78], ElGamal [ElG85], and the Paillier [Pai99] cryptosystems. However, those schemes are homomorphic with respect to only a single type of operation. In 2009, Gentry proposed the first Fully HE (FHE) scheme able to support two types of operations (addition and multiplication) and, as such, can evaluate arbitrary arithmetic circuits [Gen09a, Gen09b, GH11]. Following Gentry's blueprint building on lattice-based problems, numerous constructions have been proposed: *e.g.,* BGV [BGV14, KPZ21], BFV [Bra12, FV12, HPS19], CKKS [CKKS17], and TFHE [CGGI20, CGGI16]. We refer

the reader to Marcolla *et al.*'s recent survey about HE for more details about FHE schemes since their debuts [MSM+22].

Overall, an FHE scheme can be formally defined as:

Definition 1 (FHE Scheme)**.** An FHE scheme $\mathcal{E}$ is a tuple of probabilistic polynomial time algorithms $\mathcal{E} = (\texttt{KGen}, \texttt{Enc}, \texttt{Eval}, \texttt{Dec})$ such that:

- $\texttt{KGen}(1^\lambda) \rightarrow \mathbf{sk}_{\text{HE}}, \mathbf{pk}_{\text{HE}}, \mathbf{evk}_{\text{HE}}$ is a randomized algorithm that, for a security parameter λ, generates a secret key $\mathbf{sk}_{\text{HE}}$, public key $\mathbf{pk}_{\text{HE}}$, and an evaluation key $\mathbf{evk}_{\text{HE}}$ for a security level λ.
- $\texttt{Enc}(\mathbf{m}; \mathbf{pk}_{\text{HE}}) \rightarrow \mathbf{c}$ is a randomized algorithm that returns an encryption of the plaintext polynomial $\mathbf{m} \in \mathcal{P}$.
- $\texttt{Eval}(F(\cdot), \mathbf{C}_{in}; \mathbf{evk}_{\text{HE}}) \rightarrow \mathbf{c}_{out}$ is an algorithm that, from a set of permitted circuits $\mathcal{C}$, evaluates the homomorphic evaluation of a circuit $F \in \mathcal{C}$ on the list of input ciphertexts $\mathbf{C}_{in}$ by using the evaluation keys $\mathbf{evk}_{\text{HE}}$ and that returns the encrypted result $\mathbf{c}_{out}$. The decryption of $\mathbf{c}_{out}$ returns the evaluation of $F(\cdot)$ on the plaintexts encrypted in $\mathbf{C}_{in}$.
- $\texttt{Dec}(\mathbf{c}; \mathbf{sk}) \rightarrow \mathbf{m}$ returns the decryption of the ciphertext $\mathbf{c}$ to a plaintext $\mathbf{m}$.

We now recall some key properties achieved by a correctly instantiated FHE scheme (see §2.1 [Gen09a]):

- **Encryption Correctness.** A homomorphic encryption scheme $\mathcal{E}$ is *correct* if for a key-pair $(\mathbf{sk}, \mathbf{pk}) \leftarrow \texttt{KGen}(1^\lambda)$, a plaintext $\mathbf{m} \in \mathcal{P}$, and an encryption $\mathbf{c} \leftarrow \texttt{Enc}(\mathbf{m}; \mathbf{r})$, then $\texttt{Dec}(\mathbf{c}; \mathbf{sk}) \rightarrow \mathbf{m}$.
- **Evaluation Correctness.** A homomorphic encryption scheme $\mathcal{E}$ is *correct for circuits in* $\mathcal{C}$ if for any key-pair $(\mathbf{sk}, \mathbf{pk}) \leftarrow \texttt{KGen}(1^\lambda)$ and evaluation keys $\mathbf{evk}_{\text{HE}}$, any plaintexts $\mathbf{m}_1, \ldots \mathbf{m}_n \in \mathcal{P}$, any ciphertexts $\mathbf{c}_1, \ldots \mathbf{c}_n$ such that $\mathbf{c}_i \leftarrow \texttt{Enc}(\mathbf{m}_i; \mathbf{r}_i)$, and any circuit $F \in \mathcal{C}$ it is the case that:

  $$\text{If } \mathbf{c}_{out} \leftarrow \texttt{Eval}(F, \{\mathbf{c}_1, \ldots \mathbf{c}_n\}, \mathbf{evk}_{\text{HE}}), \text{ then } \texttt{Dec}(\mathbf{c}; \mathbf{sk}) \rightarrow \mathbf{m} = F(\mathbf{m}_1, \ldots, \mathbf{m}_n)$$

  except with negligible probability over the random coins in $\texttt{Eval}$.
- **Compactness.** A homomorphic encryption scheme $\mathcal{E}$ is *compact* if there exists a polynomial f such that for any security parameter λ, for a set $\mathcal{C}$ of circuits of depth at most d, $\texttt{Dec}$ algorithm can be expressed as a circuit of size at most $f(\lambda, d)$ and the scheme is also correct for circuits in $\mathcal{C}$.
- **Semantic Security.** A homomorphic encryption scheme $\mathcal{E}$ is *semantically secure* if it is secure against chosen plaintext attacks (CPA security).

1.3.1 Overview of Lattice-Based FHE Schemes

Since 2009, various variants of Gentry's schemes have emerged. A first generation of schemes extended Gentry's original work [GH11, SV10] or based the security on the approximate-GCD problem [VDGHV10]. A second generation of schemes was introduced by Brakerski *et al.* [BV11a, BV11b, BGV14] then optimized [GHPS12, GHS12b, GHS12a, GHS12c, KPZ21] into the BGV scheme. Other variants include the BFV [FV12, HPS19] and CKKS [CKKS17] schemes increasing the practicality and efficiency. These schemes can evaluate circuits of limited depth and hence are called somewhat homomorphic encryption (SHE) schemes. A *bootstrapping* operation evaluates a decryption/re-encryption homomorphically and transforms the schemes into fully homomorphic encryption (FHE) schemes. This second generation of schemes is based on the learning with errors (LWE) [Reg05] problems or its variant over rings (RLWE) [LPR10]. We recall the RLWE distribution and the two seemingly hard associated computational problems [LPR10, Pei14]:

The RLWE distribution For a security parameter λ, let $f : X \mapsto X^N + 1$ where $N = N(\lambda)$ is a power of two. Let $q = q(\lambda) \geqslant 2$ be an integer. Let $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$ and let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let $\chi = \chi(\lambda)$ be a distribution over $\mathcal{R}$ of polynomials with coefficients sampled from a bounded discrete Gaussian distribution of small variance σ^2 and small bound B (w.r.t. q).

For a secret $\mathbf{s} \in \mathcal{R}_q$, a sample from the $\mathrm{RLWE}_{\lambda,\chi,\mathbf{s}}$ distribution is a pair of polynomials in $\mathcal{R}$ generated by choosing $\mathbf{a} \leftarrow \mathcal{R}_q$ uniformly at random, choosing $\mathbf{e} \leftarrow \chi$, and outputting $(\mathbf{a}, \mathbf{a} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$.

The search-RLWE problem The *search* RLWE problem is to discover the secret $\mathbf{s}$ given access to $m = \mathsf{poly}[n]$ independent samples $(a_i, b_i) \leftarrow \mathrm{RLWE}_{\lambda,\chi,\mathbf{s}}$.

The decision-RLWE problem The *decision* RLWE problem is to distinguish with non-negligible advantage between independent samples from $\mathrm{RLWE}_{\lambda,\chi,\mathbf{s}}$ and the same number of *uniformly random* independent samples from $\mathcal{R}_q^2$.

In practice, the BFV, BGV, and CKKS schemes work over a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^N+1\rangle$ where N is a power of two and q is a big integer (*e.g.,* $\log N = 13$ and $\log q = 218$). We recall a variant of the BFV [FV12, HPS19] scheme that is part of the FHE standardization effort [ACC+18].

The ciphertext space is the polynomial ring $\mathcal{R}_q$. The plaintext space is the polynomial ring $\mathcal{R}_t = \mathbb{Z}_t[X]/\langle X^N + 1\rangle$, where the modulus $t < q$. We denote by $\Delta = \lfloor q/t \rfloor$ a scaling factor. Let χ_k be a key distribution of ternary secrets over $\mathcal{R}_q$: *i.e.,* polynomials

$\mathcal{R}_q$ in with coefficients in $\{-1, 0, 1\}$ modulo q. Let χ_e be an error distribution over $\mathcal{R}_q$ with coefficients distributed according to a centered discretized Gaussian distribution of standard deviation σ_{err} and bounded in infinity norm by a parameter B_{err}. The main BFV algorithms are

BFV.KeyGen$(1^\lambda) \rightarrow \mathbf{sk}_{\text{HE}}, \mathbf{pk}_{\text{HE}}, \mathbf{evk}_{\text{HE}}$: Choose a low-norm secret key $\mathbf{s} \leftarrow \chi_k$ and set $\mathbf{sk}_{\text{HE}} := (1, \mathbf{s})$. The public key is defined as $\mathbf{pk}_{\text{HE}} := (\mathbf{b}, \mathbf{a})$ 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 $\mathbf{e} \leftarrow \chi_e$ sampled uniformly at random. For the evaluation, define an evaluation key $\mathbf{evk}_{\text{HE}}$ that comprises the *relinearization* and the *rotation keys*. The use of those keys will be made clearer below in Section 1.3.3 and details about their generation will be presented in Chapter 4. Output $\mathbf{sk}_{\text{HE}}$, $\mathbf{pk}_{\text{HE}}$, and $\mathbf{evk}_{\text{HE}}$.

BFV.Enc$(\mathbf{p}; \mathbf{pk}_{\text{HE}}) \rightarrow \mathbf{c}$: For a message $\vec{m} \in \mathbb{Z}_t^N$ encoded as a polynomial $\mathbf{p} \in \mathcal{R}_t$, sample $\mathbf{u} \leftarrow \chi_k$ and $\mathbf{e}'_0, \mathbf{e}'_1 \leftarrow \chi_e$. Output $\mathbf{c} := [\mathbf{u} \cdot \mathbf{pk}_{\text{HE}} + (\mathbf{e}'_0 + \lceil q/t \rfloor \cdot \mathbf{p}, \mathbf{e}'_1)]_q$. Thus, a fresh ciphertext comprises two polynomials in $\mathcal{R}_q$, *i.e.*, $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1)$.

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

Other schemes, such as BGV and CKKS, share a similar structure. BGV [BGV14, KPZ21] has a slightly different scaling and CKKS [CKKS17] enables approximate arithmetic computation. In the remainder of this dissertation, we focus mainly on BFV (chapters 3, 4) and CKKS (chapter 5) but our approaches could be generalized to the family of schemes described as above.

1.3.2 FHE Plaintext Space

The plaintext space $\mathcal{P}$ is, by construction, the polynomial ring $\mathcal{R}_t$. However, to enable more efficient and flexible circuit evaluations, a single instruction multiple data (SIMD) approach is favoured [SV14]. A plaintext encoding, named *batching*, transforms a plaintext vector $\vec{m} \in \mathbb{Z}_t^N$ into a polynomial plaintext $\mathbf{m} \in \mathcal{R}_t$. Due to the isomorphism of the encoder, the evaluation of the circuit on the polynomial representation ports to the vector representations:

$$\vec{m}_1 + \vec{m}_2 \leftrightarrow \mathbf{m}_1 + \mathbf{m}_2 \quad \text{and} \quad \vec{m}_1 \circ \vec{m}_2 \leftrightarrow \mathbf{m}_1 \cdot \mathbf{m}_2.$$

This transformation is obtained by relying on a number theoretic transformation (NTT) of the plaintext polynomial such that $\vec{m} = \text{NTT}(\mathbf{m})$. This transformation can be represented by a matrix-vector multiplication between the vector of coefficients of the polynomial and the Van Der Monde matrix that comprises $2N$-th primitive roots of unity [SV14, CHK$^+$18b]. The different components of the plaintext vector $\vec{m}$ are named *slots*. Henceforth, unless specified, we will interchangeably use the plaintext vector or plaintext polynomial representations.

1.3.3 FHE Operations

FHE schemes are *malleable* by design and can support the evaluation of specific circuits over encryption. Overall, the scheme described in the previous section (§1.3.1) can support a bounded number of additions and multiplications. An additional operation, called *bootstrapping*, enables the refreshing of the ciphertext – *i.e.,* to homomorphically re-encrypt the message with fresh noises. We briefly detail the different FHE operations below.

A homomorphic circuit is a bounded depth-directed acyclic graph of gates. The most common gates include:

- **BFV.Add**($\mathbf{c}$,$\hat{\mathbf{c}}$): A homomorphic addition simply adds the ciphertext vectors in $\mathcal{R}_q^2$. Given $\mathbf{c}$ and $\hat{\mathbf{c}}$, it outputs $[\mathbf{c}+\hat{\mathbf{c}}]_q$.

- **BFV.Mul**($\mathbf{c}, \hat{\mathbf{c}}; \mathbf{evk}_{\mathrm{HE}}$): The multiplication requires a relinearization procedure. Given two ciphertexts $\mathbf{c}=(\mathbf{c}_0, \mathbf{c}_1)$ and $\hat{\mathbf{c}}=(\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1)$, it does the following:

  1. **Tensoring and Rescale:** Computes $\mathbf{c}'_0:=\mathbf{c}_0\hat{\mathbf{c}}_0$, $\mathbf{c}'_1:=\mathbf{c}_0\hat{\mathbf{c}}_1+\mathbf{c}_1\hat{\mathbf{c}}_0$, and $\mathbf{c}'_2 := \mathbf{c}_1\hat{\mathbf{c}}_1 \in \mathcal{R}$ *without modular reduction.* Note that this represents the convolution between the two input ciphertexts *i.e.,* leading to an additional polynomial. Then it rescales each component to $\mathbf{c}^*_i:=[\lceil t/q \cdot \mathbf{c}'_i \rfloor]_q$, for $i \in \{0, 1, 2\}$.
  2. **Relinearization:** Uses the relinearization key stored in $\mathbf{evk}_{\mathrm{HE}}$ to convert the ciphertext $\mathbf{c}^*$ of three polynomials to a ciphertext comprising only two polynomials. This operation is not arithmetic and cannot be performed over $\mathcal{R}_q$ (see [HPS19]).

- **BFV.Rot**($\mathbf{c}, k; \mathbf{evk}_{\mathrm{HE}}$): A homomorphic rotation operates on a single ciphertext using the rotation key for k stored in $\mathbf{evk}_{\mathrm{HE}}$ and returns an encryption of the plaintext polynomial with shifted slots by a step k.

- **BFV.Bootstrap**($\mathbf{c}; \mathbf{evk}_{\mathrm{HE}}$): The bootstrapping operation refreshes the ciphertext by homomorphically decrypting and re-encrypting the ciphertext. The resulting ciphertext has a noise level similar to a fresh encryption of the plaintext (see [KDE+21] for more details).

In the following, we denote by **HE.Eval**($f(\cdot), (\mathbf{c}_1,...,\mathbf{c}_n); \mathbf{evk}_{\mathrm{HE}}$) the SIMD evaluation of the plaintext function $f(\cdot)$ on the ciphertexts $(\mathbf{c}_1,...,\mathbf{c}_n)$.

We note that both bootstrapping and key-switching cannot be represented by an arithmetic circuit in $\mathcal{R}_q$, because they require arithmetic over an extended ring or require rounding operations.

1.3.4 FHE in Practice

Due to their increasing practicality, several libraries have implemented BFV, BGV, and CKKS schemes [IBM21a, SEA18, EPF21, PRR17], thus augmenting their applicability. These implementations introduce additional optimizations, for the sake of efficiency.

- **RNS:** As, for correctness reasons, the polynomial ring modulus q is very big (*i.e.,* hundreds of bits) it is incompatible *as-is* with standard efficient CPU instructions. In order to alleviate this issue, the modulus is decomposed into a chain of moduli by using the Chinese remainder theorem (CRT). The modulus is chosen to be a composite number $q = \prod_{i=0}^{L} q_i$ with prime factors of smaller size (usually around $30 - 60$ bits). Any element in $x \in \mathbb{Z}_q$ can be decomposed into its CRT components $\{x_i = x \mod q_i \in \mathbb{Z}_{q_i}\}$. Operations in $\mathbb{Z}_q$ can be conducted in parallel on all the different CRT components over their respective sub-fields (*i.e.,* $\mathbb{Z}_{q_i}$). This decomposition is called the residue number system (RNS) and provides an execution faster than when handling big integer arithmetic [HPS19].

- **NTT:** For efficiency reasons, the prime factors of q are selected to support the number theoretic transform (NTT). The prime moduli need to satisfy $q_i = 1 \mod 2N$ such that the polynomial $X^N + 1$ splits fully into linear terms modulo each q_i (see [Sei18, CHK+19]). The NTT transform can be seen as a Fourier transform for polynomials. This enables efficient polynomial multiplication.

- **Montgomery:** Additional optimizations, such as the Montgomery transformation [Mon85], can be employed to enable fast modular arithmetic. It corresponds to a linear transformation of the polynomial coefficients.

1.4 FHE System and Threat Models

Homomorphic encryption, initially introduced by Rivest *et al.* as a *privacy homomorphism* [RAD+78] was originally considered to enable information systems, which store encrypted data offloaded by a client, to also be able to compute on the encrypted data. More recently, this system model was generalized to account for settings with multiple clients who offload their local data and who seek computations to be executed on their joint-data: *i.e.,* multiparty computation (MPC) [LATV12, CCS19, CDKS19, CZW17, PS16, BP16, Par21, MTPBH21, MBH22].

This covers more real-life applications, where data is not always centralized but can be distributed among different parties such as in distributed analytics [FTPR+21, CSS+22, YZW+22] and machine learning [CDKS19, SBTP+22, SPTP+21, ATP21, XHX+22,

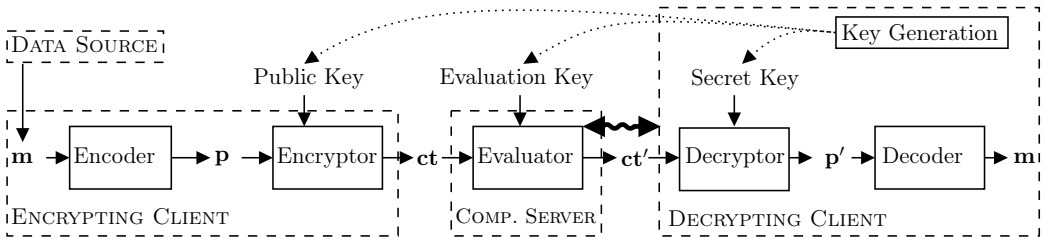


Figure 1.1: FHE pipeline with an encrypting client, a computing server, and a decrypting client.

XLG+23, AHWC19, AJLA+12, MTPBH21]. We will present, in more detail, the different approaches, in Section 4.2.

1.4.1 System Model

In the remainder of this thesis, we consider the following system model illustrated in Figure 1.1.

- *Clients*: One or more clients offload their data encrypted and subsequently want to have computations executed on them.
- *Computing server*: A server is tasked with performing computations requested by the client(s) on the offloaded data. It is supposed to return to a *decrypting client* a ciphertext holding the result of this computation.
- *Data source*: In some scenarios, the data is generated by a third party that we call the data source.

As we will see in Section 4.2, clients can interact with the computing server and assist during the FHE operation to accelerate the computations. This creates a trade-off between communication and computation costs but still remains realistic in real-life scenarios and more effective than classic MPC techniques [MTPBH21].

We define by *homomorphic pipeline* a process that comprises the FHE scheme setup, the data offloading, the homomorphic operations, and the decryption, as can be seen in Figure 1.1. Clients are involved in the FHE pipeline during the key generation, the encryption, the decryption, and in potential client-aided operations occurring during the homomorphic evaluation. The evaluation server potentially operates on the ciphertext and can ask the key owner for some assistance (*i.e.,* the curvy arrow), as we will see in Chapter 4.

1.4.2 Threat Model

Two key properties of FHE are the correctness and the semantic security [Gen09a] (see Section 1.3). Although all the different FHE schemes introduced in the literature ensure both of them, they are only guaranteed in the honest-but-curious threat model. In other words, the correctness of the homomorphic pipeline and the security against chosen-plaintext attacks are ensured against only passive adversaries that abide completely by the protocol.

This constraining assumption becomes less realistic to achieve when the number of involved parties in the FHE pipeline increases. The malleability-by-design of FHE creates a new potential vulnerability against malicious computing servers that might return erroneous ciphertexts to the client(s). In addition to destroying the application-level utility of FHE, these attacks could potentially be used in key-recovery attacks [CT14].

As a result, this honest-but-curious threat model becomes a shortcoming to creating practical applications of FHE, as it cannot account for realistic attacks, *e.g.,* the introduction of backdoors into distributed machine-learning [BVH+18], private-key recoveries [CT14], and selective censorship [GLL+20].

Thus, in this dissertation, we will propose solutions to ensure FHE pipelines' correctness and confidentiality in the presence of malicious but rational adversaries. Such adversaries, also known as covert in the literature, can actively cheat but do not want to be detected [AL07]. This adversarial model, initially introduced in the context of multiparty computation, discards arbitrarily malicious behavior (*e.g.,* denial of service, byzantine attacks) and is used to model realistic adversaries [AL07].

Chapter 2

Related Work

In this chapter, we present the state of the art in protection mechanisms against malicious adversaries for FHE pipelines. We begin by summarizing the literature on verifying the correctness of FHE evaluation in Section 2.1. Then, we look into the verification of client operations, in Section 2.2.

Our contributions in Chapters 3, 4, and 5 enable us to efficiently protect FHE pipelines during the set up, the encryption, the evaluation, and the collaborative phases. By design (and contrary to prior works), we work with state-of-the-art FHE implementations and optimizations, and we thoroughly evaluate the performance of our solutions.

2.1 Verifying FHE Evaluation

We first review the literature on the verification of FHE operations when offloading computations to an untrusted entity. The general concept of non-interactive verifiable computation (VC) was formalized by Gennaro *et al.* [GGP10], in the context of data offloading and computation outsourcing to an untrusted server. Applying this notion to FHE computations can help thwart malicious behavior from the computing server [VKH23]. However, the algebraic structure of FHE schemes and the required operations render the integration of VC with FHE difficult. We identify, from prior works, four major approaches: (i) generic MPC techniques, (ii) message authentication codes, (iii) proof systems, and (iv) trusted execution environments. We present each of them, with their strengths and limitations.

2.1.1 Multiparty Computation

Secure multiparty computation (MPC) techniques enable multiple entities to compute functions on their joint data, with strong confidentiality guarantees. Yao introduce the notion of garbled circuits [Yao86] that enable two parties to collaboratively evaluate a function on their private inputs. This protocol is the building stone of many MPC protocols. Both the original introduction of VC by Gennaro *et al.* [GGP10] and their multi-client variant rely on heavy MPC techniques (*e.g.,* Yao garbled circuits [Yao86], proxy oblivious transfer [NPS99]) combined with FHE. Other MPC techniques [DPSZ12] rely on *secret sharing* such as the one proposed by Shamir [Sha79]. Shamir's technique enables a client to divide a secret into *shares* that reveal nothing about the secret and such that the secret can only be reconstructed by re-combining at least a determined number of shares. Using computational techniques, such as the one introduced by Beaver [Bea91] for the multiplication of shared data, it is possible to construct an MPC protocol for securely computing arithmetic circuits.

Several efforts have been made to ensure the security of MPC techniques in the presence of malicious adversaries using cut-and-choose [Cha84], zero-knowledge proofs [GMW87], or authenticated secret-sharing [BDOZ11, DPSZ12]. Although MPC techniques are a generic technique for circuit evaluation, they induce a high computation and communication overhead. In more detail, the most common MPC protocols require linear communication in the size of the circuit (see [EKR$^+$18]). Their setup also often relies on a trusted third party to generate additional information.

Still, the complex structure of FHE makes compatibility with those approaches difficult. As seen in Chapter 1, FHE constructions use high-degree polynomials with very large coefficients. As such, secret sharing or other MPC techniques – whilst technically feasible – would remain highly impractical. Furthermore, MPC techniques are inherently interactive and not always aligned with the purpose of classic FHE in offloading scenarios.

2.1.2 Message Authentication Codes and Signatures

Homomorphic Message Authentication Codes (MAC) [GW13] are cryptographic primitives that enable anyone to compute over authenticated data and generate a short tag that authenticates the result. They can be viewed as a symmetric-key variant of fully homomorphic signatures [BF11]. Due to their homomorphic properties, they appear to be ideal primitives to be combined with FHE. In the literature, we identify three ways of doing so: Encrypt-and-MAC, Encrypt-then-MAC, and MAC-then-Encrypt. For completeness, we also briefly discuss the use of homomorphic signatures in the context of FHE.

2.1.2.1 Encrypt-and-MAC

In these works, the client encrypts her data and concurrently generates a homomorphic MAC of her plaintext data. She offloads both to the server for computation. Lai *et al.* [LDPW14] introduced first the concept of *homomorphic encrypted authenticators* that requires the MAC to not leak any information about the plaintext. As the choice of MAC and FHE are independent, their construction offers good flexibility. However, their initial construction suffered from poor verification efficiency. This was addressed by Li *et al.* [LWZ18, LWX22] building on a MAC proposed by Catalano *et al.* [CFW14]. But their improvement supports only the verifiable evaluation of bounded degree polynomials. However, to preserve confidentiality, the homomorphic authenticator needs to provide semantic security. As their constructions rely on the Diffie-Hellman problem and multilinear maps, they provide only a security that is weaker than the FHE scheme itself that is plausibly quantum-secure. Overall, these solutions work for only bounded degree polynomials, cannot cope with non-algebraic FHE operations (*e.g.,* relinearization, key-switching), and are only as secure as the underlying MAC.

2.1.2.2 Encrypt-then-MAC

In a different paradigm, Encrypt-then-MAC approaches consider directly generating a MAC for the ciphertext. Due to the semantic security of the FHE scheme, the MAC is no longer required to provide strong confidentiality properties. Catalano *et al.* [CMP14] authenticate linear ciphertext operations by using a previously introduced MAC [CF13]. By modifying this authenticator and relying on different security assumptions (*e.g.,* structure-preserving signatures [LPJY13], on the error-free approximate greatest common divisor problem [JY14], and on functional encryption [GKP$^+$13]), other works support quadratic functions and arithmetic circuit evaluations [TPD16, XHZ17]. Fiore *et al.* [FGP14] propose the first solution tailored to an RLWE-based scheme similar to BV [BV11a]: Ciphertexts are integrity-protected via a homomorphic MAC [BFR13] and a novel hashing technique compresses them to save storage and computational resources. Their instantiation, however, works for a simplified version of BV without relinearization and key-switching (thus, without rotations or SIMD) and supports only quadratic functions. In practice, this Encrypt-then-MAC approach was successfully employed in real-life scenarios: Cheon *et al.* [CHH$^+$18] provide a linear homomorphic authenticated-encryption scheme tailored for real-time drone systems, and Fiore *et al.*'s MAC [FGP14] was employed for verifiable federated learning [XLL$^+$19, HKKH21].

However, as we have seen in Section 1.3, the FHE ciphertexts are represented by high-degree polynomials with large coefficients. The ciphertext expansion induces a significant overhead that the MAC has to be able to cope with. Although works can rely on hashing techniques and polynomial compression, these tricks render impossible the evaluation of some FHE operations such as relinearization, bootstrapping, and key-switching.

2.1.2.3 MAC-then-Encrypt

This technique considers that the plaintext data's integrity is protected by embedding the homomorphic MAC in the plaintext space (*i.e.,* using the FHE encoder – see §1.3.2) before the encryption under FHE. If the homomorphic MAC and the FHE scheme support the same homomorphisms, then this approach guarantees that FHE modification of the plaintext ports directly to the MAC. Gennaro and Wichs pioneered this approach when they introduced their homomorphic MAC [GW13]. The use of FHE was mandated by the need to hide *challenge-values* from the server, and their MAC-then-Encrypt approach was a byproduct. More recently, Catalano and Fiore proposed a different approach that reduces the number of required challenges [CF13]. We will see in Chapter 3 how these MACs work in practice. The two limitations of these lines of work are that (i) the verifier needs to evaluate challenges and (ii) that the constructions are secure only without verification queries. The first point can easily be remedied by additional succinct non-interactive arguments (SNARG), whereas the second point can be acceptable in computation outsourcing scenarios, as we will see in Chapter 3. Contrary to the previous approaches, MAC-then-Encrypt supports any FHE operation and, hence, is far more practical than any other technique. We will discuss the trade-off that this technique induces in Chapters 3 and 6. Recently, Dolev and Kalma [DK21] proposed an encoding-based approach where a detection code is embedded in the low significant bits of the plaintext. However, to avoid overflowing, they are limited in the number of operations admitted.

2.1.2.4 Homomorphic Signatures

We note that Homomorphic signatures [BF11, CFN18, GVW15, ABC$^+$15, JMSW02, PST13, CFW14] are a public-key variant of homomorphic authenticators. Although they enable public verifiability, these constructions still remain constrained, cannot evaluate non-ring arithmetic operations of FHE, and are not practically embedded in the polynomial plaintext. Consequently, homomorphic signatures remain, at least for now, disjoint from FHE in the literature.

2.1.3 Zero-Knowledge Proofs

Initially introduced by Goldwasser, Micali, and Rackoff in the 1980s [GMR85], proof systems have become crucial building blocks for cryptographic constructions. Zero-knowledge arguments enable a computationally bounded prover to convince a verifier that an instance is in a language. A cheating prover has only a negligible probability of fooling the probabilistic verifier [BCC88, BCY91]. A stronger concept of zero-knowledge was captured by the notion of *zero-knowledge proof/arguments of knowledge* ensuring that the prover

knows at least one witness that the instance is in the *language* [GMR89]. Since then, considerable efforts have been made by the cryptographic community to increase their efficiency, practicality, and security. Recent improvements in the front of both *succinct non-interactive arguments* (SNARG) and *succinct non-interactive arguments of knowledge* (SNARK) [Lip12, Gro10, GGPR13, PHGR13, BSCG+13, Gro16, GKM+18] and other proof systems [BSCTV14, GKR15, BFH+20, BBB+18, COS20, CFH+15, KPS18, MBKM19, ZGK+17, Set20, VSBW13, WTS+18, WYKW21] render them ideal candidates for generic verifiable computations.

In the context of FHE, such SNARK techniques were recently considered to provide proof of correct computation on encrypted data. The server, in addition to computing on the input ciphertexts, also generates a proof of knowledge of the different wires in the homomorphic evaluation circuit leading to the claimed output. Fiore *et al.* [FNP20] proposed first to use commitments and SNARKs to achieve public verifiability for bounded polynomial operations over the ciphertexts. They reduce the overhead induced by the ciphertext expansion by relying on two blocks: (i) a commit-and-prove SNARK for arithmetic circuits and (ii) a commit-and-prove SNARK for multiple polynomial evaluations. Informally, their solution shows the correct relationship between the input and output polynomial ciphertexts of the homomorphic computation. A SNARK first shows the correct evaluation of the polynomials on a random point to obtain a compressed version of the polynomials and that each of the evaluations is committed to. For a second step, the correctness of the circuit evaluation is verified for the compressed polynomial with one SNARK; and a second SNARK ensures that the previous commitment opens to the evaluated polynomials on the random point. However, their solution does not support key ciphertext operations: *e.g.,* modulus switches and rounding operations. As a result, their solution cannot support several state-of-the-art schemes (*e.g.,* BFV and BGV). Furthermore, it requires the SNARKs and the FHE scheme to share the same cryptographic parameters, which can result in an unwanted overhead; the FHE modulus should match the SNARK's which is usually much larger. Although this latter constraint is lifted by Bois *et al.* [BCFK21], the resulting approach still limits the admissible FHE pipelines (as it does not support modular reduction). Their construction also relies on the GKR [GKR15] protocol that admits only *log-space uniform* and layered circuits with low depth for efficient verification. Hence, it can potentially limit the expressiveness of the FHE scheme. Ganesh *et al.* propose Rinocchio [GNSV21] as a new SNARK for rings that can be used for verifiable computations on encrypted data. Compared to [FNP20], Rinocchio supports common lattices used by FHE schemes. In particular, it supports generic polynomial rings R_q constructed from a composite modulus chain matching those used by FHE. But Rinocchio is still not as expressive as FHE and cannot practically support core operations involving non-ring arithmetic (*e.g.,* relinearization, key-switch, and bootstrapping). Their SNARK-based approach also still induces a non-negligible memory and computation cost for the prover, as we will see in Chapters 3. Although SNARK approaches benefit from succinct proofs and efficient verification procedures,

the prover still has a significant computational overhead. Whereas it is acceptable in a single client-cloud scenario, it becomes less realistic in multiparty settings. In Chapter 4, we will see that the prover overhead can become too constraining in practical instances. Overall, notwithstanding the tremendous efforts conducted to increase the practicality of generic SNARKs, the concrete structure of FHE renders interfacing SNARKs and FHE difficult and practically infeasible.

2.1.4 Trusted Hardware

A radically different approach to addressing VC is to rely on trusted execution environments (TEEs) [DL21]. To ensure computation integrity, this line of research employs trusted hardware, such as Intel SGX [AGJS13, HLP+13, MAB+13]. A TEE enables the isolation of the code running from the rest of the server. Through attestation techniques, it can also provide guarantees of the correct execution. Therefore, TEEs have been considered in numerous cases: secure two-party function evaluation [FKSW19], cloud-based machine-learning training [HWA21], MapReduce [SCF+15], database queries [PVC18], and search indices [MPC+18]. Natarajan *et al.* [NLDD21] were the first to employ TEEs combined with FHE for privacy-preserving machine learning between two clients and an entrusted computing server. More recently, Viand *et al.* [VKH23] proposed a new method combining some techniques from Rinocchio [GNSV21] with TEEs. They observe that, even though TEEs are more easily compatible with FHE, ciphertext size and FHE operations are still a bottleneck. As a result, their experimental evaluation is not considering relinearization (which is required for compact FHE multiplication) or bootstrapping (which is required for evaluating circuits of large multiplicative depth).

Additionally, TEEs rely on different trust assumptions, as they put the trust in the TEE manufacturer. TEEs also suffer from several vulnerabilities that range from side-channel privacy leakages [LKO+21, WCP+17, XCP15] to denial of service (DoS) by using the integrity-based DoS attacks [FYDX21, Ran21].

For these reasons, we consider the TEE approach orthogonal to our work. And it would be interesting to see what advances in this field could bring to FHE operation verification.

Overview of FHE Operation Verification and Contribution.
In this subsection, we have seen that the literature proposes different approaches for verifying the correctness of FHE operations. MPC protocols have high communication overhead, thus clashing with the compactness and non-interactivity benefits of FHE. Proof systems/arguments and TEE, though promising, are still limited in their expressiveness and not fully compatible with FHE operations and parameterizations. Similar observations are valid for most homomorphic authenticators in the Encrypt-then-MAC paradigm.
Therefore, in Chapter 3, we propose to follow the approach for verifiable computation sketched by Gennaro *et al.* [GW13] and Catalano and Fiore [CF13] and to design two new polynomial plaintext encoders. Our technique supports any FHE evaluation and is compatible off-the-shelf with trending integer-based FHE.

2.2 Verifying FHE Client's Operations

As discussed in Section 1.3, several parts of the FHE pipeline involve the client: *e.g.,* the key generation, the encryption, the client-aided evaluation, and the decryption. A client interacts with the FHE pipeline in different phases. At minima, to create a valid ciphertext, the client is in charge of generating the FHE key material and encrypting her data. In the *client-assisted* setting, a client can also be employed to speed up some computations such as bootstrapping (see Chapter 4 and [MTPBH21, MBH22]). Therefore, we identify three major operations to be verified: key generation, encryption, and online client-assisted operations. However, in practice, these three operations share very similar operations in lattice-based FHE. Indeed, simply put, they all generate a linear relation under constraint as we will see in Chapter 4 (§4.3.2). Hence, we recall the related literature on proving the knowledge of inputs to such constrained equations (§2.2.1). Then we survey another important aspect specific to encryption: input verification (§2.2.2).

2.2.1 Proving Correctness of Linear Relations

As discussed in the introduction, proving the correctness of the client's operations is paramount to securing FHE pipelines and particularly in HE-based MPC settings. As we will expand in Chapter 4, proving the FHE key generation operation boils down to proving exact knowledge of a short solution to a linear relation (and the same holds for the encryption and client-assisted operations). Several works dealt with proving the exact knowledge of a small-norm solution $\mathbf{s}$ to the linear equation $\mathbf{As} = \mathbf{u}$. The first protocols to prove such linear relation with a short norm secret [KTX08, LNSW13] involved the combinatorial approach from Stern [Ste93]. The main drawback of their approach is the large soundness error that forces many repetitions of the protocol to achieve acceptable

soundness error. Libert *et al.* [LLNW18] also employed a Stern-proof for lattices but this led to prohibitive proof sizes (see [LNS20]). More recently, to decrease the soundness error at the cost of higher run-times, Beullens [Beu20] combined it with a cut-and-choose approach. To verify the relation, Del Pino *et al.* [dPLS18] employed the 'Bulletproof' proof system [BCC$^+$16, BBB$^+$18]. In summary, their approach relies on the hardness of the discrete logarithm problem and, though offering short proofs, suffers from long running times (see [BLNS21]). In a different paradigm, Baum and Nof proposed a solution based on Multiparty Computation-in-the-head combined with cut-and-choose [BN20], thus leading to proofs at least an order of magnitude larger than commitment-based approaches [Beu20]. Boschini *et al.* [BCOS20] proposed a solution that uses the generic proof system Aurora [BSCR$^+$19]. Although it generates very short proofs, their system still imposes a major overhead for the prover. We will see in Chapter 4 that, for a small lattice ($\log N = 13$), it takes 845s and more than 80GB of RAM to generate a proof of a single RLWE sample over one single sub-ring, and that it takes an additional 312s to verify the proof. By relying on lattice-based commitments and their corresponding zero-knowledge proofs, several protocols were able to prove the correctness of the linear relation with low-norm secret [BLS19, YAZ$^+$19, ESLL19a, ESS$^+$19]. However, these initial works had prohibitive costs, required specific polynomial rings not compatible with trending FHE, and considered only simple linear relations. Improvements on the front of the commitment scheme were made by Attema *et al.* [ALS20], Esgin *et al.* [ENS20], and Lyubashevsky *et al.* [LNS21b, LNS20, LNS21a, LNP22] to provide more efficient product proof, to support NTT optimizations, and to reduce the proof size. In a way very similar to our construction, Yang *et al.* [YAZ$^+$19], and Beullens [Beu20] rely on the BDLOP commitment scheme [BDL$^+$18a] and its improvements, but this still requires a specific structure for the polynomial ring that is not compatible with current FHE implementations. The work of Bootle *et al.* [BLS19] works in the NTT domain and was improved by Attema *et al.* [ALS20], Esgin *et al.* [ENS20], and Lyubashevsky *et al.* [LNS20] to obtain more efficient proofs.

Overview and Contribution.
This line of thinking paved the way for proofs that can be combined with FHE efficiently. In Chapter 4, we show how we can extend these works to prove statements pertaining to HE-based MPC. By combining these works, we achieve a practical solution that outperforms approaches with generic proof systems.

2.2.2 Input Verification

As we have seen in Section 1.4.1, FHE can often be employed in scenarios where multiple clients provide data. This can, for instance, be the case where clients collaborate to obtain the result of the computation on their joint data *e.g.,* in federated learning [SPTP$^+$21] but also in crowdsourcing scenarios where clients offload their data encrypted and an

additional entity seeks to obtain the result of the computation [Int19a, San19]. However, by design, FHE ciphertexts cannot provide any guarantees about the plaintext inputs (due to semantic security and malleability). This problem, also present in MPC, has forced researchers to assume that all inputting clients provide properly-formed and honest inputs [ZPGS19, CKR+20, PKY+21]. Homomorphic signatures *e.g.,* [ABC+15] are, in theory, a solution to this problem. The data source would, before handing it over to the users, authenticate the data by using such signatures. However, the way to practically homomorphically evaluate the encryption circuit using them is still under investigation. Furthermore, it would imply that the signer uses such signature schemes that are not yet widely deployed.

Three lines of work have looked into this specific problem. The first one considers binary or ternary plaintext space, whereas the second one checks that the plaintext data satisfies some properties. The final line of research relies on external authentication.

Verifiable Encryption. The first category of works solved the problem of verifiable encryption for FHE. Although they prove the correct encryption, these works also prove that the plaintext is small (*i.e.,* binary [BCOS20], ternary [ENS20], or with small coefficients [LN17]). However, it is far from the reality of FHE applications with plaintexts that are potentially bigger than 16 bits.

Statistical Tests. In the second category, Chen *et al.* [CSC+23] recently proposed Holmes, a set of techniques built into a platform for executing a wide range of statistical tests on inputs of MPC protocols. Their contribution is use-case agnostic and can be seen as a toolbox to verify inputs. It combines zero-knowledge proof with MPC protocols.

External Authentication. Finally, the third category considers that a third party has generated and authenticated the inputs. In Chapter 5, we call this entity a *data source*. Backes *et al.* [BBFR15] introduced this problem as having a client proving computations on authenticated data to a third party oblivious of the data. Their solution, named ADSNARK, relies on SNARKs [BSCTV14] and directly embeds the authentication verification in the proof system. However, ADSNARK does not support the feature of data offloading and every new computation on the data requires a new proof. And, by relying on SNARKs [PHGR13, BSCTV14] their approach suffers from the need for a trusted setup and relies on different cryptographic assumptions. Finally, as with any other SNARK approach, it is unclear how feasible the proof system will cope with the large dimensionality of trending FHE based on polynomial lattices without affecting the proving time. Consequently, because ADSNARK does not provide circuit privacy, the client cannot prove the correct encryption of the authenticated data.

Overview and Contributions.
Due to the limitations of prior works when combined with FHE, in Chapter 5, we present a solution based on multiparty computation in the head to generate concurrently a proof of correct FHE encryption of authenticated inputs.

In the following chapters, we propose three different constructions to address the challenges unaddressed by prior works. In Chapter 3, we explore how to practically verify the computing server's evaluation. In Chapter 4 we propose a construction to efficiently verify the client's operations. Finally, in Chapter 5, we introduce a solution to verify the correct encryption of authenticated inputs. Overall, our contributions enable us to efficiently protect FHE pipelines without compromising on the FHE scheme and optimizations, and we thoroughly evaluate the performance of our solutions.

Chapter 3

Verifiable Encodings for Secure Homomorphic Analytics

Homomorphic encryption has become a practical solution for protecting the privacy of computations on sensitive data. However, existing homomorphic encryption pipelines do not guarantee the *correctness* of the computation result in the presence of a malicious adversary. We propose two encodings compatible with state-of-the-art fully homomorphic encryption schemes that enable practical client-verification of homomorphic computations, while enabling all the operations required for modern privacy-preserving analytics. Based on these encodings, we introduce VERITAS, a ready-to-use library for the verification of computations executed over encrypted data. VERITAS is the first library that supports the verification of *any* homomorphic operation. We demonstrate its practicality for various applications such as ride-hailing, genomic-data analysis, encrypted search, and machine-learning training and inference.

Contents

3.1 Overview

With HE gaining in popularity, it becomes crucial to ensure its security against malicious actors and, in particular, during HE evaluations. However, existing HE schemes [FV12, HPS19, BGV14, KPZ21] do not provide clients with any guarantees about the *correctness* of the computations performed by the computing server. Thus, a malicious adversary can break the security and privacy of the HE computation without being detected [CMS+23, VKH23]. For instance, clients can be fooled into accepting a wrongful computation result and potentially leak unintended information from the decryption [CT14]. In current HE applications, the lack of computational integrity can lead to catastrophic consequences. For example, in medical applications, an adversary inducing a wrongful prediction might cause a misdiagnosis, and in machine learning an adversary able to inject backdoors during training can create vulnerabilities when the model is deployed [BS21, BR18, BNL12].

A trivial way for clients to check the correctness of the computation would be to recompute the result in plaintext. However, this is not always feasible, *e.g.,* in multi-client scenarios where not all the input data is available to the clients. To address this problem, researchers have proposed generic verifiable computation techniques, *e.g.,* [BFH+20, BBB+18, GKR15, GGP10, PHGR13, WYKW21], to check the integrity of server computations. These techniques, however, are hard to integrate with lattice-based HE. This is because the efficiency gains that make modern HE practical (*e.g.,* for medical research [CJLL17, KL15, LYS15] and ML [BGBE19, CGBH+18, GLN12, LKS17, SPTP+21, SBTP+22]) stem from the use of (i) specific polynomial constructions and algebraic parameters, and (ii) non-linear operations (*e.g.,* relinearization and rotation) and parallel processing over

Table 3.1: Comparison between VERITAS and prior work w.r.t. the supported HE operations (linear, multiplicative, rotation, relinearization, bootstrapping) and parameterization.

Scheme	Linear	Mult. depth	Relin.	Rot.	Bootstrap.	Flex. params.
[FGP14]	✓	1	✗	✗	✗	✗
[FNP20]	✓	any	✗	✗	✗	✗
[BCFK21, GNSV21]	✓	any	✗	✗	✗	✓
Ours	✓	any	✓	✓	✓	✓

encrypted data (Single Input, Multiple Data; SIMD). Generic verification techniques cannot cope with all these constraints efficiently.

To date, only a handful of verification works are tailored to homomorphically encrypted data. Fiore *et al.* [FGP14] pioneered a solution based on homomorphic Message Authentication Codes (MACs) to verify encrypted computations. Their work focuses only on verifying quadratic functions over a constrained version of the BV scheme [BV11b]. Thus, it cannot support flexible parameterization (*e.g.,* large-degree polynomials) and complex operations (*e.g.,* rotations) required to implement modern HE-based applications. Subsequent works rely on generating a proof-of-correct computation using succinct non-interactive arguments of knowledge (SNARKs) [FNP20, BCFK21, GNSV21]. But, due to incompatibilities of SNARKs with the algebraic structure of recent HE schemes and their non-algebraic operations (*e.g.,* rounding) these works also do not support variable HE parameters [FNP20] and core HE operations [BCFK21, GNSV21] such as relinearization, rotation, and bootstrapping. Without support for flexible HE parameters and critical HE operations, these works have very limited application in practice.

Our Contribution. We provide in this chapter the first practical solution that enables the verifiability of **all** the operations supported by state-of-the-art lattice-based HE schemes over the integers (see Table 3.1). Therefore, our solution supports any existing privacy-preserving application that employs such HE schemes and can protect its computation integrity against a malicious computing server.

The key idea behind our solution is to shift the verification from the ciphertext domain, where the constraints imposed by the algebraic structure of HE ciphertexts limit the practicality of previous solutions, to the plaintext space. To this end, we design two new plaintext HE-encoders that, combined with HE schemes, instantiate homomorphic authenticators [CF13, CKV10, GW13], *i.e.,* efficient constructions to verify homomorphically-executed computations. These authenticators permit the verification of the computations by checking a set of challenge values that can be pre-computed without access to the original input data (*i.e.,* this pre-computation can be assigned to any entity that does not collude with the computing server). This makes our solution suitable for multi-party scenarios [CZW17, CDKS19, MTPBH21] and computationally constrained

clients. Our first encoding is based on *replication* and takes advantage of the batching encoder supported by modern HE to introduce error-detection and redundancy in the data. The second one, achieves more compact authentications and requires less challenge values to be verified by encoding the data using a polynomial information-theoretic MAC. The two encoders achieve different tradeoffs depending on the volume of input data and the depth of the computation, hence, they can support various application settings.

Using our novel encodings, we design VERITAS, an open-source library that facilitates the conversion of existing HE computing pipelines, that assume an honest-but-curious computing server, into pipelines that can resist a malicious-but-rational server [AL07]. We benchmark VERITAS on native HE operations and evaluate its performance on five use cases: ride-hailing, genomic-data analysis, encrypted search, and machine-learning training and inference, where computation integrity is crucial. Our results demonstrate that VERITAS provides verification capabilities that were out of reach using prior work, at minimal costs for the client and the server. For instance, our solution is the first to practically enable the verifiability of a two-layer neural network evaluation for image recognition, with less than 1× computation overhead for the client and server, compared to the HE baseline. It also enables the verifiability of a disease prediction result on genomic data with less than 3×. In both cases, the communication overhead is at most 2×.

In summary, our contributions are the following:

- The design of two error-detecting HE encodings that support all the operations enabled by efficient HE schemes over integer plaintexts. We provide an in-depth study of their respective utility tradeoffs.
- Several optimizations that reduce the communication and computation overhead induced by our encodings. We present, in particular, a new communication-efficient *polynomial compression protocol* and a client-aided *re-quadratization* technique that reduces both the server's computation overhead and the overall communication overhead.
- The implementation of VERITAS, a library that clients can use to detect with high probability a malicious server attacking the HE pipeline. VERITAS is the first open-source and ready-to-use library that enables off-the-shelf secure homomorphic analytics under malicious-but-rational adversaries. We evaluate VERITAS' performance on five use-cases and show that it outperforms the state of the art [FGP14, GNSV21].

3.2 Problem Statement

In this section, we present our system and threat models and the desired objectives, before presenting an overview of our solution.

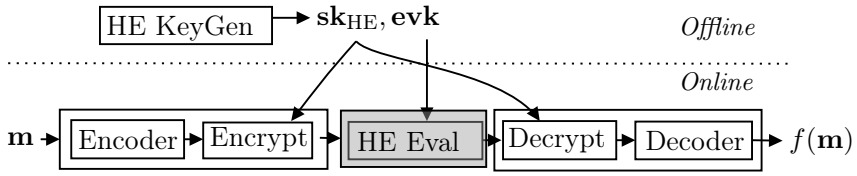


Figure 3.1: HE pipeline. It returns the homomorphic evaluation of a function $f(\cdot)$ over the encryption of a message $\mathbf{m}$.

System & Threat Model. We consider an HE-based computation scenario where one or more clients desire to perform computations on their sensitive data, which they encrypt with an HE scheme. The clients employ a computing server that is responsible for carrying out the computations; these range from the entirety of the computation in the case of a single-client [CGBH$^+$18, CJLL17, KL15], to aiding in the homomorphic evaluation for multiparty scenarios [CZW17, CDKS19, MTPBH21, CZW17, CDKS19]. We consider a malicious-but-rational server [AL07] (*i.e.,* covert) that tampers with the computations only if it has a high probability of not being detected. We assume that the clients and the server are authenticated to each other. We do not consider network faults in the communication.

Objectives. The client wants (i) to guarantee the privacy of its input data and the computation output vis-à-vis the server. In particular, the server should learn nothing about the input data and the computation result. The client also wants (ii) to ensure the correctness of the result with respect to the agreed-upon computation and inputs: the client must be able to detect a cheating server with probability at least $1 - 2^{-\lambda}$ for a security parameter λ.

Solution Overview. To protect data privacy during outsourcing and the subsequent server computation (*i.e.,* Objective (i)), the client uses state-of-the-art lattice-based HE to encrypt it (§3.3.1). This protects also the privacy of the output, as long as the decryption key remains secret on the client side. To ensure the correctness of the server's computation on the outsourced data (*i.e.,* Objective (ii)), we embed the verification of the computations into the plaintext space. During outsourcing, the client encodes its data using an error-detection encoder, encrypts and sends it (see Figure 3.2). We design two encoders with error-detecting capabilities that respect the homomorphic operations in the plaintext space. Combined with the HE scheme, they emulate homomorphic authenticators (§3.3.3) that enable client-based verification capabilities and act in lieu of the classic HE pipeline (by adapting the HE evaluation and decryption to the new plaintext encoders – see Figure 3.1 vs. Figure 3.2). After the server computation, the client decrypts and verifies the integrity of the encodings, thus vouching for the computation correctness with high probability. The first encoder, which is based on replication, mixes replicas of the data with challenge values in an extended vector (see Figure 3.3) that is encrypted by the client. After the server computation, the client can check its correctness by decrypting and

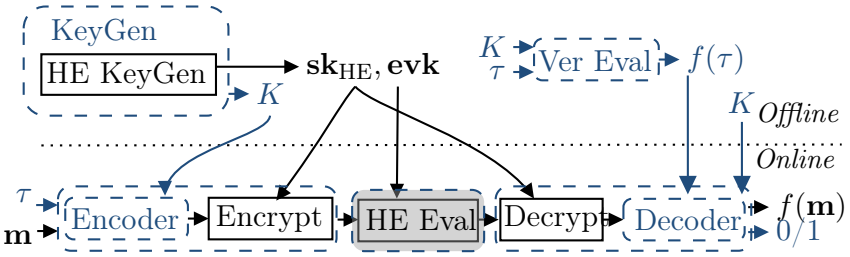


Figure 3.2: Enhanced HE pipeline. The square boxes correspond to the original HE pipeline (see §3.3.1 and Fig. 3.1) and the dotted boxes are the new components offering verification capabilities. The grey box represents the computing server. For a message $\mathbf{m}$ associated with a label τ and for a function $f(\cdot)$, the verification pipeline checks if the Eval. step performed by the computing server was executed correctly. $\mathbf{sk}_{\mathrm{HE}}$ and $\mathbf{evk}$ are the HE secret key and evaluation key respectively. K is the encoder's secret key.

inspecting the resulting extended vector. This encoder is detailed in §3.4. The second one, which is based on a polynomial encoding, encodes the message as a bivariate polynomial (see Figure 3.4) that is encrypted by the client. After the server computation, the client verifies its correctness by decrypting and evaluating the resulting polynomial on a secret point. This is presented in §3.5. The two encodings support any HE operation and prevent malicious servers from tampering with the requested computations undetected while achieving different efficiency trade-offs that we analyze in §3.6.3.

3.3 Preliminaries

We introduce some key components used in this work. We refer the reader to Chapter 1 for a description of our notation and the background on FHE.

3.3.1 Homomorphic Encryption Encoders

In this section, we work with FHE schemes that support modular arithmetic in a field $\mathbb{Z}_t$, *e.g.,* the BGV and BFV schemes. These schemes share the same plaintext algebra and differ only in the plaintext data-representation and encryption-noise management (see [KPZ21]). In the remainder of this section, we will work with the BFV scheme we described in Chapter 1.

HE Encoders. As shown in Figure 3.1, a plaintext encoder converts a vector of scalar values into a polynomial element in $\mathcal{R}_t$. Several encoders exist: *e.g.,* scalar, integer, and fractional ones (see [Lai17, §7]). In this work, we employ an encoder called *batching* that enables Single Instruction, Multiple Data (SIMD) operations. It converts a vector in $\mathbb{Z}_t^N$ to a polynomial in $\mathcal{R}_t$. To enable server-computation verifiability (see §3.2), we extend

this encoder to introduce error-detection capabilities (see §3.4, §3.5). We term *slot* a component of the plaintext vector before its encoding as a polynomial.

3.3.2 Verifiable Programs

To enable computation verification, it is necessary to clearly identify the inputs, the output, and the function being evaluated. We briefly recall here a formalization of *identifiers* and *programs* for verifiable computation [BFR13, GW13]. Any possible input message m is represented by an identifier τ. This identifier can be seen as a string uniquely identifying the data in a database. A *program* corresponds to the application of a function (*i.e.,* circuit) $f(\cdot)$ to inputs associated with specific identifiers: it is labeled as a tuple $\mathcal{P}=(f(\cdot),(\tau_1,...,\tau_n))$. The correctness of a program ensures that, given the identified inputs, the output of the function is the expected one without corruption. A program is said to be *well-defined* if all inputs contributing to the computation are authenticated.

3.3.3 Homomorphic Authenticators

Homomorphic authenticators (HA) [GW13] are cryptographic schemes used to verify computation integrity. They consist of four probabilistic-polynomial time (PPT) procedures: the key generation (**HA.KeyGen**), the authentication (**HA.Auth**) of the input data, the evaluation (**HA.Eval**) of an agreed-upon function on authenticated data, and the verification (**HA.Ver**) of the claimed output. Correctly instantiated homomorphic authenticators provide authentication and evaluation correctness as well as security (see Appendix A.1 for a formal description). In the following sections, we show how to instantiate an HA using a verifiable HE-encoding. As mentioned in §3.2, by adapting the HE pipeline to work with our new plaintext encoders, we emulate the HA procedures for the authentication, the evaluation, and the verification. Figure 3.2 displays the modifications performed to the original HE pipeline as dashed boxes.

3.4 Replication Encoding

We first design a replication-based verifiable encoding (§3.4.1) that takes advantage of the native HE batching to introduce error-detection and redundancy in the data. Then, we construct an authenticator (§3.4.2) following the footprint of homomorphic MACs [GW13, CKV10].

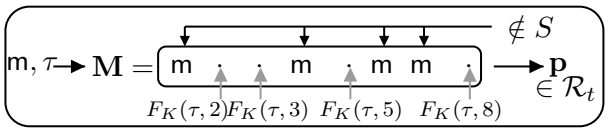


Figure 3.3: Replication Encoding. A message m with identifier τ is encoded as a vector $\mathbf{M}$ with challenge values (using the PRF $F_K(\cdot)$) for indices in the challenge set S and replications of m for all the others. For ease of presentation, here $\mathsf{m} \in \mathbb{Z}_t$ and $\lambda = 8$.

Scheme 1: Replication-Based Encoder in $\mathcal{R}_t{=}\mathbb{Z}_t[X]/\langle X^N{+}1\rangle$

ReplicationEncoder$(\mathbf{m}, \boldsymbol{\tau}; \lambda, S, F_K(\cdot))$:

1. Set an empty vector $\mathbf{M}{=}()$.
2. For the message $\mathbf{m}{\in}\mathbb{Z}_t^N$ with identifiers in $\boldsymbol{\tau}$. $\forall i{\in}[N]$:
   - Set a vector $\mathbf{M}_i \in \mathbb{Z}_t^\lambda$ s.t. $\mathbf{M}_i[j] = \mathbf{m}[i]$ if $j \notin S$ and $\mathbf{M}_i[j]{=}F_K(\boldsymbol{\tau}[i], j)$ otherwise.
   - Set the concatenations $\mathbf{M}{=}\mathbf{M}|\mathbf{M}_i$.
3. Process $\mathbf{M} \in \mathbb{Z}_t^{\lambda N}$as $(\mathbf{p}_1|...|\mathbf{p}_\lambda){\in}\mathcal{R}_t^\lambda$ using the batching encoder.
4. Return $(\mathbf{p}_1|\ldots|\mathbf{p}_\lambda)$.

3.4.1 Replication-Based Encoding

On input a vector $\mathbf{m}$ with identifier $\boldsymbol{\tau}$, a power-of-two security parameter λ, a challenge set $S \subset [\lambda]$ of size $\lambda/2$, and a pseudorandom function (PRF) $F_K(\cdot)$, the replication-based encoder returns an encoding as follows: For each scalar value of $\mathbf{m}$ (say the i-th), an extended vector $\mathbf{M}_i$ of length λ is created. For all indices of $\mathbf{M}_i$ in S, the vector is filled with a challenge value obtained from the PRF and the identifier $\boldsymbol{\tau}[i]$ associated with $\mathbf{m}[i]$. The remaining empty elements of $\mathbf{M}_i$ are filled with replicas of the initial message $\mathbf{m}[i]$. All the extended vectors are then concatenated to create the encoding $\mathbf{M}$. The encoding is then parsed as a concatenation of HE plaintexts and encoded in $\mathcal{R}_t$. A simplified version of the replication encoding for one scalar value is presented in Figure 3.3, and the encoder details are shown in Scheme 1.

3.4.2 Replication-Based Authenticator

We combine the replication-based encoding with an HE scheme to obtain a homomorphic authenticator [GW13]. As we change the plaintext encoder, the evaluation and decryption algorithms of the HE pipeline need to be adapted. We present how in the following.

Let $F_K{:}\mathcal{I}{\to}\mathbb{Z}_t$ be a variable length pseudorandom function (PRF) and HE a secure homomorphic encryption scheme as in §3.3.1 with plaintext space $\mathcal{R}_t{=}\mathbb{Z}_t[X]/\langle X^N{+}1\rangle$ of degree N. We define a *Replication Encoding*-based authenticator (REP) as in Scheme 2 that we describe here.

REP.KeyGen$(1^\lambda)\rightarrow(\mathbf{evk}, \mathbf{sk})$: For a power-of-two security parameter λ, this procedure sets up the PRF with key K and instantiates the HE scheme such that both achieve at least λ-bits security. It also generates the encryption and evaluation keys. The latter comprise the relinearization key and the rotation keys. It randomly picks a challenge set of indices $S \subset [\lambda]$ subject to $|S|=\frac{\lambda}{2}$. Finally, it sets the authenticator evaluation and secret keys.

REP.Auth$(\mathbf{m}, \boldsymbol{\tau}; \mathbf{sk})\rightarrow \boldsymbol{\sigma}$: For an input vector $\mathbf{m}\in\mathbb{Z}_t^N$ with identifier $\boldsymbol{\tau}$ it proceeds sequentially. It first encodes $\mathbf{m}$ by calling **ReplicationEncoder**$(\mathbf{m}, \boldsymbol{\tau}; \lambda, S, F_K(\cdot))$ which returns a list of plaintexts over $\mathcal{R}_t$ (Encode). Then, it encrypts each encoded plaintext and appends it to a list of ciphertexts $\mathbf{c}$ (Encrypt). Finally, it outputs the authentication $\boldsymbol{\sigma}:=\mathbf{c}$.

REP.Eval$(f(\cdot), \vec{\boldsymbol{\sigma}}; \mathbf{evk})\rightarrow \boldsymbol{\sigma}'$: For a function $f(\cdot)$ and n authenticated input vectors represented by $\vec{\boldsymbol{\sigma}}=(\boldsymbol{\sigma}_1,..., \boldsymbol{\sigma}_n)$, it computes $\mathbf{c}'$ by evaluating homomorphically the corresponding SIMD circuit on the ciphertexts $(\mathbf{c}_1,..., \mathbf{c}_n)$. All operations are executed slot-wise, and rotations steps are increased by a factor λ. It outputs $\boldsymbol{\sigma}':=\mathbf{c}'$.

REP.Ver$(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})\rightarrow \{0, 1\}$: It parses the labeled program $\mathcal{P}=(f(\cdot), (\boldsymbol{\tau}_1,..., \boldsymbol{\tau}_n))$. Offline, it pre-computes the challenge values by using both the identifier and the PRF, and it evaluates on them the function $f(\cdot)$ in the plaintext space. If all the challenge evaluations return the same value, abort. Online, it decrypts the ciphertext $\mathbf{c}'$ in $\boldsymbol{\sigma}'$ to the plaintext vector $\mathbf{M}^*$ and checks if for all slots $j\in S$ the output matches the pre-computed challenge for this slot. Moreover, it ensures that all the other slots $j\in[\lambda] \setminus S$ evaluate to the same value. If the above checks pass, it outputs 1 (*i.e.,* it accepts).

Correctness. Authentication correctness follows directly from the correctness of the HE scheme, and evaluation correctness follows from the canonical property and correctness of the HE scheme: *i.e.,* the HE scheme supports circuit evaluation and composition (see [GW13]).

Security. The following theorem states that a misbehaving computing server has only a negligible probability of tampering, without getting caught, with the computation requested by the client.

Theorem 3.4.1. Let λ be a power-of-two security parameter. If the pseudorandom function F_K and the canonical HE scheme are at least λ-bit secure, then for any program $\mathcal{P}$, REP as in Scheme 2 is a secure authenticator and a PPT adversary has a probability of successfully cheating the verification negligible in λ.

Proof Intuition: The security holds by the security of the PRF and by the semantic security of the HE scheme: the server cannot distinguish which slots hold a replicated

Scheme 2: Replication Encoding-Based Authenticator

- **REP.KeyGen**(1^λ): For a power-of-two security parameter λ.
  1. Choose a PRF key $K \leftarrow \{0,1\}^*$ for at least λ-bits security.
  2. Set the HE keys $(\mathbf{pk}_{\text{HE}}, \mathbf{evk}_{\text{HE}}, \mathbf{sk}_{\text{HE}})$=HE.KeyGen$(1^\lambda)$ for at least λ-bits security.
  3. Randomly sample a challenge set $S \subset [\lambda]$ s.t. $|S|=\lambda/2$.
  4. Return $\mathbf{evk}=(\mathbf{evk}_{\text{HE}}, H)$ and $\mathbf{sk}=(K, S, \mathbf{sk}_{\text{HE}})$.
- **REP.Auth**$(\mathbf{m}, \boldsymbol{\tau}; \mathbf{sk})$: For $\mathbf{m} \in \mathbb{Z}_t^N$ with identifiers in $\boldsymbol{\tau}$.
  - Encode:$(\mathbf{p}_1|...|\mathbf{p}_\lambda)$=**ReplicationEncoder**$(\mathbf{m}, \boldsymbol{\tau}; \lambda, S, F_K(\cdot))$
  - Encrypt:
    1. Set an empty list $\mathbf{c}$=().
    2. $\forall i \in [\lambda]$, set $\mathbf{c} = \mathbf{c}|\text{HE.Enc}(\mathbf{p}_i; \mathbf{pk}_{\text{HE}})$
    3. Return $\boldsymbol{\sigma}$:=$\mathbf{c}$.
- **REP.Eval**$(f(\cdot), \vec{\boldsymbol{\sigma}}; \mathbf{evk})$: For a function $f(\cdot)$ to be computed over previously authenticated encrypted inputs stored in $\vec{\boldsymbol{\sigma}}$:
  1. Parse the authenticated inputs $\vec{\boldsymbol{\sigma}}=(\boldsymbol{\sigma}_1,..., \boldsymbol{\sigma}_n)=(\boldsymbol{c}_1,..., \boldsymbol{c}_n)$.
  2. Evaluate $\mathbf{c}' = \text{HE.Eval}(f(\cdot), (\mathbf{c}_1,..., \mathbf{c}_n); \mathbf{evk}_{\text{HE}})$.
  3. Return $\boldsymbol{\sigma}'$:=$\mathbf{c}'$.
- **REP.Ver**$(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})$: Parse the target program $\mathcal{P}=(f, (\boldsymbol{\tau}_1,..., \boldsymbol{\tau}_n))$ and $\mathbf{c}' = \boldsymbol{\sigma}'$. Pre-compute the challenge values:
  1. $\forall i \in S$, $\forall j \in [n]$, $\forall k \in [N/\lambda]$, set $r_{i,j,k}=F_K(\boldsymbol{\tau}_j[k], i)$.
  2. $\forall i \in S$ compute on the challenges $\tilde{r}_i = f(\{r_{i,j,k}\}_{j\in[n]}^{k\in[N/\lambda]})$ (w.l.o.g. the output is a scalar value stored in the first extended vector).
  3. If $\forall i \in S$ $\tilde{r}_i = \tilde{r}$, then abort.

  In the online phase:
  1. Set $\mathbf{M}^*$=$(\mathbf{M}_1,...,\mathbf{M}_l)$=HE.Dec$(\mathbf{c}'; \mathbf{sk}_{\text{HE}})$ with $l=N/\lambda$.
  2. $\forall i \in S$, check if $\mathbf{M}_1[i]=\tilde{r}_i$. If not, return 0.
  3. Check that $\forall i \in [\lambda] \setminus S$, all $\mathbf{M}_1[i]$ have the same value. If not, return 0.
  4. Return 1 (*i.e.,* accept).

value or what the challenge values are. In fine, the adversary can cheat without being detected with only a negligible probability. The formal proof is presented in Appendix A.2.

Overhead. We observe that REP induces, in the worst case, communication and computational overhead that is linear in the security parameter λ for both the client and the server. Indeed, a vector of N values needs to be encoded as λ separate ciphertexts and $\lambda/2$ challenges need to be pre-computed before the verification. In comparison with the original homomorphic MAC [GW13], REP does not modify the encryption procedure. Compared to computation delegation [CKV10], REP enables verification of computations over unknown but identified by $\boldsymbol{\tau}$ inputs.

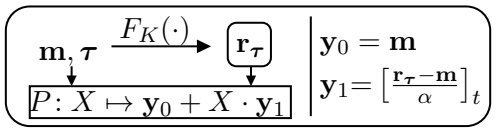


Figure 3.4: Polynomial Encoding. A message $\mathbf{m} \in \mathbb{Z}_t^N$ identified by $\boldsymbol{\tau}$ is encoded as P using the secret α and the challenge vector $\boldsymbol{r}_{\boldsymbol{\tau}}$.

3.4.3 Optimizing the Verification

REP relies on both the generation of challenges and the circuit evaluation on these values. As these are independent of the inputs (only their identifiers are required), they can be pre-processed and computed offline. The computation on the challenges can be outsourced to any entity that does not collude with the computing server. Any standard verifiable computation techniques, such as interactive proofs [GKR15], SNARKs [BBB+18] or STARKs [BSCR+19], could be employed to ensure the correct computation (see Appendix 3.7.1). Additionally, similarly to prior work on computation verification [BFR13, FG12], one can rely on a closed-form PRF [BGV11] to enable the partial re-use of the challenges for evaluation over different datasets. Depending on the function under evaluation, this could accelerate the verification at increased costs for generating the encoding and the challenges. This also holds for the second encoder that we present next.

3.5 Polynomial Encoding

We design a more compact encoding (§3.5.1) and construct an authenticator that requires fewer challenge values to be verified (§3.5.2) following Catalano and Fiore's information-theoretic MAC [CF13].

3.5.1 Polynomial-Based Encoding

On input a vector $\mathbf{m} \in \mathbb{Z}_t^N$ with identifier $\boldsymbol{\tau}$, a secret key α, and a pseudorandom function (PRF) $F_K(\cdot)$, the polynomial-based encoder outputs a polynomial P such that $P(0)=\mathbf{m}$ and $P(\alpha)=\boldsymbol{r}_{\boldsymbol{\tau}}$, with $\boldsymbol{r}_{\boldsymbol{\tau}}$ the challenge vector obtained from the PRF $F_K(\cdot)$. Then, each component of P is encoded using batching. A simplified version of the encoding is represented in Figure 3.4 and the details of the encoder are shown in Scheme 3.

3.5.2 Polynomial-Based Authenticator

We combine the polynomial-based encoding with an HE scheme and show that it instantiates a homomorphic authenticator [CF13]. Similar to REP, we describe the complete

Scheme 3: Polynomial-Based Encoder in $\mathcal{R}_t=\mathbb{Z}_t[X]/\langle X^N+1\rangle$

PolynomialEncoder$(\mathbf{m}, \boldsymbol{\tau}; \alpha, F_K(\cdot))$:
1. Compute $\mathbf{r}_{\boldsymbol{\tau}} \in \mathbb{Z}_t^N$ s.t. $\forall i \in [N]$, $\mathbf{r}_{\boldsymbol{\tau}}[i]=F_K(\boldsymbol{\tau}[i])$.
2. Set $\mathbf{y}_0$ and $\mathbf{y}_1$ to be the batching encoding of $\mathbf{m}$ and $[\frac{\mathbf{r}_{\boldsymbol{\tau}}-\mathbf{m}}{\alpha}]_t$.
3. Return $P = (\mathbf{y}_0, \mathbf{y}_1)$.

pipeline.

Let F_K:$\mathcal{I}\rightarrow\mathbb{Z}_t$ be a variable length PRF and HE a secure homomorphic encryption scheme as in §3.3.1 with plaintext space $\mathcal{R}_t=\mathbb{Z}_t[X]/\langle X^N+1\rangle$ of degree N. We define a *Polynomial Encoding*-based authenticator (PE) in Scheme 4 and briefly describe it here.

PE.KeyGen$(1^\lambda)\rightarrow(\mathbf{evk}, \mathbf{sk})$: For t a λ-bit prime number, it samples a random invertible point α in $\mathbb{Z}_t^*$. It sets up the PRF with key K and the HE scheme such that both achieve at least λ-bit security. It generates the encryption and evaluation keys for the HE scheme. Finally, it outputs the authenticator evaluation and secret keys.

PE.Auth$(\mathbf{m}, \boldsymbol{\tau}; \mathbf{sk})\rightarrow \boldsymbol{\sigma}$: On input a plaintext vector $\mathbf{m} \in \mathbb{Z}_t^N$ with identifiers $\boldsymbol{\tau}$ (and N the degree of the plaintext polynomial ring), it calls **PolynomialEncoder**$(\mathbf{m}, \boldsymbol{\tau}; \alpha, F_K(\cdot))$ that returns $(\mathbf{y}_0, \mathbf{y}_1)\in\mathcal{R}_t^2$ (Encode). Then, it encrypts $\mathbf{y}_0$ and $\mathbf{y}_1$ to $\mathbf{c}_0$ and $\mathbf{c}_1$ respectively using the HE scheme. It returns the authentication $\boldsymbol{\sigma}=(\mathbf{c}_0, \mathbf{c}_1)$. Note that initially, a message is encoded as a degree-one polynomial in $\mathcal{R}_q$ (*i.e.,* a length 2 list of polynomials) identified by its coefficients $\mathbf{c}_0$ and $\mathbf{c}_1$. Through the evaluation process, this degree can increase. We denote by d the polynomial encoding's degree (*e.g.,* d=1 after a fresh authentication).

PE.Eval$(f(\cdot), \vec{\boldsymbol{\sigma}}; \mathbf{evk})\rightarrow \boldsymbol{\sigma}'$: A function $f(\cdot)$ represented by an arithmetic circuit $\mathcal{C}$ over the ciphertexts is evaluated gate by gate on the n previously authenticated input vectors in $\vec{\boldsymbol{\sigma}}=(\boldsymbol{\sigma}_1,..., \boldsymbol{\sigma}_n)$. The output of each gate in $\mathcal{C}$ depends on its functionality: Additive gates execute the corresponding HE operation component-wise on the input authentications, and the multiplicative ones perform a convolution. Rotation gates execute the corresponding homomorphic rotation on each component of the input authentication. After the circuit evaluation, this procedure returns $\boldsymbol{\sigma}'$, the authentication output of the final gate in $\mathcal{C}$.

PE.Ver$(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})\rightarrow \{0, 1\}$: It parses the labeled program $\mathcal{P}=(f, (\boldsymbol{\tau}_1, ..., \boldsymbol{\tau}_n))$ and $\boldsymbol{\sigma}'=(\mathbf{c}_0,..., \mathbf{c}_d)$. Offline, it pre-computes the value $\boldsymbol{\rho} = f(F_K(\boldsymbol{\tau}_1),..., F_K(\boldsymbol{\tau}_n))$. Online, it decrypts the encrypted vector $\boldsymbol{\sigma}'$ to $(\mathbf{y}_0,..., \mathbf{y}_d)$ and checks if $\boldsymbol{\rho}= \sum_{i=0}^{d} \mathbf{y}_i \cdot \alpha^i$. If the check passes, it accepts the result $\mathbf{y}_0$.

Correctness. Authentication correctness follows from the correctness of the HE scheme (§1.3). The evaluation correctness holds from the construction. For linear gates (additions, multiplication by constant, and rotations), correctness follows from the correctness of the

Scheme 4: Polynomial Encoding-Based Authenticator

- **PE.KeyGen**(1^λ): Let t be a λ-bit prime number.
  1. Sample a random invertible point $\alpha \leftarrow \mathbb{Z}_t^*$.
  2. Choose a PRF key $K \leftarrow \{0,1\}^*$ for at least λ-bits security.
  3. Initialise the HE keys $(\mathbf{pk}_{\text{HE}}, \mathbf{evk}_{\text{HE}}, \mathbf{sk}_{\text{HE}}) = \text{HE.KeyGen}(1^\lambda)$ for at least λ-bits security.
  4. Return $\mathbf{evk} = (\mathbf{evk}_{\text{HE}}, t)$ and $\mathbf{sk} = (K, \alpha, \mathbf{sk}_{\text{HE}})$.
- **PE.Auth**$(\mathbf{m}, \boldsymbol{\tau}; \mathbf{sk})$: For $\mathbf{m} \in \mathbb{Z}_t^N$ with identifiers in $\boldsymbol{\tau}$.
  - Encode: $(\mathbf{y}_0, \mathbf{y}_1) = \textbf{PolynomialEncoder}(\mathbf{m}, \boldsymbol{\tau}; \alpha, F_K(\cdot))$
  - Encrypt:
  1. Create $\mathbf{c}_i = \text{HE.Enc}(\mathbf{y}_i; \mathbf{pk}_{\text{HE}})$, $\forall i \in [0{:}1]$.
  2. Return $\boldsymbol{\sigma} := (\mathbf{c}_0, \mathbf{c}_1)$.
- **PE.Eval**$(f(\cdot), \vec{\boldsymbol{\sigma}}; \mathbf{evk})$: Let $\mathcal{C}$ be the HE arithmetic circuit of a function $f(\cdot)$ to be computed over previously authenticated encrypted inputs stored in $\vec{\boldsymbol{\sigma}}$.
  1. Parse $\vec{\boldsymbol{\sigma}} = (\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_n)$.
  2. Evaluate homomorphically each gate in $\mathcal{C}$ and output $\boldsymbol{\sigma} = (\mathbf{c}_0, ..., \mathbf{c}_d)$, an authentication of degree d, s.t.:
  - Addition: On input two authentications $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ of degree d_1 and d_2 resp., set $d = \max(d_1, d_2)$ and $\boldsymbol{\sigma} = \boldsymbol{\sigma}_1 + \boldsymbol{\sigma}_2$, *i.e.,* $\forall k \in [d{+}1]$, $\boldsymbol{\sigma}[k] = \text{HE.Add}(\boldsymbol{\sigma}_1[k], \boldsymbol{\sigma}_2[k])$.
  - Multiplication: On input $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ of degree d_1 and d_2 resp., set $d = d_1 + d_2$ and perform a convolution, *i.e.,* $\forall k \in [d{+}1]$, $\boldsymbol{\sigma}[k] = \sum_{i=1}^{k} \text{HE.Mul}(\boldsymbol{\sigma}_1[i], \boldsymbol{\sigma}_2[k{-}i{+}1]; \mathbf{evk}_{\text{HE}})$.
  - Rotation by r: On input an authentication $\boldsymbol{\sigma}_1$ of degree d_1, set $d = d_1$ and rotate all components of $\boldsymbol{\sigma}$, *i.e.,* $\forall k \in [d{+}1]$, $\boldsymbol{\sigma}[k] = \text{HE.Rot}_r(\boldsymbol{\sigma}_1[k]; \mathbf{evk}_{\text{HE}})$.
  3. Return $\boldsymbol{\sigma}'$, the authentication output of the final gate in $\mathcal{C}$.
- **PE.Ver**$(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})$: Parse the target program $\mathcal{P} = (f, (\boldsymbol{\tau}_1, ..., \boldsymbol{\tau}_n))$. Precompute the challenge values:
  1. $\forall i \in [n]$, $\forall j \in [N]$, set $\mathbf{r}_{\boldsymbol{\tau}_i} \in \mathbb{Z}_t^N$ s.t. $\mathbf{r}_{\boldsymbol{\tau}_i}[j] = F_K(\boldsymbol{\tau}_i[j])$.
  2. Compute $\boldsymbol{\rho} = f(\mathbf{r}_{\boldsymbol{\tau}_1}, ..., \mathbf{r}_{\boldsymbol{\tau}_n})$.

  Then, during the online phase, parse $\boldsymbol{\sigma}' = (\mathbf{c}_0, \ldots, \mathbf{c}_d)$ obtained from the evaluation and:
  1. Set the result to $\mathbf{y}_0 = \text{HE.Dec}(\mathbf{c}_0; \mathbf{sk}_{\text{HE}})$.
  2. Set $\mathbf{y}_i = \text{HE.Dec}(\mathbf{c}_i; \mathbf{sk}_{\text{HE}})$, $\forall i \in [0{:}d]$.
  3. Check if $\boldsymbol{\rho} \stackrel{?}{=} \sum_{i=0}^{d} \mathbf{y}_i \cdot \alpha^i$, else return 0.
  4. Return 1 (*i.e.,* accept).

HE scheme encrypting the encoding of the input. The multiplicative gate is a convolution between the inputs: When executed on the ciphertext it ports to the plaintexts by the correctness of the HE scheme. By the definition of polynomial convolution, this leads to an encoding of the product of scalar inputs. PE differs from the original authenticator by

Catalano and Fiore [CF13] by injecting, before encryption, the encoding in the native plaintext polynomial ring.

Security. The following theorem states that a misbehaving server has only a negligible probability of cheating without being detected.

Theorem 3.5.1. If the pseudorandom function F_K and the canonical HE scheme are at least λ-bit secure and if t is a λ-bit prime number, then, for any program $\mathcal{P}$ with authentications of bounded degree, PE is a secure authenticator and a PPT adversary has a probability of successfully cheating the verification negligible in λ.

Proof Intuition. The security holds from the security of the PRF and the polynomial identity lemma for various polynomials over the field $\mathbb{Z}_t$ (as t is a prime number); the probability of the adversary cheating without being detected is negligible in λ. The formal proof is presented in Appendix A.3.

Overhead. After the evaluation, the size of the resulting PE authentication $\boldsymbol{\sigma}'$ grows linearly with respect to the degree of the polynomial computation due to the convolution between the authentications. Hence, for programs with large multiplicative depth, PE might introduce a significant communication overhead between the client and the server and become less communication efficient than REP. This growth can also affect the client's computational overhead due to the cost of pre-computing the challenges plus the cost of the polynomial evaluation (Step 3 in PE.Ver). To account for this issue, we design a *polynomial compression protocol* (§3.5.3). Moreover, the convolutions introduced by the multiplicative gates during the evaluation phase yield a non-negligible computational and communication overhead for the computing server (see §3.6). For client-aided scenarios, we mitigate the effects on the computation and communication overheads, by introducing a new technique called *re-quadratization* (§3.5.4).

3.5.3 Polynomial Compression Protocol

As seen in §3.5.2, the size of the PE authentication $\boldsymbol{\sigma}'$ grows linearly with the degree of the function represented by the evaluated circuit (*i.e.,* it comprises more and more ciphertexts). This can cause a significant communication overhead between the server and the client. To mitigate this overhead, we design a polynomial compression protocol (PoC) that compresses the authentication from d+1 ciphertexts to only two, for authentications of degree d. Informally, once the computing server (*i.e.,* the prover) has sent the claimed output $\mathbf{y}_0$, the client (*i.e.,* the verifier) challenges the server with a hash function. The server responds with the hash digest and the intermediate results $\{w_i\}_{i=0}^{d}$. A sketch of our PoC can be seen in Figure 3.5. This protocol is made non-interactive in the ROM using the Fiat-Shamir heuristic. As we will see in §3.6.3, the PoC reduces the communication and verifier overhead at the cost of higher runtimes for the prover.

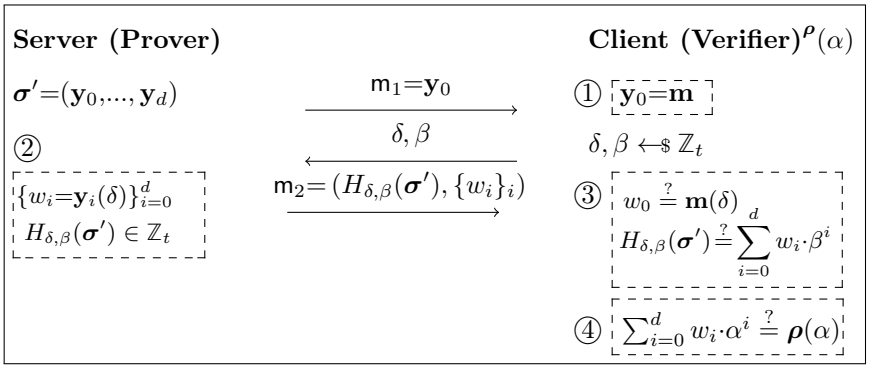


Figure 3.5: Polynomial Compression Protocol (PoC, §3.5.3). For clarity, polynomials are represented in the plaintext space.

The combination of PE with PoC leads to a secure authenticator. Informally, the compression uses a polynomial hashing mechanism that by the polynomial identity lemma gives a negligible soundness error for an appropriate choice of parameters. We provide the full proof in Appendix A.4.

3.5.4 Interactive Re-Quadratization

As discussed in §3.5.2, PE executes a convolution for every multiplication gate of the evaluation circuit. Hence, due to the growing number of homomorphic multiplications and ciphertexts, the longer the input encodings are, the heavier the server's operations are. When interactions between the server and the client are possible (*i.e.,* in client-aided settings), we propose a *re-quadratization* (ReQ) (similar to the relinearization used in HE). It converts an authentication comprising five ciphertexts (*i.e.,* $\boldsymbol{\sigma}'=(\mathbf{c}_0,...,\mathbf{c}_4)$ for the authentication of a depth-two computation) to one of three ciphertexts (*i.e.,* related to a depth-one). In a nutshell, ReQ evaluates part of the verification procedure for the terms of higher degree (*i.e.,* $\mathbf{c}_3$ and $\mathbf{c}_4$) and embeds the result in the lower terms (*i.e.,* $\mathbf{c}_1$ and $\mathbf{c}_2$). Our interactive protocol bounds the authentication to, at most, three ciphertexts, thus limiting the computational burden of computing the convolution of high-degree polynomials for the server (see §3.6.3).

Figure 3.6 presents our protocol in detail. In a nutshell, the client decrypts the higher term ciphertexts sent by the server (*i.e.,* $\mathbf{c}_3$ and $\mathbf{c}_4$) and returns to the server encryptions of two masked random linear combinations of the plaintexts. The random masks create an offset of the encoding that is removed by the shift value Δ_G that keeps track of the offset up to the re-quadratization after the G-th multiplication gate. The correctness of PE combined with ReQ holds directly by construction. The security of the combination is guaranteed by the random maskings. A full description of ReQ and its security proof are presented in Appendix A.5.

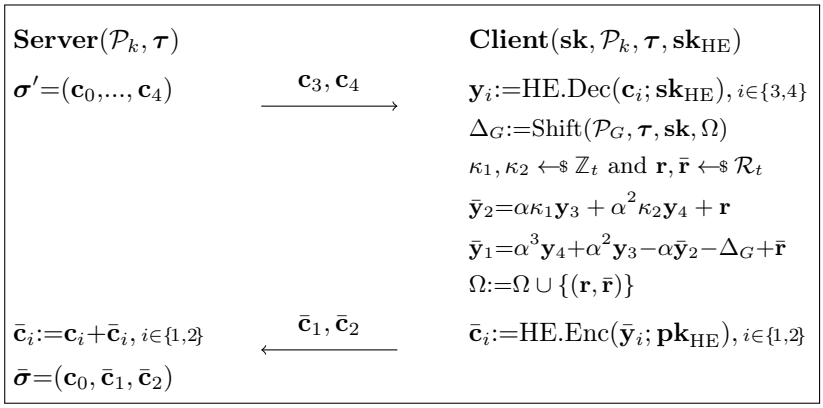


Figure 3.6: Interactive Re-Quadratization (ReQ) for gate G (§3.5.4).

Similar to other client-aided operations [JVC18, AGHV22, AFS18], ReQ trades-off interactivity for higher performance. The theoretical communication complexity of ReQ is the exchange of four ciphertexts per re-quadratization. This leads to a linear overhead in the degree of the function as in the original PE. The memory complexity for the client is to keep the state of the blinding list Ω (storing two new polynomials per re-quadratization). The computation complexity for the client is to execute two HE encryption/decryption operations and to compute the offset Δ_G (*i.e.,* a plaintext convolution). Overall, ReQ requires that (i) both client and server be online, and (ii) the client engages in some computations. Nevertheless, ReQ has the major advantage of maintaining a $\boldsymbol{\sigma}'$ of at most degree-two, which implies (i) lower communication overhead to obtain the result, and (ii) lower computation overhead for the server compared to the original PE (as the convolution now involves at most two degree-two polynomials).

3.6 VERITAS

We introduce VERITAS, a new library that implements the two encodings and authenticators (and their optimizations) described in §3.4 and §3.5. We first present our implementation (§3.6.1), before benchmarking native HE operations (§3.6.2) and evaluating it over several use-cases (§3.6.3). We then compare VERITAS' performance with prior work (§3.6.4) and summarize the main takeaways (§3.6.5).

3.6.1 Implementation and Hardware

VERITAS facilitates the transition from a standard HE pipeline (Figure 3.1) to a pipeline enhanced with computation verification capabilities (Figure 3.2) by implementing (i) our two encoders and (ii) the corresponding authenticators. VERITAS instantiates REP and PE with the BFV homomorphic encryption scheme, and Blake2b [SA15] as a PRF.

VERITAS offers flexible parameterization of the authenticator security parameter (λ) to provide sufficient security guarantees, depending on the application requirements (typically, $\lambda \geq 32$ for malicious-but-rational adversary models [Lin13]), and it sets the HE and hash function security requirements to $\max(\lambda, 128)$-bits. By default, using VERITAS does not provide any feedback to the computing server (see Appendix 3.7.3). As BFV is a semantically secure canonical homomorphic encryption scheme and Blake2b yields a secure PRF [LMN16], and as both are parameterized to achieve 128-bits security, VERITAS securely instantiates REP and PE. Thus, it protects the privacy of the client data and the computation result; by semantic security of BFV, the ciphertexts and the authentications reveal nothing about the underlying data before decryption. We build VERITAS in Golang on top of Lattigo's BFV implementation [EPF21]. It is modular and enables developers to seamlessly obtain client verification capabilities in existing homomorphic encryption pipelines (*i.e.,* by only changing a few lines of code). VERITAS can be employed to verify the result correctness of any circuit admissible by BFV. We show its versatility over various use cases in §3.6.3. All experiments were conducted on a machine with an Intel Xeon E5-2680 v3 processor and 256 GB of RAM. VERITAS' code is available in an open-source repository [LDS21].

3.6.2 Benchmarking BFV Operations

As described in §3.4 and §3.5, both authenticators trivially support the BFV linear operations (*e.g.,* addition, subtraction, constant multiplication): These operations are simply executed on all components of the authentication $\boldsymbol{\sigma}$. Although both authenticators support any rotations, REP requires keys with an increase of the rotation step by a factor of λ to avoid mixing up the various slots. These linear operations do not expand the size of the authentication. REP naturally supports multiplication operations, whereas PE requires a convolution for every multiplication. The computation complexity of this operation depends on the degree d of the input authentication. Both REP and PE naturally support relinearization. We present benchmarks of the amortized evaluation costs (over the ciphertext slots) for each BFV operation in Table 3.2 for λ=32, averaged over 100K runs. The BFV parameters are set to ($\log N$=14, $\log q$=438). We observe that REP's linear operation timings are multiplied by λ (as the messages are replicated λ times), whereas PE less than triples them. Regarding multiplications, REP's computational overhead remains constant and the computational overhead of the PE scheme increases with every depth of the circuit. Furthermore, each multiplication introduces a linear growth of the authentication size hence affects memory and communication. In terms of memory requirements, REP has in the worst case a linear overhead in λ (this can be reduced if the whole encoding fits into a single ciphertext). For PE, the size d of the authentication is linear in the degree of the evaluated function. We discuss in Appendix A.6 the influence of the security parameter λ on VERITAS' overhead.

Table 3.2: Amortized timings of VERITAS for homomorphic operations evaluation (μs) for an authenticator's security parameter λ=32. The baseline is the standard BFV scheme.

Op.	Add.	Mul. Const.	Rot.	Relin.	Mul. depth		
					1	2	3
BFV	0.02	0.04	2.3	1.8	2.5	3.5	4.0
REP	1.07	1.4	70	60	105	108	118
PE	0.09	0.1	2.9	4.8	9.3	15.0	30.1

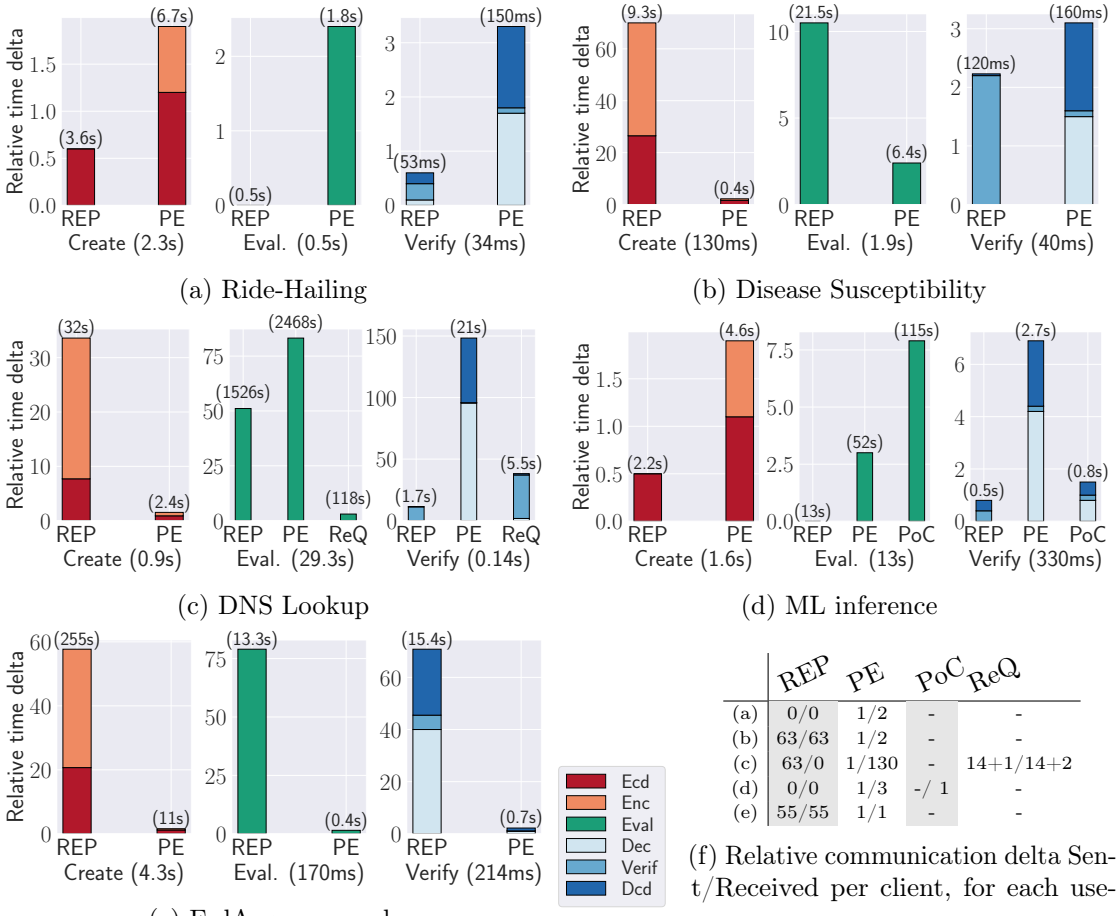
3.6.3 Experimental Case Studies

We now evaluate VERITAS on five use-cases by introducing computation verification capabilities in existing homomorphic pipelines: ride-hailing (§3.6.3.1), genomic-data analysis (§3.6.3.2), encrypted search (§3.6.3.3), and machine-learning prediction and training (§3.6.3.4 and §3.6.3.5, resp.). For each use-case, we first describe the baseline (*i.e.,* the homomorphic encryption pipeline that protects only privacy) and then analyze the performance of VERITAS using both authenticators and their optimizations when relevant. Note that, although the PE operations are embarrassingly parallelizable, we evaluate them on a single thread for the sake of comparison. VERITAS' relative (with respect to the baseline) computation and communication overheads (*i.e.,* $(x - x_{\text{base}})/x_{\text{base}}$) are presented in Figure 3.7; these overheads are averaged over 1K runs. We group the homomorphic authenticator (HA) procedures into three stages: (a) the **Create** stage represents the HA.KeyGen() and HA.Auth() procedures executed by the (offloading) client, (b) the **Eval.** stage that accounts for the HA.Eval() ran by the computing server, and (c) the **Verify** stage which invokes the (decrypting) client HA.Ver() procedure.

3.6.3.1 Ride-Hailing Services

Ride-hailing services enable the matching of a driver to a customer (or rider), depending on their locations. As location data can leak sensitive information [Her18, VDSKK18], existing solutions use HE to protect confidentiality against the ride-hailing service [AHHK18, PDE+17]. Furthermore, it is crucial to verify the correctness of the matching in order to ensure a fair and transparent process [BH20, LGC+20].

Description. We evaluate an HE pipeline inspired by ORide [PDE+17]. Each driver encrypts their location into a single ciphertext (each coordinates into a different slot piloted by the driver ID). The rider encrypts its location into a fully packed ciphertext (replicating its coordinates through the whole vector). The server computes the squared difference between the ciphertext of the rider and the ciphertext resulting from the sum of all drivers' ciphertexts. It returns the result to all drivers and to the rider who decrypts to discover the closest match. Using VERITAS, the drivers and the rider collectively generate the authenticator key and encode their respective location vectors.



(a) Ride-Hailing

(b) Disease Susceptibility

(c) DNS Lookup

(d) ML inference

(e) FedAvg per epoch

	REP	PE	PoC	ReQ
(a)	0/0	1/2	-	-
(b)	63/63	1/2	-	-
(c)	63/0	1/130	-	14+1/14+2
(d)	0/0	1/3	-/ 1	-
(e)	55/55	1/1	-	-

(f) Relative communication delta Sent/Received per client, for each use-case.

Figure 3.7: VERITAS' computation overhead (1K runs avg.) for various use-cases (Figs. 3.7a-3.7e). The x-axis displays the authenticator/optimization employed. The y-axis represents the relative time delta wrt the HE baseline. The values in parenthesis on top of the bars represent the actual execution times. The values in parenthesis after each stage (Create, Eval., Verify) correspond to the HE baseline runtimes. PoC and ReQ are PE's optimizations from §3.5.3 and §3.5.4, resp. The various operations are the client encoding (Ecd) and encryption (Enc), the server evaluation (Eval), the client decryption (Dec), verification of the challenges (Verif), and decoding (Dcd). ReQ and PoC overheads are included in the (Eval) and (Verif) stages. For each use-case, Table 3.7f displays VERITAS' relative communication overhead w.r.t. the HE baseline.

Parameterization. In our use case, we consider 32 drivers, and set the HE cryptographic parameters to $\log N$=15 and $\log q$=700. With BFV, the client and drivers need in total 2.3s to encrypt their location, and the server's matching requires 0.5s. The client decryption/decoding needs 34ms and the client egress/ingress communication per client is 6.3MB. We parameterize the encodings with a security parameter $\lambda \geqslant 40$ (*i.e.*, λ=64 for REP and t a 56-bit prime for PE). Due to this use-case's low multiplicative depth, we do not present the PoC and ReQ optimizations.

Results. In Figure 3.7a, we observe that both authenticators induce minimal overheads for the offloading client. With REP, the client's computation time for the creation of the encodings is multiplied by 1.6, whereas PE is 1.9× slower than the BFV baseline for encoding and encryption. This modest overhead is due to the small amount of data handled: REP's extended vector fits into a single ciphertext. As a result, the client needs 3.6s and 6.7s to generate, respectively, a REP and a PE authentication. REP has almost no effect on the evaluation time, whereas the circuit's multiplication leads to a 2.4× increase in the server evaluation with PE (1.8s). The effort of the decrypting client for REP is mainly for the verification of the challenges, whereas for PE this is negligible; its cost is dominated by the decryption and decoding of the ciphertexts. Indeed, compared to the baseline (34ms), it takes 3.3× more time for PE to decrypt, decode, and verify the result of the challenges. Although REP does not affect the communication overhead between the server and the client, PE doubles the server's ingress and triples the egress costs (see Table 3.7f).

3.6.3.2 Genomic-Data Analysis

The emergence of direct-to-consumer and medical services that collect DNA to improve users' health and to customize their treatment [23a19, DNA19], along with the immutable and personal nature of genomic data, raises numerous privacy concerns [EN14]. As a result, several works consider protecting the confidentiality of the genomic data with HE [ARHR13, DDC14, DCFT13, WZD+16]. The medical nature of the computations performed on this data entails the correctness of the outcome to avoid misdiagnosis or insurance fraud [BBFR15, TAD16].

Description. We apply VERITAS to a disease-susceptibility computation. Users offload an encrypted version of their DNA (single nucleotide polymorphisms or SNPs) to the server. Later on, a medical institution offloads encrypted weights to the server; these are used for a specific disease prediction. The server computes the scalar product between the weights and the SNPs, and it returns the result to the user. The user can verify the result, even if it does not have access to the weights in cleartext and has access only to their identifiers shared by the medical institution.

Parameterization. The user encrypts 2^{15} SNPs of her DNA, and the medical center encrypts their corresponding weights for breast cancer prediction [BMC+19]. The HE cryptographic parameters are thus set to ($\log N$=15, $\log q$=700). The baseline client encoding/encryption time is 0.13s, and the disease susceptibility computation by the server requires 1.9s. The decryption/decoding needs 39ms, and the server ingress/egress communication is 6.3/6.3MB, respectively. We set the authenticator security parameters to λ=64 for REP and to $\log t$=56 for PE.

Results. Figure 3.7b shows that, as the client offloads a fully packed ciphertext, the overhead induced by REP's creation is 70× the baseline (9.2s). In contrast, PE only triples the creation time (0.4s). The ingress communication to the server and its computations abide by a similar scaling (404MB for REP and 13MB for PE). The server evaluation time, compared to the baseline, is tripled in the case of PE (6s) and is 10× for REP (22s). This is due to an optimized evaluation circuit for REP that only performs the required inner sum after the 64 SNPs/weights ciphertexts are multiplied slot-wise; a naive evaluation would lead to a λ-factor overhead. The ingress communication to the decrypting client is tripled for PE, whereas it remains unchanged for REP. We also observe that REP's verification time (123ms) is dictated by the verification of the challenges ($\lambda/2$ of them), whereas the fast verification time of PE's challenge is shadowed by the decryption and decoding of 2 additional ciphertexts compared to the baseline (leading to 157ms).

3.6.3.3 Search on Encrypted Data

Encrypted lookups protect the confidentiality of sensitive data while enabling clients to query a database that stores it [AFS18, CDSG+21, RVVV17, WYXZ20]. As both the database and the query are encrypted, the executing server does not learn any information. This can be applied, for instance, to an encrypted DNS search or to a query for the existence of a password in a list of vulnerable passwords [NPR19]. In such cases, an unencrypted query would leak information to the server, *e.g.,* the websites a client is trying to access or their passwords; and an incorrect search result could lead to an application-level vulnerability, *e.g.,* redirection to a malicious website, or to the use of an insecure password.

Description. We implement an encrypted DNS search in a database of domains of at most 16 characters in ASCII bit-representation. We follow the blueprints of the lookup use-case implemented in HElib [IBM21a] that we adapt to BFV. The query's bit-representation is XORed with the bit-representation of all database entries. If any of the result's bits are *null* (*i.e.,* this entry did not match the query), its bit representation is multiplied by 0. The results for each entry are aggregated into a single ciphertext. For efficiency, several database entries are packed into a single ciphertext. We deliberately choose this use-case to demonstrate the impact of a large multiplicative depth on VERITAS' efficiency.

Parameterization. The authenticator parameters are set to λ=64 for REP and to $\log t$=58 for PE. To support the circuit, we opt for ($\log N$=16, $\log q$=1, 440). We focus on a fully packed ciphertext that comprises 512 database entries. The client needs 0.9s to encode and encrypt the search query, and the server requires 30s to run it on the encrypted database. The decryption/decoding executes at 140ms, and the client in/out communication is 25MB per ciphertext.

Results. REP introduces a significant computational and communication overhead for the offloading client as the extended vector is encrypted into λ ciphertexts. PE, on the contrary, less than triples the encoding/encryption time and communication overhead towards the server. The server computational overhead is 51× more important for REP (1,500s), compared to the baseline. As the circuit is of depth 7 (with relinearization), the convolutions required by the PE create significant computation and egress communication overheads for the server (*i.e.,* 83× or 2,500s) for computations and 131× more communication than the baseline). Accordingly, PE's verifying cost is dominated by the numerous ciphertexts that store the encoding (21s). We observe that ReQ significantly reduces by more than 20-fold the server and client computational overhead while minimizing the communication overhead, compared to the standard PE approach (at the cost of online client-server interactions). With ReQ, the server evaluation now takes only 120s with a short client involvement during the interaction (4.9s). The client verification eventually requires only 550ms (5.5s in total compared to 21s for standard PE). REP does not need this optimization but needs more challenge values to be verified in the **Verify** phase.

3.6.3.4 Machine-Learning Prediction

Advances in machine learning (ML) enable new ways for data analytics and several works explore the use of HE in order to protect, during ML inference, data confidentiality [BGBE19, MSS20, NWT+20, XLR+20, ZFW+20]. However, as the correctness of the prediction can be tantamount to confidentiality, we apply VERITAS to an encrypted ML inference pipeline. Indeed, misclassification or malicious predictions could render the application pointless and have dire consequences, such as financial misprediction or cyberthreat misclassification, for the end users [GGG17, MSS20, WYX+21, XLR+20].

Description. For this use case, we re-implemented the Low-Latency Privacy Preserving Inference (LoLa) [BGBE19] pipeline in Go. This pipeline uses a neural network composed of a convolutional layer (with 5 kernel maps of size 5×5 and a stride of 4), followed by a square activation function and a fully connected layer. The model is pre-trained on the MNIST hand-written digit dataset (comprising 28×28 grayscale images) and its weights are offloaded by an ML provider. Following [BGBE19], the client encodes and packs each image into 25 ciphertexts with different packing approaches that facilitate the neural-network operations.

Parameterization. We use the HE parameters ($\log N$=15, $\log q$=700). The client requires 1.6s to encode and encrypt the input image, and the server requires 13s to perform the inference. The decryption/decoding executes at 330ms and the server ingress/egress communication is 160/63MB. We set λ=64 for REP and $\log t$=56 for PE.

Results. Figure 3.7d shows that, as the ciphertexts are not fully packed, REP introduces

a negligible overhead for the offloading client (less than 1×). The PE creation is 2× slower than the baseline. Whereas REP is seamless for the server, PE introduces an overhead that is 3× more; the activation function requires a multiplication that introduces convolutions. The verification cost for REP is piloted by the challenge verification and the decoding, which leads to a total **Verify** phase of 0.6s. PE verifies the challenges in 0.06s but introduces decryption and decoding overhead due to the exchange of 3× more ciphertexts than the baseline. We observe that PoC reduces the communication to only two ciphertexts assuming an investment from the server (*i.e.,* 8× slower evaluation).

3.6.3.5 Federated Learning

Federated learning has emerged as a technique for improving ML training by relying on silos of training data distributed amongst several clients. Clients locally train models and exchange the model updates with a central entity that aggregates them and broadcasts the result. As the values exchanged during the training of a federated model leak information about the training data [BDS+21, MSDCS19, NSH19, ZLH19], several works rely on secure aggregation techniques by using HE to protect confidentiality [FMM+21, ZPGS19, ZLX+20]. Also, ensuring the correctness of the aggregation is crucial for avoiding the introduction of backdoors, maliciously biased weights, and/or simply the deliberate exclusion of some client models by a malicious server [BDS+21, PFA22, TSSJ+22, XLL+19, ZFW+20].

Description. We adapt the federated averaging pipeline (FedAvg) proposed by McMahan *et al.* [MMR+17] for MNIST digit recognition to a cross-silo setting where clients trust each other but not the aggregator. This pipeline employs a simple multi-layer perceptron with two hidden layers of 200 units using the ReLu activation function (199, 210 parameters in total). The dataset is split across 100 clients, and 10 of them are randomly polled at each epoch for the federated averaging. Weights are locally quantized, by each client, to integers (16-bits as in [ZLX+20]) before being sent encrypted to the aggregator. After secure aggregation, the global weights are then decrypted and de-quantized by the clients and used for the next epoch. After 100 epochs, we obtain an accuracy of 91% (achieving higher accuracy with more iterations or different parameterization is out of the scope of this work). As all the clients trust each other, one client is in charge of generating the cryptographic keys and uses a secure channel (such as TLS) to share them via the server with the other clients. As the local training datasets are never shared, VERITAS enables result verification without needing the input data.

Parameterization. We use the HE parameters ($\log N$=15, $\log q$=700) and report results per epoch. Clients require 4.3s to encode and encrypt their local models, and the server requires 0.2s to aggregate them. The client decryption/decoding requires 214ms and the client egress/ingress communication is 44/44MB. We set the authenticator parameters to

λ=64 for REP and $\log t$=55 for PE.

Results. Figure 3.7e shows that, due to the huge volume of data (199,210 model parameters encrypted in 7 ciphertexts per client), REP creation is about 60× slower than the baseline, thus making the PE encoding much more efficient (255s vs. 11s, resp.). Similarly, the non-existent multiplicative complexity of secure aggregation showcases that PE is more efficient for the server; it introduces a 1.5× server evaluation overhead, compared to REP's 80× one (0.4s vs. 13s). Accordingly, the client verification with PE is only 2.2× slower than the baseline, whereas REP introduces an overhead of 70× more (0.7s vs. 15s resp.). The overall communication overhead of REP is more significant than PE (2.4GB vs. 88MB).

3.6.4 Comparison With Prior Work

A direct performance comparison between VERITAS and the related work is not straightforward due to (i) the different models employed for verifying the integrity of homomorphic computations, (ii) the related work's limitations regarding the homomorphic operations supported (see Table 3.1), and (iii) the lack of implemented solutions. For instance, Fiore *et al.* [FNP20] cannot support the HE parameterization used in §3.6.3 and Bois *et al.* [BCFK21] cannot evaluate rotation or relinearization operations; neither work has a public implementation. We implemented Rinocchio [GNSV21] and discovered that it can practically handle only arithmetic operations: To support the rounding operations required by modern HE schemes for tensoring and rotations (key-switching) (see §3.3.1), Rinocchio requires emulating multiple rings which would significantly slow down its performance. Thus, we evaluate Rinocchio only on the federated learning use-case (§3.6.3.5). Although it yields a succinct proof (118kB), Rinocchio is computationally more expensive than our solution (23s and 8s for the Create and Verify phases resp. compared to 0.4s and 0.7s for VERITAS) – and assuming the already expensive setup phase (∼1h). Only the early work of Fiore *et al.* [FGP14] has been officially implemented. As their work is limited to quadratic functions and cannot support rotations, we can compare it with VERITAS only on the ride-hailing (§3.6.3.1) and federated learning (§3.6.3.5) use-cases. For a similar HE parameterization, we estimate that their verification would take 180ms for the ride-hailing (resp. 4.1s for federated learning), whereas VERITAS requires 50ms with the best encoding (resp. 0.7s for federated learning). Similarly, following Fiore *et al.*'s evaluation, the server's ride-hailing evaluation takes 0.8s (resp. 0.1s for federated learning), which is very similar to VERITAS (0.5s and 0.4s, resp.). We stress again that none of the other use cases evaluated in §3.6.3 can be efficiently achieved by using prior work.

3.6.5 Evaluation Take-Aways

Our evaluation shows that VERITAS is suitable for various applications and yields different trade-offs depending on the setting and the security requirements. When the volume of input data is small and REP's (§3.4) extended vector fits in a single ciphertext, then REP outperforms PE (§3.6.3.1,3.6.3.4). When the application requires fully packed ciphertexts, PE (§3.5) reduces the load of the client by an order of magnitude (§3.6.3.3,3.6.3.5). Furthermore, our analysis demonstrates that PE yields a significant overhead for the server when the circuit multiplicative depth is large (§3.6.3.3). This is alleviated by ReQ (§3.5.4) which further improves the client verification time when interactivity is possible. With some computation overhead at the server, the PoC (§3.5.3) reduces the volume of data sent back to the decrypting client (§3.6.3.4). Both ReQ and PoC optimizations improve the computation and communication overheads at the client, thus making the use of VERITAS suitable for constrained clients. Our evaluation also shows that VERITAS can cope with various use-cases relying on complex homomorphic operations that were not supported by the state of the art [BCFK21, FNP20, GNSV21]. Overall, we observe that it enables homomorphic computation verifiability with acceptable overheads for the client and server (and even more considering the overhead of the HE pipeline itself; see Appendix 3.7.2). For instance, it enables the verifiability of a disease prediction result on genomic data, with less than 3× computation and communication overhead for the client and the server, compared to the HE baseline.

3.7 Discussion

We now discuss various aspects related to the authenticators and the developed library VERITAS.

3.7.1 Challenge Verification

Both authenticators operate in the *designated verifier* setting, *i.e.,* they require the challenge computation during the verification procedure by the client (or any holder of the secret key). In some cases, this might be a limitation because the client has to spend resources on this computation (§3.6.3). Nonetheless, these challenges do not leak any information about the initial data (recall that they are computed by the keyed PRF and the identifiers), and they can be offloaded to another entity that does not collude with the server. Hence, standard verifiable computation techniques, such as interactive proofs [GKR15], SNARKs [BBB+18] or STARKs [BSCR+19], could be employed to ensure the correct computation on these challenges. With state-of-the-art SNARKs *e.g.,* [BBB+18, MBKM19, PHGR13], this would lead to a sublinear verification complexity in the size of the circuit, albeit with a modification in our system model (§3.2).

3.7.2 VERITAS' Overhead in Perspective

As observed in Figure 3.7, compared to the HE baseline, the use of VERITAS introduces an overhead factor between 0.5× and 38× for the client upon offloading and decryption (and verification) and between 0.1× and 3× for the server's evaluation. Although non-negligible, this overhead is actually much smaller than the one already induced by the use of HE. Indeed, compared to a client running the computations directly on its local data, offloading encrypted data is between one and two orders of magnitude more expensive (10–100×). At the same time, evaluating the homomorphic circuit on encrypted data at the server is between one and four orders of magnitude slower (10–15,000×). For example, in the disease susceptibility use-case (§3.6.3.2), the client's offloading work is 600× more expensive than running the computation locally, whereas its decrypting work is 186× more costly. Similarly, the server's operations under encryption are 15,000× more expensive compared to the plaintext evaluation which is much more significant than the overhead created by our encodings. Therefore, VERITAS' overhead for both the client and the server is close to a negligible addition to the cost of HE. Further improvements that lower the overhead of the HE schemes (*e.g.,* hardware accelerators [SFK+22, GVBP+22]) would directly translate to a reduction in VERITAS' overhead as well.

3.7.3 Verification Outcome

Using VERITAS, a client is able to detect with very high probability if the server misbehaves, hence malicious servers (§3.2) are strongly discouraged from tampering with the requested computations and from using VERITAS as a *decryption oracle*. By purposefully modifying the ciphertext, a nosy server could learn a bit of information about the plaintext based on whether the verification check passes or not. Hence, the client's behavior, with respect to the server, should not be affected by the verification bit. Additional system-level countermeasures, *e.g.,* fresh key generation after every verification or dummy computation requests, could be employed orthogonally to VERITAS to hide the verification bit. Some solutions, which directly provide the correctness of computation on the ciphertext space [BCFK21, FGP14, FNP20] are not constrained to concealing the verification outcome. However, these are limited in the admissible computations and hinder the functionalities of the HE scheme.

3.8 Summary

In this work, we have proposed two error-detection encodings that can be used to verify the integrity of homomorphic computations. With these encodings, a client can augment existing privacy-preserving pipelines based on homomorphic encryption with computation

verification at a minimal cost; this way, their threat model can be easily extended from an honest-but-curious adversary provider to a malicious one. We have also provided VERITAS, a new open-source library that implements our solution and we demonstrated its practicality and versatility on several uses-cases.

Chapter 4

Securing HE-based MPC against Malicious Adversaries

Multiparty variants of FHE have recently emerged to improve the efficiency and practicality of computing pipelines. However, existing Multiparty FHE (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.

Contents

4.1 Overview

Multiparty Fully Homomorphic Encryption (MFHE) schemes [LATV12, CDKS19, AJLA+12, BGG+18, Par21, KLSW21, AH19, MTPBH21, 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 [CDKS19, FMM+21, ATP21, FTPP+21, SPTP+21], medical analytics [FTPR+21, SBTP+22, CEBC22, CFC+22], and financial audits [YWZ+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 [AJLA+12, MW16], the existing literature has focused mostly on the passive adversary model [LATV12, CDKS19, Par21, MTPBH21, MBH22] and any proposals accounting for malicious adversaries have remained theoretical [AJLA+12, MW16].

A natural approach to secure MFHE pipelines against malicious adversaries is to employ techniques based on non-interactive zero-knowledge (NIZK) proofs [AJLA+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 [BCOS20, 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 [CKKS17]) 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 [BEHZ17] and NTT-transformations [AMBG$^+$16, PG12, GFS$^+$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 [BCOS20], and show that PELTA achieves one to two orders of magnitude faster prover runtimes with 15 times smaller setup time.

4.2 Background on 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 [JWB$^+$17, RTPM$^+$18], financial [BCD$^+$09, BTW12, PAD$^+$23], and law enforcement [BJSV15] application domains). Multiparty FHE (MFHE) enables the evaluation of functions over encrypted data in these scenarios, while enforcing *joint* cryptographic access-control over the

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

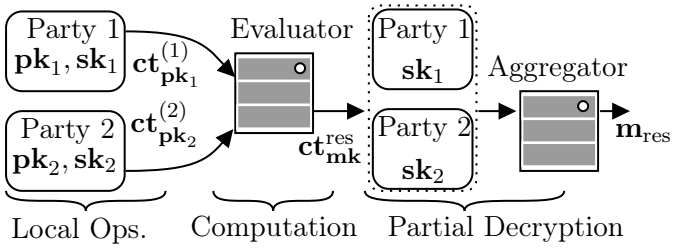


Figure 4.1: An illustration of an MkHE pipeline.

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 engaging in a multiparty decryption protocol. Modern MFHE schemes are based on the ring-learning-with-errors (RLWE) problem [CDKS19, MTPBH21, Par21, KLSW21, AH19], and recent literature has employed such schemes to build practical systems for, *e.g.,* distributed analytics [FTPR$^+$21, CSS$^+$22, YZW$^+$22], and federated machine-learning [CDKS19, SBTP$^+$22, SPTP$^+$21, ATP21, XHX$^+$22, XLG$^+$23, AHWC19].

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 4.3.2.

4.2.1 Multi-key FHE (MkHE)

In multi-key FHE [LATV12, CCS19, CDKS19, 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 4.1. 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 least linearly with the number of parties) and induce a significant overhead compared to single-party FHE schemes [LZY$^+$19, YKHK18].

4.2.2 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 [AJLA+12, BGG+18, MTPBH21, 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 4.2. 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 of parties, making them a much more practical option for settings with a fixed set of participants.

4.2.3 Multi-group FHE (MgHE)

Multi-group FHE [Par21, KLSW21, 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.

4.3 Towards Malicious Multiparty FHE

In this section, we present the system and threat models considered in our work (§4.3.1). Then, we propose the first security-oriented systematization of the MFHE families (§4.3.2). 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 (§4.3.3).

4.3.1 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, AJLA+12, BGG+18,

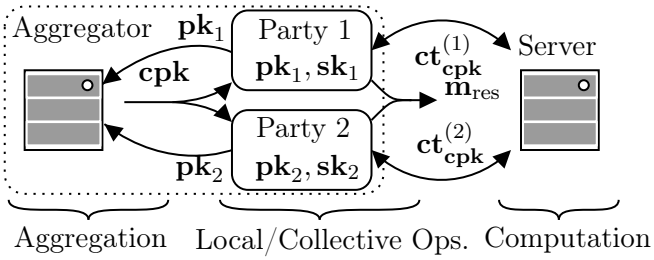


Figure 4.2: An illustration of a ThHE pipeline.

MTPBH21], 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. 4.1 and 4.2). 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 4.1 and 4.2 for interaction illustrations).

Threat Model. We consider that the parties are in an *anytrust* model [WCGFJ12], *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.

4.3.2 Systematizing Multiparty FHE

To secure MFHE pipelines in the presence of malicious adversaries (§4.3.1), 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 §4.2, under a unified model. We observe that the currently implemented RLWE-based MkHE, ThHE, and MgHE schemes [CDKS19, MTPBH21, Par21, KLSW21, 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 §4.3.2.1. 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 §4.3.2.2. 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.

4.3.2.1 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 (§4.2), 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:

$$\mathbf{b}_i = \mathbf{a} \cdot \mathbf{s}_i + \mathbf{X} + \mathbf{e}_i \tag{4.1}$$

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 [AJLA+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. (4.1) 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 [MTPBH21]. These operations are special cases of Equation 4.1 and they can be expressed by setting $\mathbf{X}$ accordingly. As such, they can also be decomposed following **Step 1** and **2**, described above.

4.3.2.2 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 §1.3, 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 §1.3), *i.e.,* it is very similar to Eq.4.1. 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 §4.7.2) [ENS20, LN17, BCOS20, 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, CKP$^+$23, FGP14, BCFK21, FNP20, NLDD21] (see §4.7.1). 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.

4.3.3 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** (§4.3.2.1), 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.* [AJLA$^+$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 §4.2), 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 §4.4. 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 §4.5.

4.4 Verification of MFHE Local Share Generation

As discussed in §4.2 and §4.3.2, at the core of MFHE interactive operations is the local share generation performed by the parties (*i.e.,* **Step 1** in §4.3.2.1). 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 §4.3.2). 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 (§4.4.1) and provide background information on lattice-based commitments (§4.4.2). Then, we present our techniques for assembling, in a practical manner, the statement required to verify the correct generation of RLWE samples (§4.4.3) and the ways this can be extended to various MFHE operations, *e.g.,* local key generation, collective decryption, public-key switching, and bootstrapping (§4.4.4). 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. 4.1.

4.4.1 Challenges of Verifying MFHE Local Share Generation

Whereas several works, *e.g.,* [DPLS19, BCOS20, BLS19, ENS20, LNP22, LLNW18], propose solutions to verify variants of Eq. 4.1 by using proof systems such as Bulletproofs [BBB$^+$18] and Aurora [BSCR$^+$19], or commitments and proofs [BDL$^+$18a], 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 [BBB$^+$18, BSCR$^+$19, PHGR13]).
4. **Infinity Norm:** As the secret and noise bounds need to be exactly evaluated in infinity norm, the proofs cannot rely on approximate results or optimized ℓ_2 variants, as in prior work [LNP22].
5. **Large Polynomial Noises:** The noise polynomials are sampled from a Gaussian distribution with bounded infinity norm. Depending on the type of noise, this bound can be large and not easily handled by proof systems, *e.g.,* [BSCR$^+$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.,* [BBB$^+$18, BSCR$^+$19, PHGR13], 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. 4.1, 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.

4.4.2 Background: Lattice-Based Commitments

Lattice-based commitments, *e.g.,* [Ajt96, KTX08, BDL$^+$18a], rely on the hardness of lattice-based problems such as the module short integer solution problem (MSIS) and the module learning-with-error problem (MLWE) [LS15, DKL$^+$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 [BDL+18a] 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, ESLL19a, 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 [BDL+18a] lattice-based commitment scheme. Suppose the module ranks κ_{sis} and λ_{lwe} 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$(\kappa_{\text{sis}}, \lambda_{\text{lwe}}, \ell) \rightarrow \mathbf{B}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_\ell$: Sample a uniformly random matrix $\mathbf{B}_0 \xleftarrow{\$} \mathcal{R}_q^{\kappa_{\text{sis}} \times (\lambda_{\text{lwe}} + \kappa_{\text{sis}} + \ell)}$ and vectors $\vec{\mathbf{b}}_1, \ldots, \vec{\mathbf{b}}_\ell \xleftarrow{\$} \mathcal{R}_q^{\lambda_{\text{lwe}} + \kappa_{\text{sis}} + \ell}$. Output them as public parameters.

Com.Commit$(\vec{\mathbf{m}}) \rightarrow \vec{\mathbf{t}}$: First, sample a random vector $\vec{\mathbf{r}} \xleftarrow{\$} \chi^{(\kappa_{\text{sis}} + \lambda_{\text{lwe}} + \ell)}$ 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 [BDL+18a, ALS20, ENS20, LNS21a]. Whereas the original construction requires a specific parameterization of the polynomial rings that is incompatible with NTT-friendly prime modulus [BDL+18a], 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.

4.4.3 Practically Verifying RLWE Samples

Here, we present the way we combine lattice-based proofs to address the challenges outlined in §4.4.1 and verify the correctness of Eq. 4.1. 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 §4.4.4, we will show how to extend our techniques to other MFHE

operations (that instantiate Eq. 4.1 with a non-zero polynomial $\mathbf{X}$).

Verifying the Linear Relation. Recall that, during **Step 1** of the MFHE interactive operations (§4.3.2.1), 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.,* [BDL+18a, 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.* [LNSW13] 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 $k{=}\lfloor \log(B) \rfloor + 1$), 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 bound on the norm of the noise polynomials exactly.

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.

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

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: $\mathbf{p}_0, \mathbf{c}_{\text{ajtai}} \in \mathcal{R}_q$

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

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:

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

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{Z}_p^{\kappa_{\text{sis}} N \times 2N}$. Similarly to del Pino *et al.* [dPLS18], we rewrite it in $\mathbb{Z}$ to ensure no wrap-around $\mod q_j$ as

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

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, $\|\mathbf{c}_{\text{ajtai}} + \vec{\mathrm{k}} \cdot p\|_\infty \leqslant 2Np$. 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 [LNS20, LNS21a] (see Appendix §B.4). Note that, in practice, both the ternarity and the 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 $(\mathbf{p}_0, \mathbf{c}_{\text{ajtai}})^T = \mathbf{Aw}$,

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:

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

with $\vec{\mathbf{p}}_1$ 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 §4.4.4, 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].

4.4.4 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 (§4.4.4.1), collective decryption (§4.4.4.2), public-key switching (§4.4.4.3), and bootstrapping (§4.4.4.4). 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 §4.2), all use *smudging noise* [AJLA+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 (§4.4.3) is impractical: Indeed, such a decomposition would introduce additional polynomial inputs into the proof system (as many as the decomposition size), and hence affect both its runtime and proof size. To avoid this issue, we propose a novel trick. Instead of

[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.

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 [AJLA+12]. Then, we use an observation stemming from the RLWE problem [LS15, DKL+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.

4.4.4.1 Local Key Generation

The MFHE local key generation operation comprises the generation of a private/public key pair (SecKeyGen and EncKeyGen – §4.3.2), 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 [MTPBH21] 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 the 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 §4.5, this statement can be combined with Statement 7 to verify the correctness of the MFHE collective key-generation protocol.

4.4.4.2 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 [AJLA+12]. As seen above, we rely on 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

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: $\|\vec{\mathrm{k}}_1\|_\infty \ll q$

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 $j \in [0, L]$).

4.4.4.3 Collective Public-key Switching

In ThHE and hybrid settings (§4.2), 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 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.

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

4.4.4.4 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.

4.5 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** – §4.3.2.1). A malicious aggregator can tamper with the aggregation

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

operation and harm the confidentiality of the honest parties and/or the correctness of the MFHE pipeline (see §4.3.2); 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 (§4.4), 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 (§4.3.1), existing protocols for maliciously secure aggregation based on splitting trust among multiple servers [RSWP23] 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 §4.3.2, 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}).

Verifiable Aggregation Protocol. As in our threat model all but one MFHE parties

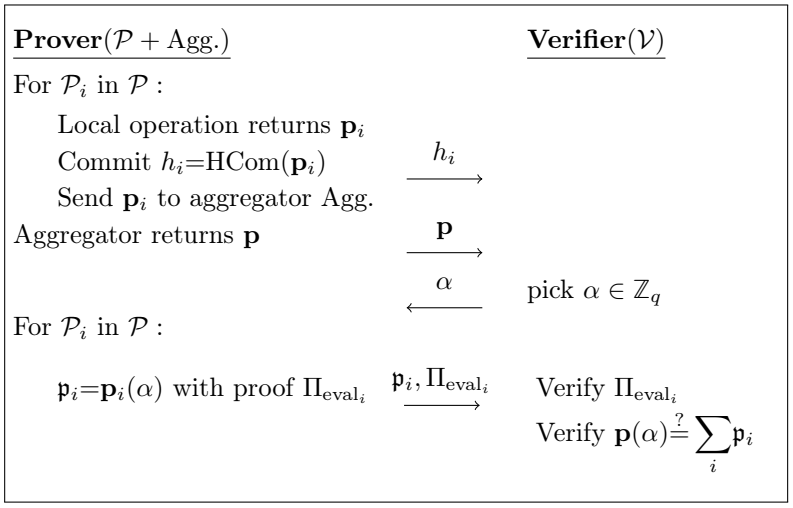


Figure 4.3: An illustration of the verifiable aggregation interactive protocol. In practice, this protocol is made non-interactive using the Fiat-Shamir heuristic, i.e., the challenge α is generated in the random oracle model (ROM) from the parties' commitments h_i and the aggregate $\mathbf{p}$. Without 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.

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 (§4.4) 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 4.3. 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 (§4.4.4). In particular, when creating a proof of correct local share generation, each party also appends a linear relation to the statements of §4.4.4 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 §4.6, 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

Statement 7: Verifiable Aggregation

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

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 [WTS+18], FRI [BSBHR18], 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.

4.6 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.

4.6.1 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 [BDL+18a] 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 (§4.4 and §4.5). Our code is available in an open source repository [LDS23].

4.6.2 Experimental Setup

We focus on key MFHE protocols, i.e., key generation (§4.4.4.1), collective decryption (§4.4.4.2), public-key switching (§4.4.4.3), and collective bootstrapping (§4.4.4.4); these

play a critical role in MFHE pipelines (both MkHE and ThHE – see §4.3.2). 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_{sis}=1 and λ_{lwe}=1 for a $\mathfrak{R}_q$ of degree d=2^{13}. Unless otherwise specified throughout the experiments, we consider FHE parameters with a 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 B.2) 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 optimized 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 §4.4 and §4.5. 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.* [BCOS20] that relies on the Aurora [BSCR$^+$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.

4.6.3 Performance Analysis

We first compare the performance of PELTA and the two baselines for the local key-generation protocol (§4.4.4.1). Table 4.1 shows that PELTA outperforms both 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

Table 4.1: Performance results of the local key-generation protocol (§4.4.4.1) 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
[BCOS20]	>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

order of GBs (see [SPTP+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 [BCOS20]. 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 4.2 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 4.2(a)), and 67× and 42× more efficient for the collective public-key switching and bootstrapping protocols, resp. (Tables 4.2(b) and 4.2(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 [BCOS20] 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 4.2 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 §4.4 as discussed in §4.5. Table 4.3 shows the performance results for the collective key-generation protocol (Statements 2 and 7), for [BCOS20] 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 4.1). Although the runtime of PELTA is almost not affected, the inclusion of the aggregation relation significantly impacts Boschini *et al.*'s approach [BCOS20]. Indeed, for Aurora, the new polynomial evaluations (used for our compression technique in §4.5) 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 **A**, §4.4.3 and Figure B.1). As a result, the aggregation comes almost for *free* due to our trick of using a polynomial evaluation to verify it (§4.5).

Table 4.2: Performance results of various MFHE protocols with $\log N{=}13$ over a single sub-ring of the RNS representation.

(a) Collective decryption (§4.4.4.2).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[BCOS20]	>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 (§4.4.4.3).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[BCOS20]	>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 (§4.4.4.4).

	Setup	Prover (s)	Verifier (s)	Proof (MB)
[BCOS20]	>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

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 [ACC+18]. For simplicity, we consider a single modulus, as we experimentally observe a linear correlation with the number of sub-rings (see Table B.1 in Appendix §B.2). As expected, we observe in Table 4.4 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 4.1–4.3 show that they are larger than Boschini *et al.* [BCOS20]. Nonetheless, PELTA's proof size can be reduced and made on par with [BCOS20] 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 B.2 in Appendix B.3). 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 4.3: 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)
[BCOS20]	>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

4.7 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 (§4.3.2.2) are required. In particular, countermeasures that ensure (i) the validity of the homomorphic evaluation (§4.7.1), as well as (ii) the trustworthiness of the input provided to the computation (§4.7.2), should be incorporated. As discussed in §4.3.2.2, such countermeasures are orthogonal and complementary to PELTA.

4.7.1 Homomorphic Evaluation Correctness

Verifying the correctness of the homomorphic evaluation (Eval– §4.3.2.2) 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 [LDPW14, LWZ18, LWX22, CMP14, TPD16, LPJY13, CHH$^+$18, JY14]. Whereas our work presented in Chapter 3 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, BCFK21]. These solutions cannot, however, cope with crucial FHE operations such as key-switching and relinearization (§1.3) 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.,* [NLDD21], introduce different trust assumptions and are potentially prone to integrity attacks [FYDX21].

Table 4.4: PELTA's performance (single-th.) for EncKeyGen (§4.4.4.1) 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

Therefore, we argue that without novel and efficient techniques for proving the homomorphic evaluation correctness, a simple approach to secure this part of MFHE pipelines (*i.e.,* Eval – §4.3.2.2) 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 (§4.3.1), 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.

4.7.2 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– §4.3.2.2). Although ensuring the correct encryption operation of a valid input can be achieved using techniques similar to §4.4.3 and to prior work [ENS20, LN17, BCOS20, DPLS19], verifying (plaintext) input correctness is a more challenging problem that requires specialized solutions. We could use commitments and range proofs [Gro11, BBB$^+$18, CKLR21, 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 [JOB$^+$18, BNL12]). Therefore, additional techniques that can verify input correctness based on, *e.g.,* statistical tests [CSC$^+$23], proofs of correct computation on authenticated data [BBFR15], or authenticity checks on encrypted data [CPTPH21a], are required.

4.8 Summary

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.

Chapter 5

Verifiable Encryption and Trustworthy Data Release

In recent digital scenarios, users share their personal data with service providers to obtain some utility, *e.g.,* access to high-quality services. Yet, the induced information flows raise privacy and integrity concerns. Consequently, cautious users may want to protect their privacy by minimizing the amount of information they disclose to curious service providers. Service providers are interested in verifying the integrity of the users' data to improve their services and obtain useful knowledge for their business. In this chapter, we present a generic solution to the trade-off between privacy, integrity, and utility, by achieving authenticity verification of data that has been encrypted for offloading to service providers. Combining lattice-based homomorphic encryption and commitments, as well as zero-knowledge proofs, our construction enables a service provider to process and reuse third-party signed data in a privacy-friendly manner with integrity guarantees. We evaluate our solution on different use cases such as smart-metering, disease susceptibility, and location-based activity tracking, thus showing its versatility.

Contents

5.1 Overview

As discussed in the Introduction, more and more data flows operate in the three-party model depicted in Figure 5.1: a *user* obtains data from a *data-source* and subsequently shares it with a *service provider* to obtain some utility from the data. This is for example the case in smart metering, personalized health, and location-based activity tracking as we will see in this chapter.

To avoid any unintended leakage from her data and still enable computations to be executed, the user employs FHE. Still, some scenarios (*e.g.,* smart metering billing) require the service provider to access in clear the result of the computation. However, as we have seen in Chapters 1 and 2, by default, FHE does not provide any guarantee of correctness in the presence of malicious users.

Thus in this chapter, we investigate how to enable both *integrity* protection and *confidentiality* without hindering *utility*. We consider a model comprising (i) a malicious user, and (ii) honest-but-curious service providers and data sources. The data source is in charge of

honestly generating and authenticating data pertaining to the user. This certification should require minimal to no changes to the data source: using only deployed hardware and software infrastructure. The user should be able to offload their protected data to service providers (*i.e.,* transfer a copy of the data only once) in such a way that their privacy is preserved, the data integrity can be verified, and various flexible computations are feasible.

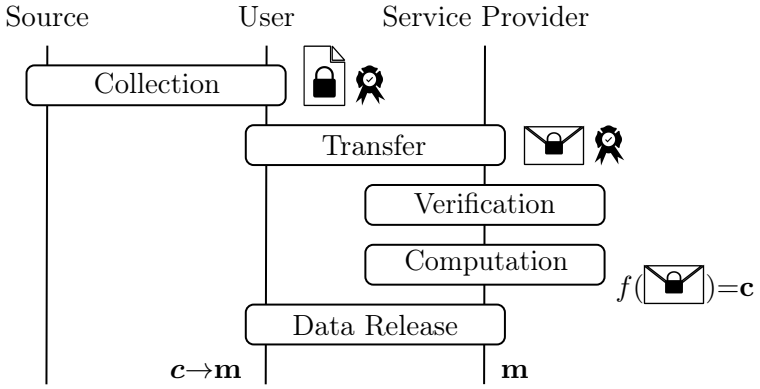


Figure 5.1: Three-party model and their interaction phases. 🗋 is the private information authenticated with 🎖. The user protects it via 🔒 to get ✉. The service provider computes $f(\cdot)$ on the protected data and obtains an output which is revealed as $\mathbf{m}$.

A simple solution is to establish a direct communication channel between the data source and the service provider. This way, the data source could compute the operations queried by the service provider on the user's data. However, this would prevent the user from remaining in control of her data and require the data source to bear computations that are outside of its interests. Another approach is to let the data source certify the user's data by using specialized digital signature schemes such as homomorphic signatures [BF11, CF13, CFGN14, CFN18, GVW15] or homomorphic authenticators [ABC$^+$15, FMNP16, GW13, SBB19]. Thus, the user could locally compute the queried operation and provide the service provider with the result and a homomorphic signature attesting to its correct computation on her data. However, this would require software modifications at the data source, which would come at a prohibitive cost for existing services, and introduce significant overhead for the user.

As discussed in Chapter 2, solutions based on generic zero-knowledge proofs and SNARKs fall short of achieving practicality for this setting. In particular, ADSNARK [BBFR15], which addresses a problem closest to ours, does not support offloading of data under FHE and relies on different trust assumptions (*e.g.,* a trusted setup).

In this chapter, we propose a new construction called CRISP to solve this problem. It is generic, supports data offloading with minimal modification to existing infrastructures and relaxes the need for a trusted setup. Motivated by the need to protect users' privacy and by the offloading requirement to support multiple computations on their data,

CRISP relies on plausibly quantum-resistant lattice-based approximate homomorphic encryption (HE) primitives [CKKS17] that support flexible polynomial computations on encrypted data without degrading utility. To ensure data integrity, we employ lattice-based commitments [BDL+18a] and zero-knowledge proofs [CDG+17] based on the multi-party-computation-in-the-head (or MPC-in-the-head) paradigm [IKOS09], which enable users to simultaneously convince service providers about the correctness of the encrypted data, as well as the authenticity of the underlying plaintext data, using the deployed certification mechanism.

We evaluate our solution on three use cases covering a wide range of applications and computations: smart metering, disease susceptibility, and location-based activity-tracking. Our experimental results show that our construction introduces acceptable computation overhead for users to privately offload their data and for service providers to both verify its authenticity and to perform the desired computations. The magnitude of the communication overhead compared to the unprotected HE pipeline fluctuates between tens and hundreds of megabytes per proof and is highly dependent on the use case and its security requirements. To this end, in Section 5.6.5, we also present different optimizations that can reduce the proof size, thus making our construction more practical for real-life scenarios. Compared to the state of the art [BBFR15], we reach comparable performance and achieve complete outsourcing features with additional post-quantum security guarantees.

Our contributions are the following:

- A generic solution that enables privacy and integrity preserving computations in the three-party model of Figure 5.1, with minimal modifications of the existing infrastructure;
- the necessary primitives to achieve authenticity verification of homomorphically encrypted data in the random oracle model;
- an implementation of CRISP [LDS20] and its performance evaluation on various representative use cases that rely on different types of computations and real-world datasets.

5.2 Model

We describe the model, assumptions, and objectives of CRISP.

5.2.1 System Model

We consider three entities: a user, a service provider, and a data source, as depicted in Figure 5.1. The user obtains from the data source certified data about herself and/or her activities. She subsequently shares this data with the service provider to obtain some service. The user is interested in sharing (*i.e.,* offloading) her data while protecting her privacy, *i.e.,* she wants to have full control over it but still obtain utility from the service provider. The service provider is interested in (i) verifying the authenticity of the user's data, and (ii) performing on it multiple computations that are required to provide the service and/or improve its quality. The data source can tolerate only minimal changes to its operational process and cannot cope with any heavy modification to the underlying infrastructure and dependencies of the hardware and software. Finally, we assume the existence of a public key infrastructure that verifies the identities of the involved parties as well as secure communication channels between the user and the data source, and between the user and the service provider.

5.2.2 Threat Model

We present the assumed adversarial behavior for the three entities of our model with computationally bounded adversaries.

Data Source. The data source is considered honest and is trusted to generate valid authenticated data about the users' attributes or activities.

Service Provider. The service provider is considered *honest-but-curious*, *i.e.,* it abides by the protocol and does not engage in denial-of-service attacks. However, it might try to infer as much information as possible from the user's data and perform computations on it without the user's consent.

User. We consider a *malicious but rational* user. In other words, she engages in the protocol and will try to cheat only if she believes that she will not get caught – and hence be identified and banned – by the service provider. This type of adversary is also referred to as *covert* in the literature [AL07]. The user is malicious in that she might try to modify her data, on input or output of the data exchange, in order to influence the outcome of the service provider's computations to her advantage. Nonetheless, the user is rational, as she desires to obtain utility from the service provider and thus engages in the protocol. We assume that the key generation process has been executed honestly. As we have seen in Chapter 4, additional countermeasures could be in place to ensure that the keys are properly formed.

5.2.3 Objectives

The main objective of our solution is to provide a proof of the correct encryption of authenticated data. Our construction should enable flexible computations in the considered three-party model. In particular, the user's privacy should be protected by keeping her in control of the data even in a post-quantum adversarial setting, and the service provider's utility should be retained by ensuring the integrity of the processed data. The above objectives should be achieved by limiting the impact on already deployed infrastructures, thus, by requiring only minimal changes to the data source's operational process. More formally, the desired properties are:

(a) **Utility**: Both user and service provider are able to obtain the correct result of a public computation on the user's private data;

(b) **Privacy**: The service provider does not learn anything more than the output of the computation on the user's private data;

(c) **Integrity**: The service provider is ensured that the computation is executed on non-corrupted data certified by the data source and that the result is correct; and

(d) **Deployability**: Only operational changes at the data source are possible; this is in contrast to data sources deploying new hardware components or novel software stacks for data certification.

5.3 Preliminaries

We introduce the cryptographic primitives used in Section 5.4 to instantiate CRISP. We refer the reader to Chapter 1.1 for the explanation of the mathematical notations used in this chapter.

5.3.1 Approximate Homomorphic Encryption

Cheon *et al.* recently introduced the CKKS cryptosystem [CKKS17] (improved in [CHK$^+$18b]), an efficient and versatile leveled homomorphic scheme for approximate arithmetic operations. An approximate homomorphic encryption scheme enables the execution of approximate additions and multiplications on ciphertexts without requiring decryption. The structure of the CKKS cryptosystem is very similar to BFV presented in Chapter 1. For χ_{enc} an error distribution over the polynomial ring $\mathcal{R}_q$, the encryption of a polynomial message $\mathbf{p}$ is:

Encryption($\mathbf{p}; \mathbf{pk_{HE}}$): For a message $\mathbf{m} \in \mathbb{Z}_t^N$ encoded as a polynomial $\mathbf{p}$, for $\mathbf{r}_0 \leftarrow \chi_{\text{enc}}$ and $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi_{\text{err}}$, output the pair of polynomials $\mathbf{ct}=(\mathbf{ct}_0, \mathbf{ct}_1)=\mathbf{r}_0 \cdot \mathbf{pk}_{\text{HE}} + (\mathbf{p} + \mathbf{e}_0, \mathbf{e}_1) \mod q$.

We present the CKKS scheme in more detail in Appendix C.4.

5.3.2 Zero-Knowledge Circuit Evaluation

As discussed in Chapter 2, verifiable computation (VC) and proof systems enable a prover to generate a proof of correct execution of a specific computation. Among the numerous lines of research, zero-knowledge circuit evaluation (ZKCE) protocols based on MPC techniques enable a user to prove the knowledge of an input that yields a public output on an arithmetic or Boolean circuit that implements a specific public function [CDG+17, GMO16, KKW18]. A circuit is defined as a series of gates connected by wires. Based on the multi-party computation (MPC) *in-the-head* approach from Ishai *et al.* [IKOS09], ZKCE techniques emulate players and create a decomposition of the circuit (see Figure C.1 in Appendix C.1). The secret is shared among the emulated players, who evaluate the circuit in an MPC fashion and commit to their respective states. The prover then reveals the states of a subset of players depending on the verifier's challenge. By inspecting the revealed states, the verifier builds confidence in the prover's knowledge.

In particular, ZKB^{++} [CDG+17] is a Σ-protocol for languages of the type $\{y \mid \exists x \text{ s.t.} y=\Phi(x)\}$, where $\Phi(\cdot)$ is the representation of the circuit. With randomized runs, the verifier builds confidence in the prover's knowledge of the secret. The number of iterations is determined according to the desired soundness: For instance, to prove the knowledge of a message that yields a specific SHA-256 digest, a security level of 128-bits requires 219 iterations. The proof size is linked to the number of iterations but also to the number of gates that require non-local computations (*e.g.,* AND for Boolean circuits, multiplication for arithmetic ones). Compared to earlier work, *i.e.,* ZKBoo [GMO16], ZKB^{++} reduces the proof size by not sending information that can be computed by the verifier. The security of ZKB^{++} is based on the quantum random oracle model.

Overall, it achieves the following properties:

(a) *2-privacy*, opening two out of the three players' views to the verifier reveals no information regarding the secret input,

(b) *special soundness*, a correct execution yields a valid witness with soundness error linked to the number of iterations, and

(c) *completeness*, an honest execution of ZKB^{++} ensures a correct output.

Compared to other approaches such as most succinct non-interactive arguments of

knowledge (SNARKs) [BSCG+13, CFH+15, GGPR13, Gro16, PHGR13], MPC-in-the-head ZKCE achieve good prover performance, does not require a trusted setup, and can be instantiated with plausibly quantum-security. As such, it has been used for several applications such as signatures [CDG+17] and lattice-based cryptography [BN20]. Because of its design, MPC-in-the-head ZKCE such as ZKB++ [CDG+17] are ideally combined with lattice-based FHE.[1]

5.3.3 BLDOP Commitment

The lattice-based commitment scheme due to Baum *et al.* [BDL+18a] presented in Section 4.4.2 in Chapter 4 is linearly homomorphic and, thus, compatible with the ZKCE presented above. In addition to the sigma protocols that we presented in the context of PELTA in Chapter 4, we present here a simple approximate bound proof due to Baum *et al.* [BDL+18b]. Given a commitment $\mathbf{c}$=BDLOP($\mathbf{m}, \mathbf{r}_c$) and a proof of correct opening, the prover additionally computes a commitment for a vector of small values μ as $\mathbf{t}$=BDLOP(μ, ρ) and commits to this commitment in an auxiliary commitment c_{aux}=$C_{\text{aux}}(\mathbf{t})$. The verifier selects a challenge $d \in \{0, 1\}$ and sends it to the prover who verifies its small norm and eventually opens c_{aux}. The prover also opens $\mathbf{t} + d \cdot \mathbf{c}$ to $\mathbf{z}$=μ+d·$\mathbf{m}$ and $\mathbf{r}_z$=ρ+d·$\mathbf{r}_c$. Upon reception, the verifier checks that BDLOP($\mathbf{z}, \mathbf{r}_z$)=$\mathbf{t}$+d·$\mathbf{c}$ and that the norms are small. More formally, the protocol works as follows:

1. The prover computes a commitment $\mathbf{t}$ = BDLOP(μ, ρ) for μ sampled uniformly in $\mathcal{R}_q$ and ρ a valid commitment noise subjected to $\|\mu\|_\infty \leq \beta_\mu$ and $\|\rho\|_\infty \leq \beta_\rho$ (see [BD16] for details about the bounds). Then, the prover commits to this commitment through an auxiliary commitment and sends c_{aux}=$C_{\text{aux}}(\mathbf{t})$ to the verifier.

2. The verifier randomly picks and sends a challenge d$\in\{0, 1\}$.

3. The prover checks the correctness of the challenge and computes z=$\mu + \text{d}m$ as well as $\mathbf{r}_z$=$\rho + \text{d}\mathbf{r}_c$ in order to provide a valid opening of $\mathbf{t}$+d$\mathbf{c}$. The prover aborts if the resulting commitment is not properly formatted. Otherwise, it sends z, $\mathbf{r}_z$, and the opening of the auxiliary commitment to the verifier.

4. The verifier checks the correctness of the opening and that the norm of z is small.

At the end of the protocol, the verifier is ensured that the norm of the secret m is below a specific threshold. This protocol can be made non-interactive with the Fiat-Shamir heuristic and iterated to increase the soundness. We refer the reader to [BD16] for more details.

[1] Baum and Nof [BN20] showed concurrently how to apply it to other lattice problems.

5.4 Architecture

We now present our construction that enables computations on third-party certified data in a privacy and integrity-preserving manner. It builds on (i) CKKS to encrypt the data and enable computations on it, and (ii) MPC-in-the-head and BDLOP commitments to simultaneously verify a custom circuit that checks the integrity of the data and its correct encryption. Its workflow is decomposed into five phases as in Figure 5.1: *collection*, *transfer*, *verification*, *computation*, and *release*.

Collection Phase: the user obtains data about herself or her activities from the data source, along with a certificate that vouches for its integrity and authenticity.

Transfer Phase: The user then encrypts the data, generates a proof for correct encryption of the certified data, and sends it with the ciphertexts to the service provider.

Verification Phase: The service provider verifies the proof in the verification phase.

Computation Phase: It performs the desired computations on it.

Release Phase: Both user and service provider communicate with each other to obtain the corresponding result in the release phase.

5.4.1 Collection Phase

In this phase, the user (identified by her unique identifier *uid*) collects from the data source certified data about herself or her activities. The data source certifies each user's data point $\mathbf{x}$ using a digital signature $\sigma(\cdot)$ that relies on a cryptographic hash function $\mathrm{H}(\cdot)$ to ensure integrity. We opt for SHA-256 as the hash function due to its widespread use as an accepted standard for hash functions [Dan15]; our solution works with any signature scheme building on it. For example, Bernstein *et al.* [BHK$^+$19] recently proposed a quantum-secure signature scheme employing SHA-256. In more detail, the data source generates a payload msg=$\{nonce, uid, \mathbf{x}\}$ and sends to the user a message M_0 defined by: M_0=$\{msg, \sigma(\mathrm{H}(msg))\}$.

5.4.2 Transfer Phase

In this phase, the user protects her certified data points with the CKKS homomorphic encryption scheme (see Section 5.3.1) and generates a proof of correct protection. To this end, CRISP employs a ZKCE approach to simultaneously prove the integrity of the underlying data and its correct encryption, *i.e.,* to convince a service provider that

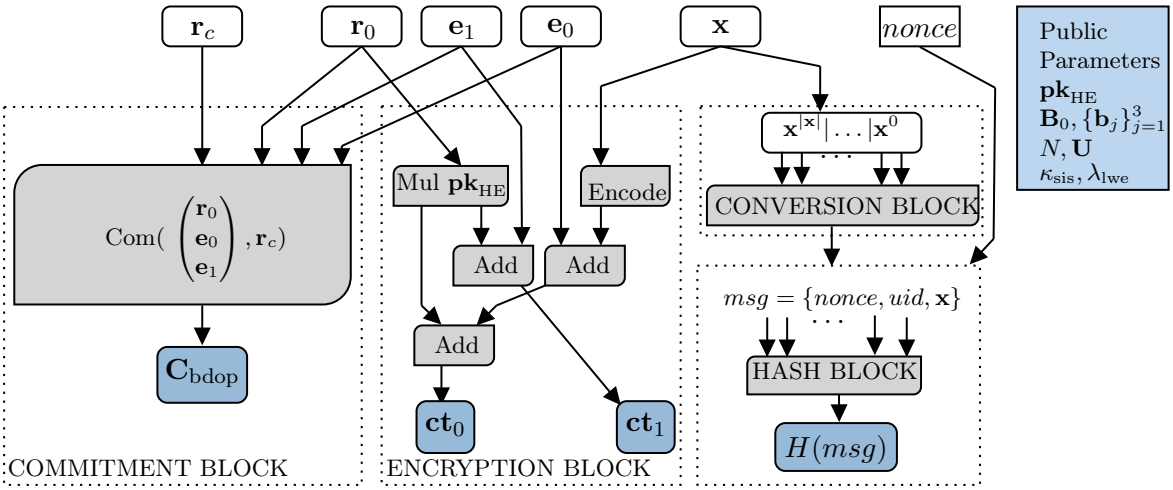


Figure 5.2: Overview of the verification circuit $\mathcal{C}$. Its inputs are denoted by rectangles and its outputs by rounded rectangles.

the noises used for encryption did not distort the plaintexts. In particular, the user evaluates a tailored circuit $\mathcal{C}$ (depicted in Figure 5.2) that (i) computes the encryption of the data with the CKKS scheme, (ii) generates BDLOP commitments to the noises used for encryption, and (iii) produces the hash digests of the messages signed by the data source to verify their integrity. For ease of presentation, we describe the circuit that processes one data point $\mathbf{x}$. However, this can easily be extended to a vector $\mathbf{d}$ obtained from multiple data points $\{\mathbf{x}_i\}$. The circuit's structure is publicly known and its public parameters are the encryption public information $\mathbf{pk}, \mathbf{U}, N$, the matrix $\mathbf{B}_0$ and vectors $\{\mathbf{b}_j\}_{j=1}^3$ used in the BDLOP commitment scheme and its parameters $\kappa_{\text{sis}}, \lambda_{\text{lwe}}$, and additional information such as the user's identifier. The circuit's private inputs are the user's secret data point $\mathbf{x}$ and nonce, the encryption private parameters r_0, e_0, and e_1, and the private parameters of the BDLOP commitment scheme $\mathbf{r}_c$. These inputs are arithmetically secret-shared among the three simulated players, according to the ZKB^{++} protocol. The outputs of the circuit are the ciphertext $\mathbf{ct}$, the commitment to the encryption noises $\mathbf{C}_{\text{BDLOP}}$=BDLOP$((\mathbf{r}_0, \mathbf{e}_0, \mathbf{e}_1)^T, \mathbf{r}_c)$, and the digest of the message $H(msg)$ signed by the data source.

For efficiency reasons, CKKS operates over the RNS decomposition of polynomial rings. Each sub-ring has a prime modulus which is NTT-friendly. While the original lattice-based commitments required specific moduli, the recent work from Attema *et al.* [ALS20] showed how to construct commitments over NTT-friendly polynomial rings used by encryption schemes such as CKKS.

Moreover, we note that our circuit requires computations to be executed on the underlying arithmetic ring $\mathbb{Z}_q$ used for the lattice-based encryption and commitment schemes, as well as a Boolean ring $\mathbb{Z}_2$ for the computation of the SHA-256 hash digests. We also design a

block that converts MPC-in-the-head arithmetic shares of the input data of the circuit into Boolean ones.

Overall, our circuit $\mathcal{C}$ consists of four blocks, shown in Figure 5.2: encryption, commitment, conversion, and hash block.

Encryption Block. This block operates in the arithmetic ring $\mathbb{Z}_q$ and takes as inputs the vector of integers in $\mathbb{Z}_q$ derived by quantization from the plaintext $\mathbf{x}$ produced during the data collection phase (see Section 5.4.1), as well as the encryption with private noise parameters $\mathbf{r}_0$, $\mathbf{e}_0$, and $\mathbf{e}_1$. It first encodes the secret input data $\mathbf{x}$ to a polynomial $\mathbf{p} \in \mathcal{R}_q$ before computing the ciphertext $\mathbf{ct}=(\mathbf{ct}_0, \mathbf{ct}_1)=\mathbf{r}_0 \cdot \mathbf{pk} + (\mathbf{p} + \mathbf{e}_0, \mathbf{e}_1) \mod q$. This step requires only affine operations that can be computed locally for each simulated player of ZKB^{++} protocol. The encryption block is depicted in the middle part of Figure 5.2.

Commitment Block. This block also operates in the arithmetic ring $\mathbb{Z}_q$; its inputs are the private parameters of the encryption (*i.e.,* $\mathbf{r}_0$, $\mathbf{e}_0$, and $\mathbf{e}_1$) and commitment (*i.e.,* $\mathbf{r}_c$) schemes. As the commitment scheme has the same external structure as the encryption one, this block operates equivalently and returns $\text{BDLOP}((\mathbf{r}_0, \mathbf{e}_0, \mathbf{e}_1)^T, \mathbf{r}_c)$, requiring only local operations at each simulated player. An overview of the commitment block is shown in the leftmost part of Figure 5.2.

Conversion Block. This block enables us to interface two types of circuits that would otherwise be incompatible when following a ZKCE approach. The main idea is to transform an arithmetic secret sharing into a Boolean secret sharing in the context of MPC-in-the-head. Let $[x]_B$ denote the Boolean sharing of a value x and $[x]_A$ its arithmetic one. An arithmetic additive secret sharing in $\mathbb{Z}_q$ splits x into three sub-secrets x_0, x_1, and x_2 such that $x=x_0+x_1+x_2 \mod q$. Let x_i^k, be the k-th bit of the arithmetic sharing of the secret x for player i. A Boolean sharing $[x]_B$ cannot be directly translated from $[x]_A$ as the latter does not account for the carry when adding different bits. Considering that the modulus q can be represented by $|q|$ bits, the conversion block generates $|q|$ Boolean sub-secrets $[y]_B^j=\{y_0^j, y_1^j, y_2^j\}_B$, such that

$$\forall j \in [1, |q|] : \; x^j = \bigoplus_{i=0}^{2} y_i^j,$$

where $\oplus$ denotes the XOR operation (*i.e.,* addition modulo 2), and x^j is the j-th bit of x. When designing such a block in the MPC-in-the-head context, we must make the circuit (2,3)-decomposable (see § 5.3.2) and ensure the 2-privacy property, *i.e.,* revealing two out of the three players' views to the verifier should not leak any information about the input.

To reconstruct the secret in zero-knowledge and obtain a bit-wise secret sharing, the procedure is as follows: For every bit, starting from the least significant one, the conversion block computes (i) the sum of the bits held by each player, plus the carry from the previous

bits, and (ii) the carry of the bit. The computation of the carry requires interaction between the different players (*i.e.,* making the operation a "multiplicative" one), hence we design a conversion block with a Boolean circuit that minimizes the number of multiplicative gates.

More precisely, we design a bit-decomposition block for MPC-in-the-head building on Araki *et al.*'s optimized conversion [ABF+18] between a power-of-two arithmetic ring and a Boolean ring. Let Maj$(\cdot)$ be the function returning the majority bit among three elements. Then, the conversion circuit, for every bit $k \in [1, |x|]$, does the following:

1. locally reads $[\alpha_k]_B = \{x_0^k, x_1^k, x_2^k\}$ (*i.e.,* for each player);
2. computes the first carry $[\beta_k]_B$ amongst those inputs:
$$\beta_k = \text{Maj}(x_0^k, x_1^k, x_2^k) = (x_0^k \oplus x_2^k \oplus 1)(x_1^k \oplus x_2^k) \oplus x_1^k;$$
3. computes the second carry $[\gamma_k]_B$ amongst those inputs with $\gamma_0 = \beta_0 = 0$:
$$\gamma_k = \text{Maj}(\alpha_k, \beta_{k-1}, \gamma_{k-1}) = (\alpha_k \oplus \gamma_{k-1} \oplus 1)(\beta_{k-1} \oplus \gamma_{k-1}) \oplus \beta_{k-1};$$
4. sets the new Boolean sharing of the secret to
$$[y]_B^k = [\alpha_k] \oplus [\beta_{k-1}] \oplus [\gamma_{k-1}].$$

To the best of our knowledge, this is the first time a bit-decomposition circuit is used for MPC-in-the-head, which enables to interface circuits working in different rings.

Hash Block. This block uses the SHA-256 circuit presented in [GMO16] to compute the hash digest of the message $msg = \{nonce, uid, \mathbf{x}\}$ signed by the data source in the collection phase.

Full Circuit. With the above building blocks, and following the ZKB^{++} protocol, the user generates a proof that can convince the service provider that she has not tampered with the data obtained by the data source.

Furthermore, using BDLOP's approximate bound proof protocol (see Section 4.4.2 in Chapter 4 and Section 5.3.3) the user produces a proof of correct encryption, *i.e.,* that the encryption noise has not distorted the underlying plaintext. The cryptographic material of the combined proofs (ZKCE & BDLOP) is denoted by Π: *i.e.,* it contains the cryptographic material to verify the circuit C as well as the BDLOP bound proof (see Section 4.4.2 in Chapter 4). At the end of the transfer phase, the user sends to the service provider the message:
$$M_1 = \{\mathbf{ct}, \mathbf{C}_{\text{BDLOP}}, \Pi, \text{H}(msg), \sigma(\text{H}(msg))\}.$$

Since the publication of our work, this approximate bound proof has been improved alleviating the need for repetition [LNS20]. It would be interesting to see the impact on

the performance this new protocol brings as it trades-off computation for communication costs.

5.4.3 Verification Phase

Upon reception of a message M_1, the service provider verifies the signature using the provided hash digest. If satisfied, it verifies the proof Π by first evaluating the circuit $\mathcal{C}$ following the ZKB^{++} protocol and then checking the bound proof for the encryption noises. Hence, it is assured that $\mathbf{ct}$ is the encryption of a data point $\mathbf{x}$ giving the hash that has been certified by the data source.

5.4.4 Computation Phase

Using the homomorphic capabilities of the CKKS encryption scheme, the service provider can perform any operation with a bounded predefined multiplicative depth (and arbitrary depth, with bootstrapping [CHK$^+$18a]) on validated ciphertexts received by the user. In particular, CKKS enables the computation of a wide range of operations on ciphertexts: additions, scalar operations, multiplications, and a rescaling procedure that reduces the scale of the plaintexts. Those functions enable the computation of polynomial functions on the ciphertexts. Moreover, it supports the evaluation of other functions such as exponential, inverse or square root [CKKS17, CHK$^+$18a, CKK$^+$19], by employing polynomial approximations (*e.g.,* least squares). Hence, the service provider can independently compute any number of operations on the user's encrypted data simply requiring interactions with the user to reveal their outputs (see Section 5.4.5).

5.4.5 Release Phase

At the end of the computation phase, the service provider holds a ciphertext of the desired output that can only be decrypted by the holder of the secret key. To this end, the service provider and the user engage in a two-round release protocol, which ensures the service provider that the decrypted output is the expected result of the computation on the user's data. The release protocol is depicted in Figure 5.3 and detailed next.

Let $\mathbf{ct}'$ denote the ciphertext obtained by the service provider after performing computations on validated ciphertext(s), and $\hat{m}$ the corresponding plaintext. First, the service provider informs the user of the computation $f(\cdot)$ whose result it wants to obtain. Then, the service provider homomorphically blinds $\mathbf{ct}_\psi$ by applying the function $B_{\boldsymbol{\nu},\boldsymbol{\eta}}(x)=\boldsymbol{\nu}\cdot x+\boldsymbol{\eta}$, with $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ uniformly sampled in $\mathbb{Z}_q^N \setminus \{0\}$ and $\mathbb{Z}_q^N$ resp., and commits to the secret parameters used for blinding (*i.e.,* $\boldsymbol{\nu}, \boldsymbol{\eta}$) using a hiding and binding crypto-

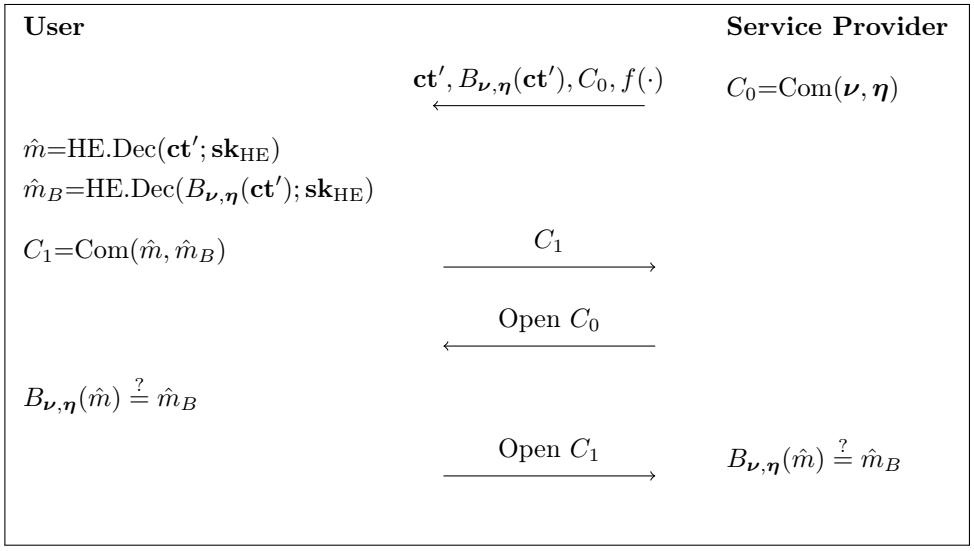


Figure 5.3: Release protocol for a computed value $\hat{m}$. For simplicity, we only focus on the first scalar entry resulting from the homomorphic decoding procedure.

graphic commitment Com$(\cdot)$ as C_0=Com$(\boldsymbol{\nu}, \boldsymbol{\eta})$. A hash-based commitment scheme can be used for this purpose [CDG+17]. Subsequently, the service provider sends to the user the encrypted result $\mathbf{ct}'$, its blinding $B_{\boldsymbol{\nu},\boldsymbol{\eta}}(\mathbf{ct}')$, and the commitment C_0. Upon reception, the user checks if the function $f(\cdot)$ is admissible. If the user accepts the computation $f(\cdot)$, she decrypts both ciphertexts as: HE.Dec$(\mathbf{ct}'; \mathbf{sk}_{\text{HE}})$=$\hat{m}$ and HE.Dec$(B_{\boldsymbol{\nu},\boldsymbol{\eta}}(\mathbf{ct}'); \mathbf{sk}_{\text{HE}})$=$\hat{m}_B$. Then, she commits to the decrypted results, *i.e.,* C_1=Com$(\hat{m}, \hat{m}_B)$, and communicates C_1 to the service provider who opens the commitment C_0 to the user (*i.e.,* revealing $\boldsymbol{\nu}, \boldsymbol{\eta}$). The user verifies that the initial blinding was correct by checking if $B_{\boldsymbol{\nu},\boldsymbol{\eta}}(\hat{m}) \stackrel{?}{=} \hat{m}_B$. If this is the case, she opens the commitment C_1 (*i.e.,* revealing $\hat{m}, \hat{m}_B$) to the service provider who verifies that the cleartext result matches the blinded information (*i.e.,* by also checking if $B_{\boldsymbol{\nu},\boldsymbol{\eta}}(\hat{m}) \stackrel{?}{=} \hat{m}_B$). At the end of the release phase, both parties are confident that the decrypted output is the expected result of the computation, while the service provider learns only the computation's result and nothing else about the user's data.

5.5 Privacy and Security Analysis

CRISP protects the user's privacy by revealing only the output of the agreed computation on her data, and it protects the service provider's integrity by preventing any cheating or forgery from the user. Here, we present these two properties and their corresponding proofs.

5.5.1 Privacy

Proposition 5.5.1. Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $H(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N, q, \chi_{\text{enc}}, \chi_{\text{key}}, \chi_{\text{err}})$ and BDLOP $(\delta_1, k, \kappa_{\text{lwe}}, \kappa_{\text{sis}}, q, N)$ schemes have been configured to ensure post-quantum security, that the circuit $\mathcal{C}$ is a valid (2,3)-decomposition, and that the cryptographic commitment $\text{Com}(\cdot)$ is hiding and binding. Then, our solution achieves privacy by yielding nothing more than the result $\hat{m}$ of the computation on the user's data $\{\mathbf{x}_i\}$.

We provide the full proof in Appendix C.2

5.5.2 Integrity

Proposition 5.5.2. Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $\text{H}(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N, q, \chi_{\text{enc}}, \chi_{\text{key}}, \chi_{\text{err}})$ and BDLOP $(\delta_1, k, \kappa_{\text{lwe}}, \kappa_{\text{sis}}, q, N)$ schemes have been configured to ensure post-quantum security, that the ZKB^{++} protocol execution of $\mathcal{C}$ achieves soundness κ_{ZK}, that the blinding function $B_{\nu,\eta}$ is hiding, and that the cryptographic commitment $\text{Com}(\cdot)$ is hiding and binding. Then, our solution achieves integrity as defined in Section 5.2.3, as it ensures with soundness κ_{ZK} that the output $\hat{m}$ is the result of the computation on the user's data.

We provide the full proof in Appendix C.3

5.6 Evaluation

We evaluate CRISP on three use cases: smart metering, disease susceptibility, and location-based activity tracking, using public real-world datasets. We detail first the instantiation and parameterization of our proposed solution, then illustrate its overall performance per use case, in terms of both overhead and utility. As previously mentioned, CRISP enables to offload the data and to conduct multiple operations on it. For simplicity, we present only one operation per dataset.

5.6.1 Implementation Details

We detail how the various blocks of our construction are implemented and configured.

Implementation. We implement the various blocks of CRISP on top of different libraries. The homomorphic computations are implemented using the Lattigo library [EPF21]. The commitment and encryption blocks of the circuit are implemented using CKKS from [SNU19] by employing a ZKB^{++} approach. The circuit's Boolean part (*i.e.,* the hash and conversion blocks) is implemented on top of the SHA-256 MPC-in-the-head circuit of [GMO16]. All the experiments are executed on a modest Manjaro 4.19 virtual machine with an i5-8279U processor running at 2,4 GHz with 8GB RAM.

CKKS & BDLOP. For CKKS, we use a Gaussian noise distribution of standard deviation 3.2, ternary keys with i.i.d. coefficients in $\{0, \pm 1\}^N$, and we choose q and N depending on the computation and precision required for each use case, such that the achieved bit security is always at least 128 bits. Each ciphertext encrypts a vector $\mathbf{d}$ consisting of the data points $\{x_i\}$ in the series of messages $\{msg_i\}$. Our three use cases need only computations over real numbers, hence we extend the real vector to a complex vector with a null imaginary part. The parameterization of CKKS for our three use cases leads to a computation accuracy higher than 98%. Similarly to CKKS, the BDLOP parameters for the commitment to the encryption noises are use-case dependent. In principle, we choose the smallest parameters to ensure a 128-bit security ($\lambda_{\text{lwe}} = \kappa_{\text{sis}} = 1$) and δ_1 and k are chosen according to N and q.

ZKB^{++}. We set the security parameter κ_{ZK} to 128, which corresponds to 219 iterations of the ZKB^{++} protocol. We also consider seeds of size 128 bits and a commitment size of $|c|$=256 bits using SHA-256 as in [CDG$^+$17]. Overall, considering the full circuit, the proof size per ZKB^{++} protocol iteration $|p_i|$ is calculated as

$$|p_i| = |c| + 2\kappa_{\text{ZK}} + \log_2 3 + \frac{2}{3}(|\mathbf{d}| + |\text{Com}| + |\text{Enc}| + |\mathbf{t}|) + b_{\text{hash}} + b_{\text{A2B}},$$

with $|\mathbf{d}|$ being the bit size of the secret inputs, |Com| the bit size of the commitment parameters, |Enc| the bit size of the encryption parameters, b_{hash} the number of multiplicative gates in the SHA-256 circuit, b_{A2B} the number of AND gates in the conversion block, and $|\mathbf{t}|$ the bit size of the additional information required to reconstruct the data source's message but not needed for the service provider's computation (*e.g.,* user identifier, nonce, timestamps, etc.). We note that according to the NIST specification [Dan15], SHA-256 operates by hashing data blocks of 447 bits. If the size of the user's input data exceeds this, it is split into chunks on which the SHA-256 digest is evaluated iteratively, taking as initial state the output of the previous chunk (see [Dan15]). We adapt the SHA-256 Boolean circuit described in [GMO16], which uses 22,272 multiplication gates per hash block, to the setting of ZKB^{++} [CDG$^+$17]. The Boolean part of the circuit is focused on the $|\mathbf{x}|$ least significant bits of the arithmetic sharing of $\mathbf{d}$ which is concatenated locally with a Boolean secret sharing of the additional information (nonce, uid, etc.). In our implementation, the user needs 182ms to run the Boolean part of the circuit associated with generating a hash from a 32-bits shared input $\mathbf{x}$. The verifier needs 73ms to verify

this part of the circuit.

Release Protocol. We use SHA-256 as a commitment scheme $\mathrm{Com}(\cdot)$ and a linear blinding operation $B_{\nu,\eta}(\cdot)$ in $\mathbb{Z}_q$.

Evaluation Metrics. We evaluate the performance of our solution on different use cases with varying complexity in terms of computation (*i.e.,* execution time) and communication (*i.e.,* proof size) overhead. The proof Π is detailed as the proof for the ZKCE, as well as the BDLOP bound proof. We also report the optimal ZKCE proof size per datapoint: *i.e.,* if the ciphertexts are fully packed. To cover a wide range of applications we evaluate various types of operations on the protected data such as additions, weighted sums, as well as a polynomial approximation of the non-linear Euclidean distance computation.

5.6.2 Smart Metering

We consider a smart meter that monitors the household's electricity consumption and signs data points containing a fresh nonce, the household identifier, the timestamp, and its consumption. A user's privacy can be jeopardized as energy consumption patterns can reveal her habits [CPW10, KLB+19]. The energy authority is interested in estimating the total household consumption (*i.e.,* the sum over the consumption data points) over a specified time period I (*e.g.,* a month or a year) for billing purposes and to provide reliable services [ADMC17]. The computation is

$$m_{\mathrm{sm}} = \sum_{i \in I} \mathbf{d}[i],$$

where $\mathbf{d}$ is the vector of the household consumption per half hour. As our solution offloads the encrypted data to the service provider, additional computations, *e.g.,* statistics about the household's consumption, are possible without requiring a new proof; this improves flexibility for the service provider.

Dataset & Experiment Setup. We use the publicly available and pre-processed UKPN dataset [UKP19] that contains the per half-hour (phh) consumption of thousands of households in London between November 2011 and February 2014. Each entry in the dataset comprises a household identifier, a timestamp, and its consumption phh. For our experiment, we randomly sample a subset of 1,035 households and estimate their total energy consumption over the time span of the dataset with our solution. We set the parameters as follows: We use a modulus of $\log q$=45 bits and a precision of 25 bits, which imposes a maximum of 2^{10} slots for the input vectors ($\log N$=11). Hence, each household's consumption phh is encoded with multiple vectors $\mathbf{d}_k$ to cover the time span of the dataset. To evaluate its proof size, we assume that the messages obtained from the

Table 5.1: Evaluation summary of CRISP (reported timings are the averages over 50 runs ± their standard deviation).

Use Case	Computation	Mean Absolute Relative Error	t_{enc} (ms)	t_{comp} (ms)	t_{dec} (ms)	Proof Size (MB)	t_{prove} (s)	t_{ver} (s)
Smart Metering	Sum	$5.1 \cdot 10^{-5}$	70 ± 10	130 ± 30	0.7 ± 0.3	650.5	200 ± 10	82 ± 5
Disease Susceptibility	Weighted Sum	$2.2 \cdot 10^{-5}$	60 ± 10	22 ± 5	2.7 ± 0.8	53.9	26 ± 4	13 ± 2
Location-Based Activity Tracking	Euclidean Distance	$1.5 \cdot 10^{-2}$	980 ± 70	180 ± 30	7 ± 2	1,603	470 ± 40	210 ± 10

smart meter include a 16-bit household id, a 128-bit nonce, a 32-bit timestamp, and a 16-bit consumption entry.

Results. The average time for encryption of a vector of 1,024 datapoints at the user side is t_{enc}=70ms, and the decryption requires t_{dec}=0.7ms. The mean time for the energy computation at the service provider side is t_{comp}=130 ms. To generate the proof for one ciphertext, containing 1,024phh measurements (*i.e.,* 21 days worth of data), the user requires t_{prove}=3.3min, and its verification at the service provider's side is executed in t_{ver}=1.4min. The estimated ZKCE proof size for each ciphertext of 1,024 elements is 643.4MB, whereas the bound proof is 9.9MB.

For fully packed ciphertexts (1,024 datapoints), CRISP's proof generation and verification respectively take 195ms and 80ms per datapoint, with a communication of 628KB.

5.6.3 Disease Susceptibility

To improve a user's health and to customize her treatments, medical centers and direct-to-consumer services [23a19, DNA19] can provide a user with her DNA sequence. Genomic data can be used for disease-susceptibility tests offered by service providers, *e.g.,* research institutions that seek to form the appropriate cohorts for their studies. The user wants to protect her data, because DNA is considered a very sensitive and immutable piece of information for her and her relatives [EN14]. Correspondingly, service providers are keen on collecting users' data and verifying its integrity so that they can use it for disease-risk estimation or other types of analyses, *e.g.,* drug-effect prediction or health certificates. To disrupt this process and/or pass a medical examination, malicious users could tamper with the genomic data they share. We assume a medical center that sequences a patient's genome and certifies batches of single nucleotide polymorphisms (SNPs) that are associated with a particular disease ∂. A privacy-conscious direct-to-consumer service is interested in estimating the user's susceptibility to that disease by

calculating the following normalized weighted sum

$$m_\partial = \sum_{i \in S_\partial} \omega_i \cdot \mathbf{d}[i],$$

where S_∂ is the set of SNPs associated with ∂ and ω_i are their corresponding weights. The vector $\mathbf{d}$ comprises values in $\{0, 1, 2\}$, thus indicating the presence of an SNP in 0, 1, or both chromosomes that can be represented by two bits. This use case illustrates the need for flexibility in the service provider's computations, as it may be required to evaluate, at different times, several diseases on the same input data. Moreover, it accentuates the need for resistance against quantum adversaries, as genomic data is both immutable and highly sensitive over generations.

Dataset & Experiment Setup. We employ the 1,000 Genomes Project's public dataset [Int19b] that contains the genomic sequences of a few thousand individuals from various populations. We randomly sample 145 individuals and extract 869 SNPs related to five diseases: Alzheimer's, bipolar disorder, breast cancer, type-2 diabetes, and schizophrenia. We obtain the weight of an SNP, with respect to those diseases from the GWAS Catalog [BMC+19]. Then, for every individual, we estimate their susceptibility to each disease. For this use case, we use a precision $\log p$=25, a modulus of $\log q$=56 consumed over two levels, and a polynomial degree of $\log N$=12. The input vector $\mathbf{d}$ (consisting of 2^{11} slots) is an ordered vector of integers containing the SNP values, which are coded on two bits, associated with the diseases. One vector is sufficient for the considered diseases. To estimate the proof size, we assume that the message signed by the data source contains a 16-bit user identifier, a 128-bit nonce, and the entire block of SNPs.

Results. The average encryption time for up to 2,048 SNPs at the user side is t_{enc}=60ms, and the decryption is t_{dec}=2.7ms. The computation time of the disease susceptibility at the service provider is t_{comp}=22ms. The user needs t_{prove}=26 s to generate the proof for the arithmetic part of the circuit, and the service provider verifies it in t_{ver}=13 s. The estimated proof size for the ZKCE is 36.6MB, whereas the bound proof is 22MB.

5.6.4 Location-Based Activity Tracking

A user's wearable device monitors her location by querying location providers. To obtain activity certificates or discount coupons, the user then shares this information with service providers, *e.g.,* online fitness social networks [Int19a] or insurance companies [San19]. As location data can reveal sensitive information, *e.g.,* her home/workplaces or habits [VDSKK18, Her18], the user is concerned about her privacy. Service providers want legitimate data to issue activity certificates, provide discounts for performance achievements, and build realistic user profiles. Malicious users might be tempted to modify their data in order to claim fake accomplishments and/or to obtain benefits they

are not entitled to. We assume that a user runs with a wearable device that retrieves her location points during the activity from a data source, *e.g.,* a cellular network. The service provider, *e.g.,* an online fitness social network [Int19a], seeks to estimate the total distance that the user ran during her activity I:

$$m_{\text{run}}=\sum_{i\in I}\sqrt{(\mathbf{d}[i+1]-\mathbf{d}[i])^2+(\mathbf{d}[\frac{N}{4}+i+1]-\mathbf{d}[\frac{N}{4}+i])^2},$$

with $\mathbf{d}$ the vector of UTM (Universal Transverse Mercator) inputs packing eastings in the first half of the vector and northings in the second. Given that Euclidean distance computations require the evaluation of a non-linear square root function, we consider its least-squares approximation by a degree seven polynomial on a Legendre polynomial base.

Dataset & Experiment Setup. We run our experiment on a public dataset from Garmin Connect [Int19a]. This dataset contains GPS traces of thousands of users engaging in various activities such as walking, running, and cycling. We randomly sample 2,000 running traces, and we discard traces with less than 15 points and more than 2,000 points. Our initial dataset analysis shows that the traces are very *noisy*: we identified unrealistic distances between consecutive points, timestamps, and locations. We use GPSBabel [Lip19], an open-source software, to interpolate the running traces such that the following criteria are met: (a) the maximum speed of a runner is less than 10m/s, (b) the maximum distance between consecutive points is less than 30m, and (c) the time delta between two points is less than 3s. These criteria are realistic for running activities. We remove traces of the time samplings that were improperly executed by the data source (difference more than 10s, standard deviation more than 5, or a zero inter-quartile at 75%), as well as traces with unacceptable idleness.[2] We then convert the remaining GPS traces to UTM to obtain the northings and eastings geographic coordinates. Overall, we obtain a dataset of 1,608 traces (80% of the initial 2K running trace dataset) that, on average, contain 1,124 datapoints, and we estimate their total distance with CRISP.

Considering the polynomial approximation required for the square-root function, we set the size of the polynomial ring $N=2^{13}$ and a modulus $\log q=184$. To calculate the proof sizes, we assume that the messages obtained from the data source contain a 16-bit user identifier, a 128-bit nonce, a 32-bit timestamp, and 24-bit easting/northing coordinates.

Results. The encryption and decryption overhead for fully packed ciphertexts of up to 2,048 points at the user side are t_{enc}=980ms and t_{dec}=7ms, respectively; and the Euclidean distance computation at the service provider requires t_{comp}=180ms. For 2,048 data points, the user generates the proof for the arithmetic part of the circuit in t_{prove}=7.9min, and the service provider verifies it in t_{ver}=3.4min. Considering that each message signed by

[2]Idleness of a trace is a situation where the interquartile at 25% of the instant speed is less than 0.3m/s and where the covered distance is less than 15m.

the data source is 96-bits, the proof size per trace for the ZKCE is $1,499.2$MB, and the bound proof is 94.9MB.

For our data set, the average proof size is 922.1MB, considering the mean number of points in the traces. In Section 5.6.5, we will show how to reduce this proof size. With fully packed ciphertexts, CRISP's proof generation requires 230ms per datapoint and 100 ms for its verification, at a communication cost of 732KB.

5.6.5 Reducing the Communication Overhead

Table 5.1 summarizes CRISP's overhead for three use cases: smart metering, disease susceptibility, and location-based activity tracking. We observe that it introduces acceptable computational overhead at the user and service provider sides, and that it achieves average absolute relative error between $2.2{\cdot}10^{-5}$ and 0.015 for the desired computations. We remark however that our construction uses post-quantum secure lattice-based cryptographic primitives, such as encryption and commitment and, to ensure the integrity of the user's data transfer, the MPC-in-the-head approach. This comes at the price of increased communication (*i.e.,* proof size). Therefore, to reduce this overhead, we propose several improvements to be employed, and we illustrate them in Figure 5.4 for the smart metering and location-based activity-tracking use cases.

Random Integrity Checks (RIC). A first optimization is to reduce the number of data points whose integrity is checked by the service provider. This introduces a trade-off between CRISP's security level and its communication overhead. In particular, a service provider can decide to check only a subset of the input data hashes in the data verification phase, as we assume *malicious but rational* users (Section 5.2.2) who will not cheat if there is a significant probability of getting caught. Such a strategy enables a service provider to tune the solution, depending on the level of confidence it has in the user. In Figure 5.4, we observe how the proof size decreases as the service provider checks fewer data blocks. For instance, if the service provider checks 20% of the data blocks in the verification phase (RIC-20%), the proof size for location-based activity tracking drops from $1,499.2$MB to 497MB (*i.e.,* 243KB/datapoint), whereas for smart metering it decreases from 643.4MB to 142.2MB (*i.e.,* 139KB/datapoint). This yields a reduction of more than 66% in the total ZKCE communication overhead. Computation times to generate and verify the proofs are also more than halved.

Batching (BG). Another improvement is to modify the way data sources certify the users' data points. So far, in the smart metering and location-based activity-tracking use cases, we have assumed that data sources hash and sign every data point generated by the user. However, another strategy is to hash batches of data points in a single signed message. This modification is purely operational as it does not require additional software or hardware deployment. We set the batch size depending on the use case –

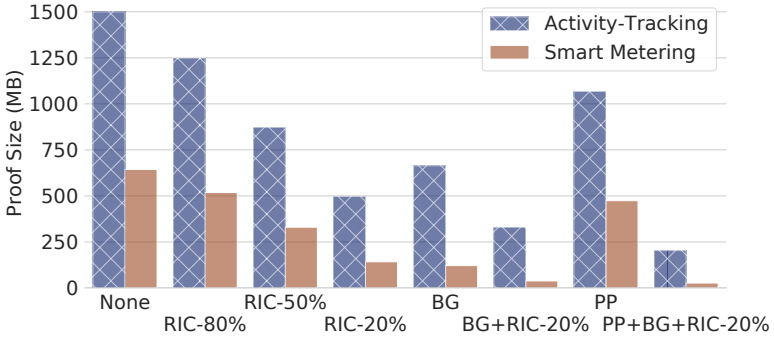


Figure 5.4: ZKCE proof size (MB) for fully packed ciphertexts and various optimizations.

i.e., considering the additional information of each message before signature – such that the overall batch can fit on a single SHA-256 input block of 447bits. Figure 5.4 shows a reduction of more than 50% in proof size for the two use cases when batching (BG) is employed compared to the non-optimized solution. Batching can also be combined with RIC-20% (BG+RIC-20% in Figure 5.4): For smart metering, the ZKCE proof size is further reduced to 38.1MB (*i.e.,* 37.2KB/datapoint), whereas for the location-based activity tracking the proof size drops to 329.7MB (*i.e.,* 161KB/datapoint). For activity tracking, the t_{prove} is reduced to 2.1min and t_{ver} to 1.1min (61 and 32ms per datapoint, resp.). For smart metering, t_{prove} is reduced to 20s and t_{ver} to 9.3s (20 and 9ms per data point, resp.).

ZKCE Pre-processing (PP). Finally, we can employ a ZKCE pre-processing model, such as that presented by Katz *et al.* [KKW18]. The pre-processing model considers that the user executes *offline* a series of circuit evaluations on committed values. The service provider challenges a subset $\mathcal{M}$ of those evaluations and checks their integrity, and the remaining τ ones are used in an *online* phase, along with the committed values. The rest of the protocol is similar to ZKB^{++}. The proof size per iteration is reduced to:

$$|p_i| = 2\kappa_{\text{ZK}} + \tau \log_2 \frac{\mathcal{M}}{\tau} 3\kappa_{\text{ZK}} + \tau(\kappa_{\text{ZK}} \log_2 3 + 2\kappa_{\text{ZK}} + (|\mathbf{d}| + |\text{Com}| + |\text{Enc}| + |\mathbf{t}|) + 2(b_{\text{hash}} + b_{\text{A2B}})).$$

Regarding our three players setting, a 128-bit security level requires $\mathcal{M}$=300 and τ=81, thus yielding a significant reduction of 25% on the proof size (see [KKW18] for the computation details) compared to the non-optimised approach. Pre-processing, batching, and RIC can also be applied together to obtain smaller proofs (see PP+BG+RIC-20% in Figure 5.4): For smart metering, the ZKCE proof is reduced to 26.8MB. Similarly, for location-based activity tracking, the ZKCE proof becomes 203.0MB. This yields an optimal ZKCE proof size per datapoint of 26.2KB, and 99.1KB for smart metering and activity-tracking, respectively. Finally, we remark that according to Katz *et al.* [KKW18], a trade-off between proof size and prover's computations could be achieved by increasing the number of players involved in the MPC-in-the-head protocol. However, such an improvement would require additional changes in CRISP, *e.g.,* the conversion block that

interfaces the arithmetic and Boolean parts of the circuit should be adapted for a larger number of players.

5.6.6 Comparison with ADSNARK

ADSNARK [BBFR15] is a generic construction that could be employed to address the trade-off between privacy, integrity, and utility. In particular, it enables users to locally compute on data certified by data sources and to provide proof of correct computation to service providers. However, ADSNARK does not support the feature of data offloading that enables service providers to reuse the collected data and to perform various computations. Indeed, ADSNARK and other zero-knowledge solutions [FKDL13, FL14, BSBHR19], require the user to compute a new proof every time the service provider needs the result of a new computation. Furthermore, it requires a trusted setup and relies on different security assumptions that are not secure in the presence of quantum adversaries [KKW18]. The latter should be taken into account, considering recent advances in quantum computing [AAB+19] and the long-term sensitivity of some data.

A fair comparison with ADSNARK is not trivial to achieve, as our solution provides post-quantum security and overcomes the constraint of a trusted setup. Nonetheless, here we provide hints of their qualitative and quantitative differences. In particular, ADSNARK considers a smart-metering use case that requires a non-linear cumulative function for the billing analysis of a month of data. We consider a similar non-linear pricing function evaluated by a degree-two polynomial, and we evaluate CRISP on the UKPN dataset for 400 households, with $N{=}2^{12}$ and $\log q{=}106$. In terms of proof size, our construction yields 889.2MB (verifying all the measurements for a month), whereas the overhead induced by ADSNARK is 53MB. However, we remark that the latter requires that, every time a different computation is needed, a new proof is generated and exchanged. In our solution, this cost is incurred only once; any subsequent operations can be computed locally by the service provider on the verified data. Furthermore, when using actual signatures (and not MACs), ADSNARK accounts for only a "*theoretical estimate*" of the complexity of the signature circuit (with only one thousand multiplicative gates for signature verification) and, if we were to evaluate our solution with this circuit, the proof size would be only 104.2MB. Therefore, our analysis shows that our construction offers comparable results to the state of the art and provides stronger security guarantees.

5.7 Discussion

In this section, we present some interesting considerations that could influence the deployment of our solution.

5.7.1 Signature Scheme

As discussed in Section 5.4.1, CRISP is agnostic of the digital signature and is compatible with any scheme that uses the SHA-256 hash function. We employ SHA-256, as it is widely deployed in current infrastructures, adopted by various signature schemes (*e.g.,* the recent post-quantum SPHINCS [BHK+19] or the standard ECDSA schemes), and as it is a benchmark for the evaluation of ZKBoo [GMO16] and ZKB^{++} [CDG+17]. This flexibility enables CRISP to be compliant with currently deployed signature schemes that might not be quantum resistant (*e.g.,* ECDSA) at the cost of CRISP's post-quantum integrity property. Working with other hash functions (*e.g.,* SHA-3 that is employed in [CDG+17]) is possible, with modifications to CRISP's circuit.

5.7.2 Integrity Attacks

CRISP copes with malicious users that might attempt to modify their data or the computed result to their benefit. However, some use cases require accounting for additional threats. For example, for smart metering, users might purposefully *fail* to report some data (*i.e.,* misreport) to reduce their billing costs. Similarly, in location-based activity-tracking, users might re-use pieces of data certified by the data source to claim higher performances and to increase their benefits (*i.e.,* double report). Such attacks can be thwarted by system-level decisions; *e.g.,* data sources can generate data points at fixed time intervals known to service providers. Message timestamps can be encrypted, along with the data points, so that service providers can verify their properties (*e.g.,* their order or their range). As these attacks are application-specific, we consider them to be out of the scope of this work.

5.7.3 Usability

Even though CRISP introduces non-negligible communication and computation overhead, it remains acceptable for modern systems. The independent iterations of the ZKCE make the proof generation highly parallelizable and require much less memory than the full proof size (experimentally, as little as 2GB of RAM). CRISP has also the advantage of being an offline system that requires interaction in only the release protocol: *e.g.,* the transfer phase can be executed when the user is idle. Additionally, recent communication systems such as fibre-optic internet or 5G offer high throughput links: With an 80Mb/s link, the proof for three weeks' worth of smart-metering data would require only about a minute to be transferred. For activity tracking, CRISP can be executed when the user plugs her wearable device into a computer and transfers the data while recharging it.

5.8 Related Work

Several works are devoted to protecting privacy for smart metering (*e.g.,* see surveys [ETPLPG13, WL13]). However, only some of them, *e.g.,* [AS16, LL12], by relying on custom homomorphic signature schemes, also address the concern of data integrity and authenticity. The applicability of such solutions is limited as, according to their technical specifications [Uni18], smart meters cope with standard digital signatures, *e.g.,* ECDSA [NIS15]. Similarly, a number of works, *e.g.,* [ARHR13, DCFT13, WZD+16], employ homomorphic encryption to protect genomic privacy and to perform disease-susceptibility computations. Their model considers a medical unit that sequences the DNA of the user who, in turn, protects it via homomorphic encryption before sending it for processing to a third party. These solutions do not address the issue of data integrity or authenticity. Finally, several works are dedicated to both privacy and integrity in location-based activity tracking [ZC11, WPZM16, LH10, PHB+15, SW09, HB11]. They also are either peer-based [ZC11, WPZM16], infrastructure-based [LH10, PHB+15], or hybrid[SW09, HB11]. SecureRun [PHB+15] offers activity proofs for estimating the distance covered in a privacy and integrity-preserving manner. Nevertheless, the system's accuracy relies on the density of access points, thus it achieves at best a median accuracy of 78% (compared to 99.9% with CRISP on a similar dataset).

5.9 Summary

Data sharing among users and service providers in the digital era incurs a trade-off between privacy, integrity, and utility. We propose a generic solution that protects the interests of both users and service providers. Building on state-of-the-art lattice-based homomorphic encryption and commitments, as well as zero-knowledge proofs, our construction enables users to offload their data to service providers in a post-quantum secure, privacy and integrity-preserving manner, yet still enables flexible computations on it. We have evaluated our solution on three different use cases, thus showing its wide potential for adoption.

Chapter 6

Conclusion

Even though homomorphic encryption (HE) has become more and more practical over the last decade, HE-based systems have been mostly envisioned in the honest-but-curious threat model only. However, this threat model assumption grows to be unrealistic against real-world adversaries that can render both the security and utility of HE pipelines moot. Because numerous applications of HE are security sensitive (*e.g.,* medical research, machine learning, etc.), this limitation is an impediment to actual deployments. Protecting these systems against malicious attacks requires solutions along the whole HE pipeline: the setup/key generation, encryption, and decryption phases executed by one or more clients and the computation phase executed by a computing server.

In this dissertation, we have proposed novel constructions for ensuring the correctness of different phases of such pipelines against malicious adversaries. By carefully combining proof systems and integrity protection mechanisms with the complex structure and operations of lattice-based HE, we provided solutions that convert an HE pipeline in the honest-but-curious threat model to a malicious one. Our constructions work *off-the-shelf* and can be applied as an add-on to the existing HE pipeline.

In Chapter 3, we have explored how to practically verify the computing server's evaluation. To achieve this, we have proposed new plaintext encoders that offer error-detection capabilities to clients when verifying the homomorphic evaluation. We have implemented our solution into the library VERITAS and showed its practicality over a wide range of use cases, thus demonstrating its applicability in practice. We show that VERITAS can be used to port standard HE pipelines to a stronger threat model with a malicious computing server almost seamlessly. Our experimental evaluation also demonstrated that VERITAS costs are acceptable for HE pipelines considering the benefits of being able to detect malicious behavior.

In Chapter 4, we have proposed to use lattice-based proof systems for verifying the correct

execution of client operations in HE pipelines. Using our system PELTA, clients can efficiently prove the correctness of their cryptographic keys and any of their operations involved in the HE pipeline. As such, PELTA is a critical building block to ensure the security of HE pipelines against a malicious adversary, particularly in multiparty scenarios. Compared to generic proof systems, PELTA is computationally more than one order of magnitude more efficient.

Finally, in Chapter 5, we have proposed a new method for verifying the correct encryption of authenticated data. Our solution enables a verifier to assess the correctness of offloaded encrypted data in one shot. Our technique carefully combines multi-party computation in-the-head to simultaneously verify the encryption and the authentication of the data. Our implementation shows the performance of CRISP and we propose several trade-offs to reduce the communication overhead.

Overall, in this dissertation, we have shown that the malicious tolerance for HE pipelines is in the realm of practicality. In addition, we believe that our contributions are significant first steps towards efficient malicious-tolerant HE pipelines. We trust our open-source implementations and evaluations will serve as a baseline for future works that will investigate further malicious-resistant HE pipelines.

6.1 Open Problems and Future Research Directions

Throughout our work, we have identified potential improvements of the works presented in this thesis and interesting future research directions.

6.1.1 Open Problems for Verifiable Homomorphic Computation

Public verifiability. Our new plaintext encodings, which enable error detection, are limited to private verifiability. Indeed, VERITAS builds on homomorphic MACs with symmetric keys. As such, it limits the potential application to multiparty scenarios. In a recent work, Fernàndez-València extended VERITAS to the multiparty scenario and proposed that the parties share a common encoding key [FV23]. This approach is thus limited to the setting with honest clients. A potentially interesting avenue would be to explore the advances in homomorphic signatures [GVW15] and functional commitments [WW23] and assess if they could mitigate this limitation.

Protecting ciphertexts. Because VERITAS embeds verification capabilities in the plaintext space, the result of the decryption needs to be kept private from the computing server. Performing the verification directly on the ciphertexts alleviates this limitation. An attractive option would be to build on very recent work on the front of verifiable

HE [VKH23] and to use lattice-based proofs [ESLL19b, BLNS20] to achieve this without affecting the HE pipeline operations.

Differential privacy. HE does not provide any privacy guarantees after decryption. Thus, combining it with differentially private mechanisms [Dwo06] is an interesting avenue for research. Some works have started exploring how to combine HE with differential privacy using the encryption noise induced by RLWE construction as a base [LMSS22, Ogi23]. It would be interesting to explore how to ensure the verifiable use of differentially private mechanisms in such contexts.

6.1.2 Open Problems for Verifiable Client's Operations

PELTA's proof size. A limitation of the lattice-based approach for PELTA is the relatively large-sized proof, compared to the achievements when using SNARKs, such as Pinocchio [PHGR13]. Even if we show that PELTA 's proofs are acceptable compared to the size of the encryptions themselves, it is worth keeping in mind that HE already induces a significant communication overhead. As a result, new works on the front of lattice-based HE could help reduce the additional overhead of malicious protection.

Input verification. The input verification offered by CRISP considered the use of SHA256. Our experimental evaluation showed that this hash is a clear bottleneck in our construction, as it leads to large proof sizes. The development of new SNARK-friendly signatures (*i.e.,* with low multiplicative complexity) and their deployment could help reduce CRISP's proof size increasing its practicality. Furthermore, depending on the signature type, these signatures could be compatible with lattice-based proofs that would offer more attractive proof sizes. Additionally to CRISP, an interesting future work would be to explore techniques to verify properties of the input data directly using for instance statistical tests [ZPGS19].

Matching rapid advances in HE. Research on the front of HE is a rapidly evolving field. We have proposed several blocks to protect HE pipelines that use common HE schemes such as BFV, BGV, or CKKS. It would be interesting to explore how our techniques could be adapted to different constructions such as TFHE [CGGI20]. Additionally, several improvements have been proposed since our work. For instance, Belorgey *et al.* [BCG$^+$23] recently proposed to decouple the big integer decomposition from the cyclotomic arithmetic aspect. This could alleviate compatibility issues with some proof systems that we observed in Chapter 3. Additionally, incorporating protections against malicious adversaries early on in the development of HE systems and libraries would increase their usability.

6.2 Final Remarks

Throughout the different projects that comprise this dissertation, I encountered several challenges. The first notable observation is that state-of-the-art cryptographic research is often not translated into practical implementations. However, it is crucial to recognize that implementation plays a pivotal role in promoting the adoption of new cryptographic schemes. In the case of the constructions examined in this dissertation, this limitation hindered straightforward comparisons with prior work. As a valuable lesson, my work has underscored the significance of publishing code that is well-documented and easily accessible, facilitating reproducibility, comparisons, and adoption.

Another challenge I encountered in my research was navigating the intersection of two dynamic and thriving fields: homomorphic encryption and proof systems. Both areas are currently evolving rapidly, making it a complex task to identify optimal points of integration. In particular, the progress made on one front could clash with the requirements of the other leading to incompatibilities. Additionally, as our system models and problem statements are at the crossroads between two fields, the related work to practically compare with was limited. Nevertheless, I am delighted by the increasing number of works addressing this subject in recent years. I am very optimistic that this important question will continue to receive attention as both homomorphic encryption and proof systems advance.

Appendix A

Appendix for VERITAS (§3)

A.1 Homomorphic Authenticators

In this section, we recall the formalism of homomorphic authenticators from Gennaro and Wichs [GW13].

HA.KeyGen$(1^\lambda) \rightarrow (\text{evk}, \text{sk})$: Output a secret key sk and an evaluation key evk for the authenticator.

HA.Auth$(m, \tau; \text{sk}) \rightarrow \sigma$: Create an error-detecting authentication σ for a message m associated with an identifier τ.

HA.Eval$(f(\cdot), \vec{\boldsymbol{\sigma}}; \text{evk}) \rightarrow \sigma'$: Evaluate a deterministic function $f(\cdot)$ on authenticated data where $\vec{\boldsymbol{\sigma}}=(\sigma_1,...,\sigma_n)$ and each σ_i authenticates the i-th input m_i. σ' authenticates the result $m'=f(m_1,...,m_n)$.

HA.Ver$(\mathcal{P}, \sigma'; \text{sk}) \rightarrow \{0,1\}$: Check that m' obtained from the authentication σ' is the correct output of the program $\mathcal{P}=(f(\cdot), (\tau_1,...,\tau_n))$; *i.e.,* the evaluation of $f(\cdot)$ on authenticated inputs identified by $\tau_1,...,\tau_n$.

We now recall some of the properties of an HA scheme.

- **Authentication Correctness:** For any message m associated with τ,

$$\Pr\left[\text{HA.Ver}(\mathcal{I}_\tau, \boldsymbol{\sigma}; \text{sk})=1 \;\middle|\; \begin{array}{l}(\text{evk}, \text{sk}) \leftarrow \text{HA.KeyGen}(1^\lambda) \\ \boldsymbol{\sigma} \leftarrow \text{HA.Auth}(m, \tau; \text{sk})\end{array}\right]=1,$$

  with $\mathcal{I}_\tau=(Id, \tau)$ the program associated with the identity function.

Experiment 1: $\mathbf{Exp}_{\mathcal{A}}[\mathrm{HA}, \lambda]$

1. The challenger generates the keys $(\mathrm{sk}, \mathrm{evk}) \leftarrow \mathbf{HA.KeyGen}(1^{\lambda})$ and initializes the list of authenticated inputs: $T{=}\emptyset$.
2. $\mathcal{A}$ asks the challenger for authentication of m with identifier τ. If the identifier was already queried (*i.e.,* $(\tau, \cdot){\in}T$), then the challenger aborts. Otherwise, it returns $\sigma \leftarrow \mathbf{HA.Auth}(m, \tau; \mathrm{sk})$ and appends (m, τ) to T.
3. **Forgery**: The adversary $\mathcal{A}$ outputs a forgery $(\mathbf{m}^*, \mathcal{P}^* = (f, (\tau_1^*,...,\tau_n^*), \boldsymbol{\sigma}^*)$. The experiment outputs 1 if the verification accepts and that either:
   - (i) $\exists i{\in}[n], (\boldsymbol{\tau}_i^*, \cdot){\notin}T$ (*i.e.,* not authenticated input).
   - (ii) $\forall i \in [n], (\boldsymbol{\tau}_i^*, \cdot) \in T$ and $f(\mathbf{m}_1^*,..., \mathbf{m}_n^*) \neq \mathbf{m}^*$ (*i.e.,* wrongful computation).

- **Evaluation Correctness:** Consider any key pair $(\mathrm{evk}, \mathrm{sk})$ generated through the $\mathrm{HA.KeyGen}(1^{\lambda})$ procedure. Define any fixed circuit $f(\cdot)$, and any correctly generated triplet $\{(\mathcal{P}_i, m_i, \sigma_i)\}_{i=1}^n$. If $m^*{:=}f(m_1,..., m_n)$, $\mathcal{P}^*{:=}f(\mathcal{P}_1,..., \mathcal{P}_n)$, and $\sigma^* := \mathrm{HA.Eval}(f, (\sigma_1,..., \sigma_n); \mathrm{evk})$, then $\mathrm{HA.Ver}(\mathcal{P}^*, \sigma^*; \mathrm{sk}){=}1$.
- **Authenticator Security:** Given the security parameter λ, the probability of a malicious adversary convincing the verifier to accept a wrongfully computed result is negligible. More formally, define $\mathbf{Exp}_{\mathcal{A}}[\mathrm{HA}, \lambda]$ as in Experiment 1. The authenticator HA is said to be *secure* if for any PPT adversary $\mathcal{A}$, $\Pr[\mathbf{Exp}_{\mathcal{A}}[\mathrm{HA}, \lambda]{=}1] \leq \mathrm{negl}(\lambda)$.

A.2 Security Proof for REP (§3.4.2)

Theorem 3.4.1: Let λ be a power-of-two security parameter. If the pseudorandom function F_K and the canonical HE scheme are at least λ-bit secure, then for any program $\mathcal{P}$, REP as in Scheme 2 is a secure authenticator and a PPT adversary has a probability of successfully cheating the verification negligible in λ.

Proof. We follow Gennaro and Wichs' Theorem 3.1 on the security of their homomorphic MAC [GW13]. Let $\mathcal{A}(1^{\lambda})$ be a probabilistic polynomial time (PPT) attacker. We define the following game following Experiment 1:

Game0: This game is the forgery game based on Experiment 1 as $\mathbf{Exp}_{\mathcal{A}}[\mathrm{REP}, \lambda]$. We recall that the game outputs 1 if the verification procedure $\mathrm{REP.Ver}(\mathcal{P}^*, \boldsymbol{\sigma}^*; \mathbf{sk}) = 1$ and one of the two conditions holds:

- **Type 1:** $\mathcal{P}^*$ is not *well-defined* w.r.t. the set of inputs T (see §3.3.2).
- **Type 2:** $\mathcal{P}^*$ is *well-defined* on T and $f(\mathbf{m}_1^*,..., \mathbf{m}_n^*){\neq}\mathbf{m}^*$ (*i.e.,* wrongful computation).

The scheme REP is said secure if for all PPT $\mathcal{A}$,

$$\Pr\left[\text{Game0}(1^\lambda)=1\right] \leqslant \text{negl}(\lambda).$$

Now let us design hybrid games by modifying Game0.

Game1: We modify Game0 by replacing the PRF F_K with random values in the authentication and verification procedures such that a random oracle outputs random values on the fly. By the pseudorandomness of F_K, it is clear that

$$\Pr\left[\text{Game1}(1^\lambda) = 1\right] \geqslant \Pr\left[\text{Game0}(1^\lambda) = 1\right] - \text{negl}(\lambda).$$

Game2: We change the winning condition of Game1 such that the adversary wins only if the Type 2 attacks succeed and Type 1 fail. Let E be the event of winning a Type 1 forgery in Game1. We have $\Pr\left[\text{Game2}(1^\lambda) = 1\right] \geqslant \Pr\left[\text{Game1}(1^\lambda) = 1\right] - \Pr[E]$. Recall that E occurs when $\mathcal{A}$ outputs $(\mathbf{m}^*, \mathcal{P}^*{=}(f, (\boldsymbol{\tau}_1^*,...,\boldsymbol{\tau}_n^*)), \boldsymbol{\sigma}^*)$ and there exists $j{\in}[n]$ s.t. (i) the j-th input is used in the evaluation of the function f and (ii) $(\tau_j^*, \cdot){\notin}T$. Because the challenger directly rejects if $\exists(\tau_j^*, \cdot){\notin}T$, $\Pr[E] = 0$. This game is a mere syntactic change from Game1.

Game3: We modify the winning condition of Game2. Now, the challenger remembers the authentication $\boldsymbol{\sigma}$ associated to a message $\mathbf{m}$ with identifier $\boldsymbol{\tau}$ (*i.e.,* it stores $(\boldsymbol{\tau}, \mathbf{m}, \boldsymbol{\sigma}){\in}T$). Consider the adversary outputting a forgery $(\mathbf{m}^*, \mathcal{P}^*{=}(f(\cdot), (\boldsymbol{\tau}_1^*,..., \boldsymbol{\tau}_n^*)), \boldsymbol{\sigma}^*{=}\mathbf{c}^*)$. Denote by $\hat{\mathbf{c}}$ the honest ciphertext for the labeled program $\mathcal{P}^*$ (*i.e.,* the homomorphic evaluation of $f(\cdot)$ on inputs $\mathbf{c}_1,..., \mathbf{c}_n$). Denote by $\mathbf{M}_1^*$ the forgery of the first extended vector decrypted from $\mathbf{c}^*$[1]. We modify the verification procedure to:

1. Use the stored ciphertexts in T and decrypt them to obtain the challenge values and compute $f(\cdot)$ to get the challenge values of the output $\tilde{r}_i$. Check if $\forall i{\in}S$, $\mathbf{M}_1^*[i]{=}\tilde{r}_i$. Otherwise, reject.
2. $\forall i{\in}[\lambda] \setminus S$, check if all $\mathbf{M}_1^*[i]$ are equal to the honest output obtained from $\hat{\mathbf{c}}$ (say m). If so, reject.

By construction, any Type 2 forgery accepting in Game2 is also accepting in Game3. Thus, $\Pr\left[\text{Game3}(1^\lambda) = 1\right] \geqslant \Pr\left[\text{Game2}(1^\lambda) = 1\right]$ and

$$\Pr\left[\text{Game3}(1^\lambda) = 1\right] \geqslant \Pr\left[\text{Game0}(1^\lambda) = 1\right] - \text{negl}(\lambda).$$

Game4: We modify Game3 such that, when answering the authentication queries, the challenger encrypts the actual message regardless of the slot index (*i.e.,* regardless of the

[1]W.l.o.g. we consider only one scalar output. The general case is a trivial extension.

set S). By the semantic security of the homomorphic encryption scheme, the adversary cannot distinguish if a specific slot encrypts m or a challenge value r_i. Given a plaintext with slots that either store the original slot message or a challenge value, we can embed it into the authentication procedure for each position $i \in S$ and simulate either Game3 or Game4. Suppose $\Pr\left[\text{Game4}(1^\lambda)=1\right] < \Pr\left[\text{Game3}(1^\lambda)=1\right] - \text{negl}(\lambda)$. This would lead to $\text{negl}(\lambda) \leqslant |\Pr\left[\text{Game3}(1^\lambda)=1\right] - \Pr\left[\text{Game4}(1^\lambda)=1\right]|$ thus breaking the semantic security of the HE scheme.

In Game4, S is never used at authentication time. Thus, we can think of the challenger picking S only at verification time. For any Type 2 forgery $(\mathbf{m}^*, \mathcal{P}^*=(f,(\boldsymbol{\tau}_1^*,...,\boldsymbol{\tau}_n^*)), \boldsymbol{\sigma}^*)$, decryption leads to $\text{HE.Dec}(\mathbf{c}^*;\mathbf{sk}_{\text{HE}})=(\mathbf{M}_1^*,\ldots,\mathbf{M}_{N/\lambda}^*)$. W.l.o.g consider only one scalar output (*i.e.,* consider only the first extended vector $\mathbf{M}_1^*$). Denote by $\hat{\mathbf{M}}_1$ the first extended vector of the honestly generated ciphertext. Let $S'=\{i \in [\lambda] : \mathbf{M}_1^*[i]=\hat{\mathbf{M}}_1[i]\}$ be the indices on which the forged and honest extended vectors match. The adversary wins if the second and third steps of Game3 pass which only occurs if $S = S'$ (the protocol aborts if all the challenges have the same value). This event happens with probability $2^{-\lambda}$. Thus, $\Pr\left[\text{Game4}(1^\lambda)=1\right] \leqslant 2^{-\lambda}$. In turn, we see that

$$\Pr\left[\text{Game0}(1^\lambda)=1\right] \leqslant 2^{-\lambda}+\text{negl}(\lambda) \leqslant \text{negl}(\lambda)$$

which can be parameterized to be negligible in λ, concluding the proof. □

A.3 Security Proof for PE (§3.5.2)

Theorem 3.5.1: If the pseudorandom function F_K and the canonical HE scheme are at least λ-bit secure and if t is a λ-bit prime number, then, for any program $\mathcal{P}$ with authentications of bounded degree, PE is a secure authenticator and a PPT adversary has a probability of successfully cheating the verification negligible in λ.

Proof. We follow the security analysis sketched in Catalano and Fiore's information-theoretic homomorphic MAC [CF13]. We consider a PPT adversary $\mathcal{A}(1^\lambda)$. We design the following series of games.

Game0: This game is the forgery game based on Experiment 1 as $\mathbf{Exp}_{\mathcal{A}}[\text{PE},\lambda]$. The different types of forgeries (*i.e.,* Type 1 and 2) are defined in Appendix A.2. Recall that the scheme PE is said secure if $\forall$ PPT $\mathcal{A}$,

$$\Pr\left[\text{Game0}(1^\lambda)=1\right] \leqslant \text{negl}(\lambda).$$

Now let us design hybrid games by modifying Game0.

Game1: We modify the verification procedure in Game0 such that the challenger checks whether the program $\mathcal{P}$ is well-formed or not (*i.e.,* all inputs are authenticated) using probabilistic polynomial identity testing (see [CF13] Prop.1):

$$\Big|\Pr[\text{Game1}(1^\lambda)=1]-\Pr[\text{Game0}(1^\lambda)=1]\Big| \leqslant 2^{-\lambda}.$$

Game2: We modify Game1 by replacing the PRF F_K with random values in the authentication and verification procedures such that a random function outputs random values on the fly. By the pseudorandomness of F_K, it is clear that

$$\Big|\Pr\Big[\text{Game2}(1^\lambda)=1\Big]-\Pr\Big[\text{Game1}(1^\lambda)=1\Big]\Big| \leqslant \text{negl}(\lambda).$$

Game3: We modify only the verification procedure of Game2. For the labeled program $\mathcal{P}=(f(\cdot),(\boldsymbol{\tau}_1,...,\boldsymbol{\tau}_n))$, and a verification query $(\mathbf{m},\mathcal{P},\boldsymbol{\sigma}')$, parse $\boldsymbol{\sigma}'=(\mathbf{c}_0,...,\mathbf{c}_d)$.

- If $\mathbf{y}_0=\text{HE.Dec}(\mathbf{c}_0;\mathbf{sk}_{\text{HE}})\neq\mathbf{m}$, then reject.
- If $\mathcal{P}$ is not well-defined on T, then the challenger sequentially:
  1. $\forall i\in[n]$, s.t. $(\boldsymbol{\tau}_i,\cdot)\notin T$, samples uniformly at random $\mathbf{r}_{\boldsymbol{\tau}_i}$ and computes $\boldsymbol{\rho}=f(\mathbf{r}_{\boldsymbol{\tau}_1},...,\mathbf{r}_{\boldsymbol{\tau}_n})$.
  2. Decrypts the ciphertexts s.t. $\forall i\in[0{:}d]$, $\mathbf{y}_i=\text{HE.Dec}(\mathbf{c}_i;\mathbf{sk}_{\text{HE}})$.
  3. Computes $\mathbf{Z}{=}\boldsymbol{\rho}-\sum_{i=0}^{d}\mathbf{y}_i\cdot\alpha^i$.
  4. If $\mathbf{Z}{=}\mathbf{0}$ mod t then the challenger accepts (*i.e.,* outputs 1), otherwise it rejects.

As it is merely a syntactic change from Game2,

$$\Pr[\text{Game2}(1^\lambda)=1]\equiv\Pr[\text{Game3}(1^\lambda)=1].$$

Game4: We modify the verification procedure of Game3. Consider that $\mathcal{P}$ is well-defined over T. For all $i\in[n]$ such that $(\boldsymbol{\tau}_i,\cdot)\notin T$ (*i.e.,* one of the slot values was not in T), the challenger chooses a dummy $\boldsymbol{\sigma}_i$ generated for a random message for the corresponding slot. The challenger then computes $\hat{\boldsymbol{\sigma}}'=(\hat{\mathbf{c}}_0,...,\hat{\mathbf{c}}_d)\leftarrow\text{PE.Eval}(f(\cdot),\vec{\boldsymbol{\sigma}};\mathbf{evk})$. Now:

- Ⓐ If $\forall k{\in}[0{:}d]$ $\text{HE.Dec}(\mathbf{c}_k;\mathbf{sk}_{\text{HE}}){=}\text{HE.Dec}(\hat{\mathbf{c}}_k;\mathbf{sk}_{\text{HE}})$, then accept.
- Ⓑ If $\exists k$ s.t. $\text{HE.Dec}(\mathbf{c}_k;\mathbf{sk}_{\text{HE}}){\neq}\text{HE.Dec}(\hat{\mathbf{c}}_k;\mathbf{sk}_{\text{HE}})$, compute

$$\mathbf{Z}{=}\sum_{i=0}^{d}(\text{HE.Dec}(\mathbf{c}_i;\mathbf{sk}_{\text{HE}}){-}\text{HE.Dec}(\hat{\mathbf{c}}_i;\mathbf{sk}_{\text{HE}}))\cdot\alpha^i.$$

  If $\mathbf{Z}{=}\mathbf{0}$ mod t then accept, otherwise reject.

We show that the adversary has the same view as in Game3. Consider $(\mathbf{m}, \mathcal{P}, \boldsymbol{\sigma}' = (\mathbf{c}_0,...,\mathbf{c}_d))$ with $\mathcal{P} = (f, (\boldsymbol{\tau}_1,..., \boldsymbol{\tau}_n))$ well-defined on T. We distinguish two cases:

1. **All inputs were authenticated:** $\forall i \in [n]$, $(\boldsymbol{\tau}_i, \mathbf{m}_i) \in T$ and $\boldsymbol{\sigma}_i \leftarrow \text{PE.Auth}(\mathbf{m}_i, \boldsymbol{\tau}_i; \mathbf{sk})$. Recall that the challenger computes $\hat{\boldsymbol{\sigma}}' = (\hat{\mathbf{c}}_0,..., \hat{\mathbf{c}}_d) \leftarrow \text{PE.Eval}(f(\cdot), \vec{\boldsymbol{\sigma}}; \mathbf{evk})$. Now if Ⓐ occurs, then the answer is correct by the correctness of the HE scheme. If Ⓑ occurs, as the same values $\{\mathbf{r}_{\boldsymbol{\tau}_i}\}$ are generated, $\boldsymbol{\rho} = \text{PE.Ver}(\mathcal{P}, \hat{\boldsymbol{\sigma}}'; \mathbf{sk}) = \text{PE.Ver}(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})$. So accepting if $\mathbf{Z} = \mathbf{0}$ is the same as returning the output of $\text{PE.Ver}(\mathcal{P}, \boldsymbol{\sigma}'; \mathbf{sk})$.
2. **Some of the inputs were not authenticated:** $\exists i \in [n]$, s.t. $(\boldsymbol{\tau}_i, \cdot) \notin T$ (at least in one of its N components). Thus, by definition of the well-defined program, wires with those inputs are not used in the computation (*i.e.,* same output regardless of the value of those wires). This also holds after the homomorphic transformation and thus the dummy input chosen does not impact the computation.

As Game4 is only a syntactic change from Game3, $\Pr[\text{Game3}(1^\lambda)=1] \equiv \Pr[\text{Game4}(1^\lambda)=1]$.

Game5: We modify the verification procedure of Game4. We define a flag $\boxed{\text{Bad}}$ initially set to false. When verifying $(\mathbf{m}, \mathcal{P}, \boldsymbol{\sigma}')$, if the computation $\mathbf{Z}=\mathbf{0} \bmod t$, then the challenger rejects and sets $\boxed{\text{Bad}}$=True. By definition of the verification procedure, Type 1 and Type 2 forgeries are rejected leading to $\Pr[\text{Game5}(1^\lambda) = 1]=0$. Note that Game4 and Game5 are identical unless the event $\xi=$" $\boxed{\text{Bad}}$ is true" occurs. Thus

$$|\Pr[\text{Game4}(1^\lambda) = 1] - \Pr[\text{Game5}(1^\lambda) = 1]| \leqslant \Pr[\xi].$$

For the flag $\boxed{\text{Bad}}$ to be set, the challenger needs to compute $\mathbf{Z}$. Depending on the definition of $\mathcal{P}$ two cases occur:

1. $\mathcal{P}$ is well-defined and $\mathbf{Z}=\sum_{k=0}^{d}(\mathbf{y}_k - \hat{\mathbf{y}}_k) \cdot \alpha^k = 0 \bmod t$ where $\exists \hat{k} \in [0{:}d]$ s.t. $\mathbf{y}_{\hat{k}} \neq \hat{\mathbf{y}}_{\hat{k}}$. Call this constraint ξ_1.
2. $\mathcal{P}$ is not well-defined, $\mathbf{Z}=\boldsymbol{\rho} - \sum_{k=0}^{d} \mathbf{y}_k \cdot \alpha^k = 0 \bmod t$, and $\boldsymbol{\rho}$ is computed using at least one value r_{τ^*} that was not authenticated. Call this constraint ξ_2.

This leads to

$$\Pr[\xi] \leqslant \Pr[\mathbf{Z} = 0|\xi_1] + \Pr[\mathbf{Z} = 0|\xi_2]. \tag{A.1}$$

Observe that before the verification, there exists exactly t possible tuples $(\alpha, \{r_\tau\}_{\tau \in T})$ consistent with the adversary's view. The first probability in Eq. A.1 is bounded by d/t as the polynomial $\sum_{i=0}^{d}(\mathbf{y}_i - \hat{\mathbf{y}}_i) \cdot x^i$ has at most d zeros and there are at most t possible values for α. Indeed, finding these values is equivalent to finding the common zeros to N degree d polynomial equations. By defining the bivariate polynomial $\mathcal{Z}[X, Y] = \sum_{k=0}^{d}((\mathbf{y}_k[Y] -$

$\hat{\mathbf{y}}_k[Y]) \cdot X^i)$, we can rewrite the above polynomial as $\mathcal{Z}[X,Y]=\sum_{i=0}^{d}\sum_{j=0}^{N} z_{ij} \cdot Y^j X^i$. To find the zeros in X for all possible values in Y, we can project on the basis generated by the $\{Y^j\}_j$ leading to N polynomial equations of the form $\sum_{i=0}^{d} z_{ij}x^i=0$, for $j\in[0{:}N-1]$, each with at most d zeros. Thus, the overall number of zeros in the X-dimension of the bivariate polynomial is the intersection of all these zeros which cannot be more than d. For the second probability of Eq. A.1, as the program is not well-defined, $\boldsymbol{\rho}$ can be seen as a non-constant polynomial in $\{r_{\tau^*}\}_{\tau\notin T}$. As no query has involved τ^*, the adversary can only guess its value with probability $1/t$. By the polynomial identity lemma, $Pr[Z=0|\xi_2]\leqslant\frac{d}{t}$. Thus, $\Pr[\xi] \leqslant \frac{2d}{t}$, which in turn leads to

$$\Pr[\text{Game0}(1^\lambda) = 1] \leqslant \frac{2d}{t}+2^{-\lambda}+\text{negl}(\lambda) \leqslant \text{negl}(\lambda),$$

which can be parameterized to be negligible in λ, concluding the proof. □

A.4 Security Proof for the Polynomial Compression Protocol (§3.5.3)

In this section, we analyze the security of our polynomial compression protocol (PoC).

Theorem A.4.1. PE achieving the conditions of Th.3.5.1 and combined with the polynomial compression of Fig. 3.5 is a secure authenticator (§3.3.3 and Appendix A.1).

Proof. We focus on the verification procedure of PE to estimate the probability of check (3) passing with wrongful polynomials. Let W be the event that the verifier accepts. Consider a malicious server that deviates from the honest prover. The view of the malicious prover (denoted by a tilde) is different from that of the honest one, *i.e.,* $\tilde{\boldsymbol{\sigma}}\neq\boldsymbol{\sigma'}$. Consider the messages sent by the server (m_1 and m_2) depend, respectively, on all the information previously sent by the client. For any message k, we define E_k as the event that the message $\tilde{\mathsf{m}}_k$ sent by the prover agrees with the message m_k that the honest prover would have sent on the same view $\tilde{\boldsymbol{\sigma}}$. By the law of total probability, we can write $\Pr[\text{W}] \leqslant \Pr[\text{W}|\bar{E}_2] + \Pr[\text{W}|E_2]$. The probability $\Pr[\text{W}|\bar{E}_2]$ is bounded by

$$\Pr[\text{W}|\bar{E}_2] \leqslant \Pr\left[\boldsymbol{\rho}(\alpha)=\sum_{i=0}^{d}\tilde{w}_i \cdot \alpha^i \middle| \tilde{\mathsf{m}}_2 \neq \mathsf{m}_2, \alpha \leftarrow \mathbb{Z}_t^*\right].$$

Following the security proof of PE (Appendix A.3), this probability is upper bounded by $d/(t-1)$ by the polynomial identity lemma. The probability $\Pr[\text{W}|E_2]$ can be decomposed by the law of total probability as $\Pr[\text{W}|E_2] \leqslant \Pr[\text{W}|E_2 \wedge \bar{E}_1] + \Pr[\text{W}|E_2 \wedge E_1]$. The probability $\Pr[\text{W}|E_2 \wedge \bar{E}_1]$ is bounded by

$$\Pr[\text{W}|E_2 \wedge \bar{E}_1]\leqslant\Pr\left[\begin{matrix}\tilde{w}_0=\mathbf{m}(\delta)\\ H_{\delta,\beta}(\boldsymbol{\sigma'}) \stackrel{?}{=} \sum_{i=0}^{d}\tilde{w}_i\cdot\beta^i\end{matrix}\middle|\begin{matrix}\tilde{\mathsf{m}}_1\neq\mathsf{m}_1\\ \tilde{\mathsf{m}}_2=\mathsf{m}_2\end{matrix}\right]\leqslant\frac{d+N}{t}.$$

The inequality holds by the property of the hash (which is itself derived from the polynomial identity lemma); the malicious prover's knowledge of (δ, β) arrives after $\tilde{\mathsf{m}}_1$ (see [FGP14, Th.2] for the analysis of such polynomial hash collision probability). The probability $\Pr[\mathrm{W}|E_2 \wedge E_1]$ is upper bounded by $\Pr[\mathrm{W}|E_1]$ which is, by the polynomial identity lemma on $H_{(\cdot,0)(\tilde{\boldsymbol{\sigma}})}$, bounded by $\frac{d+N}{t}$. Overall, the probability that the verifier accepts the protocol conversing with the malicious prover is bounded by

$$\Pr[\mathrm{W}] \leqslant \frac{2(d+N)}{t} + \frac{d}{t-1}.$$

As a result, with an appropriate choice of parameters, the prover can only cheat the verifier with negligible probability. □

A.5 Security Proof for the Re-Quadratisation Protocol (§3.5.4)

In this section, we describe in more depth our interactive re-quadratization protocol (ReQ) shown in Fig 3.6 and analyse its security.

Figure 3.6 presents the ReQ protocol in detail. At the G-th multiplicative gate, the server holds $\boldsymbol{\sigma}'=(\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4)$. It interacts with the client to obtain $\bar{\boldsymbol{\sigma}}=(\mathbf{c}_0, \bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2)$. The server sends the client the higher degree terms $\mathbf{c}_3$ and $\mathbf{c}_4$. The client decrypts them to $\mathbf{y}_3$ and $\mathbf{y}_4$ by using its secret key $\mathbf{sk}_{\mathrm{HE}}$. For security reasons, ReQ introduces random blindings that need to be accounted for in further computations. It samples uniformly two random numbers $\kappa_1, \kappa_2 \leftarrow \mathbb{Z}_t$ and two random polynomials $\mathbf{r}, \bar{\mathbf{r}} \leftarrow \mathcal{R}_t$. It sets $\bar{\mathbf{y}}_2 = \alpha\kappa_1\mathbf{y}_3 + \alpha^2\kappa_2\mathbf{y}_4 + \mathbf{r}$. We introduce the Shift$(\cdot)$ function that takes as input the sub-circuit $\mathcal{P}_G$ that leads to gate G, the authenticator secret key $\mathbf{sk}$, the identifiers $\boldsymbol{\tau}$, and the set Ω of previously used randomness (initially empty). This function returns the polynomial offset that was introduced by the addition of blindings in previous re-quadratizations. We call Δ_G the result of Shift$(\mathcal{P}_G, \boldsymbol{\tau}, \mathbf{sk}, \Omega)$. It then evaluates $\bar{\mathbf{y}}_1 = \bar{P}(\alpha) = \alpha^3\mathbf{y}_4 + \alpha^2\mathbf{y}_3 - \alpha\bar{\mathbf{y}}_2 - \Delta_G + \bar{\mathbf{r}}$. It encrypts $\bar{\mathbf{c}}_2 = \mathrm{HE.Enc}(\bar{y}_2; \mathbf{pk}_{\mathrm{HE}})$ and $\bar{\mathbf{c}}_1 = \mathrm{HE.Enc}(\bar{y}_1; \mathbf{pk}_{\mathrm{HE}})$ and sends them to the server. It also updates the list Ω with $(\mathbf{r}, \bar{\mathbf{r}})$. The server appends them to the corresponding ciphertexts (*i.e.*, $\bar{\mathbf{c}}_i \leftarrow \mathbf{c}_i + \bar{\mathbf{c}}_i$, for $i \in \{1, 2\}$). To remove the final offset, the verification procedure at the client is slightly modified. As a result, the client can pre-process (at least part) of the offsets and use them directly during the interactive protocol. Let us now analyze ReQ's security.

Theorem A.5.1. PE achieving the conditions of Th.3.5.1 and combined with the ReQ protocol from Fig. 3.6 is a secure authenticator (§3.3.3 and Appendix A.1).

Proof. The security of PE remains unchanged by the correctness of the ReQ that follows from its construction: evaluated on the secret point α, $\bar{\boldsymbol{\sigma}}$ equals the same evaluation of the original degree-four $\boldsymbol{\sigma}'$ (up to a deterministic shift Δ_G). The quantity $\Delta_G = \mathrm{Shift}(\mathcal{P}_G, \boldsymbol{\tau}, \mathbf{sk}, \Omega)$ enables us to remove the offset introduced by the previous

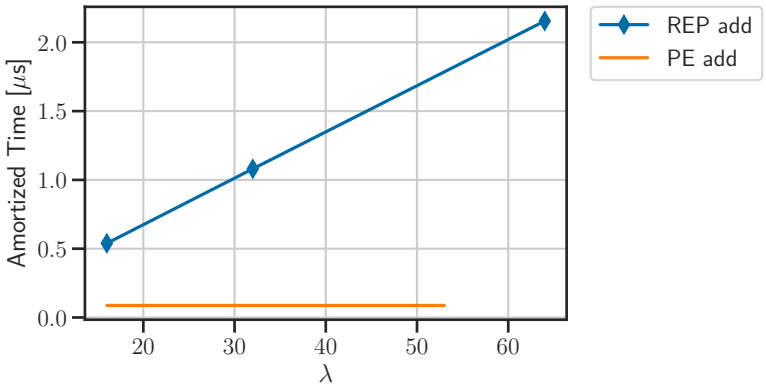


Figure A.1: Amortized runtime of the homomorphic addition operation vs. the authenticator security parameter λ.

randomness. Hence, we subtract it to the term of degree one. As κ_1 and κ_2 are uniformly random values in $\mathbb{Z}_t$, the quantity $\alpha\kappa_1\mathbf{y}_3+\alpha^2\kappa_2\mathbf{y}_4$ is a random linear combination of $\mathbf{y}_3$ and $\mathbf{y}_4$. The addition of the random polynomial $\mathbf{r}$ ensures perfect secrecy, hence the result $\bar{\mathbf{y}}_2$, and even more its encryption, reveal nothing about α: $\bar{\mathbf{y}}_2$ is indistinguishable from a random value in $\mathcal{R}_t$. The random polynomial $\bar{\mathbf{r}}$ acts again as a blinding value thus providing perfect secrecy of the polynomial $\bar{\mathbf{y}}_1$. The server knows only that $\bar{\mathbf{c}}_1$ and $\bar{\mathbf{c}}_2$ encrypt $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$, respectively. Although the (bivariate) polynomial $X^4{\cdot}\mathbf{y}_4+X^3{\cdot}\mathbf{y}_3+X^2{\cdot}(\mathbf{y}_2-\bar{\mathbf{y}}_2)+X(\mathbf{y}_1-\bar{\mathbf{y}}_1+\bar{\mathbf{r}}-\Delta_G)$ admits α as one of its roots for all N dimensions, the server cannot find the roots as it does not know its coefficients; the random polynomial $\bar{\mathbf{r}}$ is kept secret. Consequently, both ciphertexts $\bar{\mathbf{c}}_1$ and $\bar{\mathbf{c}}_2$ reveal nothing about the secrets (*i.e.,* $\mathbf{sk}, \Omega$). □

A.6 VERITAS Security Configuration vs. Overhead

We analyze VERITAS' overhead with respect to the authenticator security parameter λ for a fixed evaluation circuit. Figure A.1 presents our experimental results for the BFV homomorphic addition operation ($\log N$=14, $\log q$=438) with $\lambda\in\{16, 32, 64\}$. We observe that REP induces a linear computation (and communication) overhead with respect to λ. This is unsurprising since with REP there is an expansion factor of λ due to the replication and the challenges (§3.4.2). On the contrary, PE's overhead is constant with respect to λ since its security is linked only to the size of the plaintext space (§3.5.2). For appropriate parameterization of the HE scheme, increasing λ does not affect the ciphertexts. However, we remind that PE's efficiency is directly linked to the authenticator's growth at every multiplication (*e.g.,* §3.6.3.4 and 3.6.3.3). Moreover, we note that in the Lattigo implementation, the HE circuit constraints the plaintext space and PE can be parameterized to achieve a security level of up to λ=53 (depending on the ring size and the evaluation circuit). We emphasize that this is only due to the specific implementation and using arbitrary-precision arithmetic would overcome this issue.

Appendix B

Appendix for PELTA (§4)

B.1 Lattice-based Hard Problems

We recall the Module versions of the SIS [Ajt96] and LWE [Reg09, PR06] problems (*i.e.,* MSIS and MLWE resp.) [LS15, DKL+18]. Both problems are defined over the ring $\mathcal{R}_q$ for $q \in \mathbb{Z}^+$.

Definition 2. ($\text{MSIS}_{\kappa_{\text{sis}},m,\beta_{SIS}}$) D2.11 [LNP22] For a Module-SIS problem with parameters $m, \kappa_{\text{sis}} > 0$, and $\beta_{SIS} > q$, the objective is to find for a given matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa_{\text{sis}} \times m}$ a vector $\mathbf{x} \in \mathcal{R}_q^m$ such that $\mathbf{Ax} = \mathbf{0}$ over $\mathcal{R}_q$ and $\|\mathbf{x}\|_\infty < \beta_{SIS}$. We say that a PPT adversary $\mathcal{A}$ has advantage ε in solving $\text{MSIS}_{\kappa_{\text{sis}},m,\beta_{SIS}}$ if:

$$\Pr\left[0 < \|\mathbf{x}\|_\infty \leqslant \beta_{SIS} \wedge \mathbf{Ax} = \mathbf{0} \text{ in } \mathcal{R}_q | \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa_{\text{sis}} \times m}; \mathbf{x} \xleftarrow{\$} \mathcal{A}(A)\right] \geq \varepsilon$$

Definition 3. ($\text{MLWE}_{m,\lambda_{\text{lwe}},\chi}$) D2.12 [LNP22] For a Module-LWE problem with parameters $m, \lambda_{\text{lwe}} > 0$, and an error distribution χ over $\mathcal{R}$, the objective is for a PPT adversary $\mathcal{A}$ to distinguish $(\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{R}_q^{m \times \lambda_{\text{lwe}}} \times \mathcal{R}_q^m$ from $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ for a given matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m \times \lambda_{\text{lwe}}}$, a secret vector $\mathbf{s} \xleftarrow{\$} \chi_{\text{lwe}}^{\lambda}$, and an error vector $\mathbf{e} \leftarrow \chi^m$. We say that $\mathcal{A}$ has advantage ε in solving $\text{MLWE}_{m,\lambda_{\text{lwe}},\chi}$ if:

$$\Bigg| \Pr\left[b = 1 | \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m \times \lambda_{\text{lwe}}}; \mathbf{s} \xleftarrow{\$} \chi_{\text{lwe}}^{\lambda}; \mathbf{e} \xleftarrow{\$} \chi^m; b \leftarrow \mathcal{A}((\mathbf{A}, \mathbf{As} + \mathbf{e}))\right] - \Pr\left[b = 1 | \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m \times \lambda_{\text{lwe}}}; \mathbf{t} \xleftarrow{\$} \mathcal{R}_q^m; b \leftarrow \mathcal{A}((\mathbf{A}, \mathbf{t}))\right] \Bigg| \geq \varepsilon.$$

The parameterization of the different constant $\kappa_{\text{sis}}, \lambda_{\text{lwe}}, q, \beta_{SIS}, m$ are chosen to make the problem resistant to known attacks and as in prior work [BLS19, ALS20, ENS20, LNS21a]

we set the root Hermite factor $\delta < 1.0045$. This gives a negligible advantage in solving the MSIS and MLWE problems.

B.2 Influence of the Number of RNS Sub-rings

Table B.1 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 B.1: PELTA's performance for the local key-generation protocol (§4.4.4.1) 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.3 Parameterization

We detail the different parameters used in our construction and present, in Table B.2, 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 [ESS+19, ESLL19a]. $\log q_j$ corresponds to the number of bits of the FHE sub-ring modulus. κ_{sis} and λ_{lwe} are respectively the MSIS and MLWE dimensions in the sub-ring $\mathfrak{R}_{q_j}$.

Table B.2: Parameters for the key generation (PN13).

$\log d$	κ_{sis}	λ_{lwe}	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

B.4 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 B.1 presents the prover's operations while the verifier's are in Figure B.2.

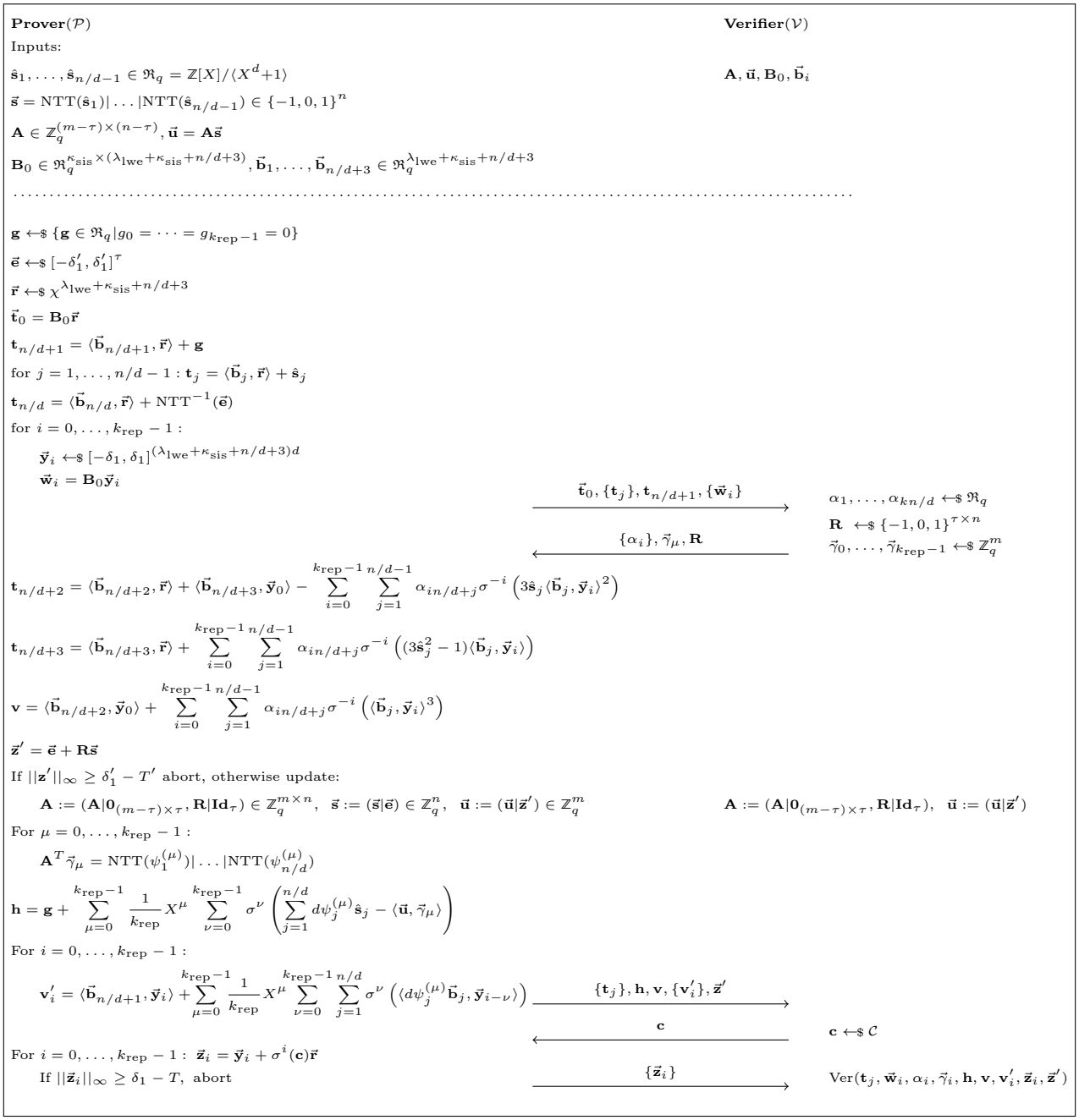


Figure B.1: Interactive proof generation of a ternary solution (of size n inputs in $\mathbb{Z}_q$) to an unstructured linear relation with additional approximate bound proof (ABP). For a polynomial ring $\mathfrak{R}_q{=}\mathbb{Z}_q[X]/\langle X^d{+}1\rangle$, N a power-of-two, κ_{sis} and λ_{lwe} 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 : $$\sum_{\mu=0}^{k_{\mathrm{rep}}-1} \frac{1}{k_{\mathrm{rep}}} X^\mu \sum_{\nu=0}^{k_{\mathrm{rep}}-1} \sum_{j=1}^{n/d} \sigma^\nu \left(N\psi_j^{(\mu)} \langle \vec{\mathbf{b}}_j, \vec{\mathbf{z}}_{i-\nu \mod k_{\mathrm{rep}}} \rangle \right) + \langle \vec{\mathbf{b}}_{n/d+1}, \vec{\mathbf{z}}_i \rangle \overset{?}{=} \mathbf{v}'_i + \sigma^i(\mathbf{c})(\tau + \mathbf{t}_{n/d+1} - \mathbf{h})$$

Figure B.2: Verification equations for Figure B.1.

Appendix C

Appendix for CRISP (§5)

C.1 Zero-Knowledge Circuit Evaluation

We now present how ZKB^{++} [CDG^{+}17] operates in more details:

(2,3)-decomposition of a Circuit from [GMO16, CDG^{+}17] For a function f represented by a circuit $\mathcal{C}$, a (2,3)-decomposition consists of a series of algorithms $(\textsc{Share}, \textsc{Output}, \textsc{Rec}) \cup \textsc{Update}$, with $\textsc{Share}$ surjective, and allows to create three separate views of the circuit. Then, ZKB^{++} enables to prove knowledge of a secret input x such that $f(x) = y$, with y the publicly known output. For an iteration k, the *views* for player i is denoted by a vector $\mathbf{View}_i^k = \{\text{view}_i^0, \cdots, \text{view}_i^{N_g}\}$.

- The $\textsc{Share}$ algorithm splits a secret x:

$$\left(\text{view}_1^0, \text{view}_2^0, \text{view}_3^0\right) = \textsc{Share}(x, \mathbf{k}_1, \mathbf{k}_2, \mathbf{k_3}),$$

  with $\mathbf{k}_i$ being a random tape, $\forall i \in \{1, 2, 3\}$.
- $\mathcal{F} = \bigcup_{i=1}^{N_g} \{\phi_1^j, \phi_2^j, \phi_2^j\}$, where N_g is the number of gates in $\mathcal{C}$ and ϕ_i^j is the j-th gate of player i.
- The $\textsc{Update}$ algorithm evaluates the gates into the views:

$$\text{view}_i^{j+1} = \phi_i^j(\text{view}_i^j, \text{view}_{i+1}^j, \mathbf{k}_i, \mathbf{k}_{i+1}),$$

  where $j \in [0, N_g - 1], \forall i \in \{1, 2, 3\}$.
- The $\textsc{Output}$ algorithm returns the output wires:

$$y_i = \textsc{Output}(\text{view}_i^{N_g}), \forall i \in \{1, 2, 3\}.$$

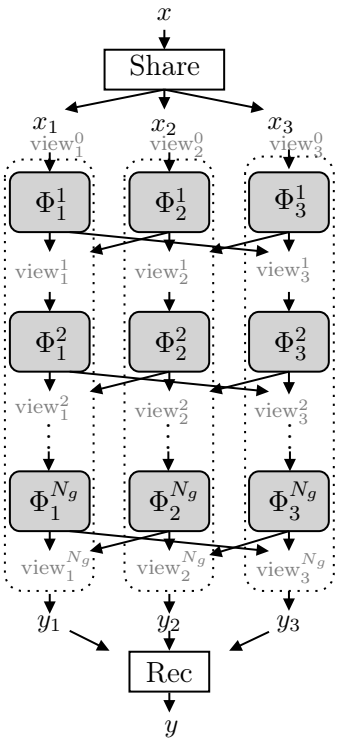


Figure C.1: (2,3)-decomposition of a circuit.

- Rec reconstructs the output: $y = \text{Rec}(y_1, y_2, y_3)$.

The intermediary functions ϕ_i^j are defined by running a linear decomposition on $\mathcal{C}$ such that:

- Each player i has wire $\mathbf{w}^i$ and $w_k^{(1)} + w_k^{(2)} + w_k^{(3)}$ is equal to the wire state of the k-th gate of $\mathcal{C}$.
- **Addition by a constant** d**:** $\forall i \in \{1,2,3\}$

$$w_b^{(i)} = \begin{cases} w_a^{(i)} + d & \text{if } i = 1, \\ w_a^{(i)} & \text{otherwise.} \end{cases}$$

- **Multiplication by a constant** d**:** $\forall i \in \{1,2,3\}$

$$w_b^{(i)} = d \cdot w_a^{(i)}.$$

- **Binary addition:** $\forall i \in \{1,2,3\}$

$$w_c^{(i)} = w_a^{(i)} + w_b^{(i)}.$$

- **Binary multiplication:** $\forall i \in \{1,2,3\}$

$$w_c^{(i)} = w_a^{(i)} \cdot w_b^{(i)} + w_a^{(i+1)} \cdot w_b^{(i)} + w_a^{(i)} \cdot w_b^{(i+1)} + R_i(c) - R_{i+1}(c),$$

  where $R_i(c)$ is the c-th output of a pseudo-random generator seeded with $\mathbf{k}_i$.

The 2-privacy property ensures that revealing two views (*i.e.,* opening two players) does not leak information about the witness. Overall, the ZKB^{++} protocol works as follows:

(i) The prover emulates three players. For each iteration $i \in [1,t]$, each player $j \in \{0,1,2\}$ evaluates the (2,3)-decomposition of the circuit and:

  – commits to:

$$\left[C_j^{(i)}, D_j^{(i)}\right] \leftarrow \left[H\left(k_j^{(i)}, x_j^{(i)}, \mathbf{View}_j^i\right), k_j^{(i)} \| \mathbf{View}_j^i\right],$$

  – and lets $a^{(i)} = \left(y_1^{(i)}, y_2^{(i)}, y_3^{(i)}, C_1^{(i)}, C_2^{(i)}, C_3^{(i)}\right)$.

(ii) The prover computes the challenge $e{=}H\left(a^{(1)}, \ldots, a^{(t)}\right)$ and reads it as a value $e^{(i)}{\in}\{1,2,3\}$, for all $i{\in}[1,t]$. For all $i{\in}[1,t]$, the prover lets $b^{(i)}{=}$ $\left(y_{e^{(i)}+2}^{(i)}, C_{e^{(i)}+2}^{(i)}\right)$ and

$$z^{(i)} \leftarrow \begin{cases} \left(\mathbf{View}_2^{(i)}, \mathbf{k}_1^i, \mathbf{k}_2^i\right) & \text{if } e^{(i)} = 1, \\ \left(\mathbf{View}_3^{(i)}, \mathbf{k}_2^{(i)}, \mathbf{k}_3^{(i)}, x_3^{(i)}\right) & \text{if } e^{(i)} = 2, \\ \left(\mathbf{View}_1^{(i)}, \mathbf{k}_3^{(i)}, \mathbf{k}_1^{(i)}, x_3^{(i)}\right) & \text{if } e^{(i)} = 3. \end{cases}$$

(iii) Then, the prover computes the proof

$$p{=}\left[e, \left(b^{(1)}, z^{(1)}\right), \left(b^{(2)}, z^{(2)}\right), \cdots, \left(b^{(t)}, z^{(t)}\right)\right].$$

(iv) The verifier, for each iteration $i \in [1,t]$, reconstructs the input and output views that were not given as part of the proof by:

  – running the circuit for player e^i with the information in the proof, and
  – computing C_j^i, D_j^i, and a^i, with the information provided in b^i.

(v) Finally, the verifier computes the challenge $e'{=}H\left(a^{(1)}, \ldots, a^{(t)}\right)$ and checks that $e \stackrel{?}{=} e'$ is true.

In particular, ZKB^{++} [CDG$^+$17] is a Σ-protocol for languages of the type $\{y \mid \exists x \text{ s.t.} y{=}\Phi(x)\}$, where $\Phi(\cdot)$ is the representation of the circuit. With randomized runs, the verifier builds confidence in the prover's knowledge of the secret. The number of iterations is determined according to the desired soundness: For instance, to prove the knowledge of a message

that yields a specific SHA-256 digest, a security level of 128-bits requires 219 iterations. The proof size is linked to the number of iterations but also to the number of gates that require non-local computations (*e.g.,* AND for Boolean circuits, multiplication for arithmetic ones). Compared to earlier work, *i.e.,* ZKBoo [GMO16], ZKB^{++} reduces the proof size by not sending information that can be computed by the verifier. The security of ZKB^{++} is based on the quantum random oracle model.

Overall, it achieves the following properties:

(a) *2-privacy*, opening two out of the three players' views to the verifier reveals no information regarding the secret input,

(b) *special soundness*, a correct execution yields a valid witness with soundness error linked to the number of iterations, and

(c) *completeness*, an honest execution of ZKB^{++} ensures a correct output.

In the remainder of this chapter, we will refer to the ZKB^{++} protocol described above as ZKCE.

C.2 CRISP's privacy proof (§5.5.1)

Proposition C.2.1. Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $H(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N, q, \chi_{\text{enc}}, \chi_{\text{key}}, \chi_{\text{err}})$ and BDLOP $(\delta_1, k, \kappa_{sis}, \kappa_{lwe}, q, N)$ schemes have been configured to ensure post-quantum security, that the circuit $\mathcal{C}$ is a valid (2,3)-decomposition, and that the cryptographic commitment $\text{Com}(\cdot)$ is hiding and binding. Then, our solution achieves privacy by yielding nothing more than the result $\hat{m}$ of the computation on the user's data $\{\mathbf{x}_i\}$.

Proof. To prove the privacy of CRISP, we construct an ideal simulator whose outputs are indistinguishable from the real outputs of CRISP's transfer and release phases.

Transfer Phase. In the random oracle model (ROM), consider an ideal-world simulator $\mathcal{S}_t$ and any corrupted probabilistic polynomial time (PPT) service provider (*i.e.,* the Verifier). Without loss of generality, we consider only one round of communication between the user and service provider (*i.e.,* one set of challenges). The simulator $\mathcal{S}_t$ generates a public-private key pair $(\mathbf{pk}'_{\text{HE}}, \mathbf{sk}'_{\text{HE}})$. Following the encryption protocol, $\mathcal{S}_t$ samples $\mathbf{r}'_0 \leftarrow \chi_{\text{enc}}$ and $\mathbf{e}'_0, \mathbf{e}'_1 \leftarrow \chi_{\text{err}}$ and computes the encryption of a random input vector $\mathbf{m}'$ into $\mathbf{ct}'$. Similarly, it samples a commitment noise vector $\mathbf{r}'_c$ and commits $(\mathbf{r}'_0, \mathbf{e}'_0, \mathbf{e}'_1)$ into $\mathbf{C}'_{\text{BDLOP}}$. Using a random nonce, the simulator also hashes $H(\mathbf{m}'[1])$.

Without loss of generality, this can be extended to all components of $\mathbf{m}'$. $\mathcal{S}_t$ then sends $\{\mathbf{ct}', \mathbf{C}'_{\text{BDLOP}}, \text{H}(\mathbf{m}'[1])\}$ to the service provider. The view of the service provider in the real protocol comprises $\{\mathbf{ct}, \mathbf{C}_{\text{BDLOP}}, \text{H}(msg)\}$. By the semantic security of the underlying encryption scheme [CKKS17], the hiding property of the BDLOP commitment scheme (see [BDL$^+$18b]), and the hiding and binding properties of the commitments instantiated by $H(\cdot)$, the simulated view is indistinguishable from the real view.

Following the properties of the BDLOP commitment scheme [BD16], for each iteration of the bound proof with challenge $\text{d}\in\{0,1\}$, the simulator $\mathcal{S}_t$ can randomly draw z' and $\mathbf{r}'_z$ with small norm and set $\mathbf{t}$=BDLOP$(z', \mathbf{r}'_z)-\text{d}\mathbf{C}_{\text{BDLOP}}$ (see [BD16]). The simulator then commits to $\mathbf{t}$ in the bound proof protocol. Both ideal and real distributions are indistinguishable by the hiding property of the auxiliary commitment.

In parallel, following [GMO16], given $e \in \{1, 2, 3\}$, the simulator $\mathcal{S}_t$ sequentially

- Evaluates the SHARE function on the vector $\mathbf{m}'$, the encryption noises $\mathbf{e}'_0$, $\mathbf{e}'_1$, and $\mathbf{r}'_0$ and commitment noises $\mathbf{r}'_c$. We denote the result by $(\text{view}'^0_1, \text{view}'^0_2, \text{view}'^0_3)$ (See § 5.3.2).
- Samples random tapes $\mathbf{k}'_e$, $\mathbf{k}'_{e+1}$.
- Evaluates the arithmetic circuit according to: If gate c is linear, it defines view'^c_e and view'^c_{e+1} using ϕ^c_e and ϕ^c_{e+1}. If gate c is a multiplication one, it samples uniformly at random view'^c_{e+1} and uses ϕ^c_e to compute view'^c_e.
- Once all the gates are evaluated and the vectors of views $\mathbf{View}'_e$ and $\mathbf{View}'_{e+1}$ are defined (see § 5.3.2), the simulator computes the respective outputs y'_e and y'_{e+1}.
- Computes $y'_{e+2} = y - (y'_e + y'_{e+1})$.
- Computes z'_e following step (ii) of the ZKB^{++} protocol using $\mathbf{View}'_{e+1}$, $\mathbf{k}'_e$, $\mathbf{k}'_{e+1}$ (and optionally x'_{e+2} depending on the challenge).
- Outputs (z'_e, y'_{e+2}).

The simulator $\mathcal{S}_t$ follows a protocol similar to the original ZKB^{++} protocol. The only difference is that for a multiplicative gate c, the simulated view value view'^c_{e+1} is sampled uniformly at random, whereas the original view value view^c_{e+1} is blinded by adding $R_i(c) - R_{i+1}(c)$, with $R_i(c)$ and $R_{i+1}(c)$ the outputs of a uniformly random function sampled using the tapes $\mathbf{k}_e$ and $\mathbf{k}_{e+1}$. Thus, the distribution of view^c_{e+1} is uniform and view'^c_{e+1} follows the same distribution in the simulation. Therefore, the ZKB^{++} simulator's output has the same distribution as the original transcript (z_e, y_{e+2}) the output of the simulator $\mathcal{S}_t$ is indistinguishable from the valid transcript to a corrupted verifier. Following the ideal functionality of $\mathcal{S}_t$, the ideal view of the service provider (*i.e.*, $\{\mathbf{ct}', \mathbf{C}'_{\text{BDLOP}}, \text{H}(\mathbf{m}'[1]), P'\}$)

is indistinguishable from the real view (*i.e.,* $\{\mathbf{ct}, \mathbf{C}_{\text{BDLOP}}, \text{H}(msg), P\}$, with P the real ZKB^{++} proof). Thus, the ideal and real outputs are indistinguishable for the corrupted PPT service provider proving the privacy property of CRISP's transfer phase. □

Release Phase. We construct a second simulator $\mathcal{S}_r$ to prove that CRISP's release protocol (Section 5.4.5) reveals nothing more than the result $\hat{m}$ to a honest but curious verifier. A different simulator is required, as the release phase is independent of the transfer phase. We consider that $\mathcal{S}_r$ knows the blinding function ahead of time (*i.e.,* it knows $(\boldsymbol{\nu}, \boldsymbol{\eta})$) for the real conversation leading to the service provider accepting $\hat{\mathbf{m}}$. Upon reception of the first message $\{\mathbf{ct}', B_{\boldsymbol{\nu},\boldsymbol{\eta}}(\mathbf{ct}'), C_0, f(\cdot)\}$ such that $\text{HE.Dec}(\mathbf{ct}'; \mathbf{sk}_{\text{HE}}) = \hat{\mathbf{m}}$, $\mathcal{S}_r$ creates $\hat{\mathbf{m}}_B$ using the blinding parameters. The simulator commits to C_1'=Com$(\hat{\mathbf{m}}, \hat{\mathbf{m}}_B)$, which is indistinguishable from C_1 to the curious verifier according to the hiding property of the commitment scheme. After receiving an opening for C_0, the simulator opens C_1' to $\hat{\mathbf{m}}$ and $\hat{\mathbf{m}}_B$, which sustain the verifier checks as defined in Section 5.4.5. The binding property of the commitment scheme asserts that $(\boldsymbol{\nu}, \boldsymbol{\eta})$ is used for the blinding. The aforementioned conversation between the prover and verifier is indistinguishable from the real conversation. By checking the function $f(\cdot)$, and as the service provider is honest-but-curious, the user is assured that the service provider evaluated $f(\cdot)$ and is not using her as a decryption oracle. If the user deems the function inadmissible, she aborts. □

C.3 CRISP's integrity (§5.5.2)

Proposition C.3.1. Consider a series of messages $\{msg_i\}$ certified by the data source with a digital signature scheme $\sigma(\cdot)$ that uses a cryptographic hash function $\text{H}(\cdot)$ with nonces. Assume that the parameters of the CKKS $(N, q, \chi_{\text{enc}}, \chi_{\text{key}}, \chi_{\text{err}})$ and BDLOP $(\delta_1, k, \kappa_{sis}, \kappa_{lwe}, q, N)$ schemes have been configured to ensure post-quantum security, that the ZKB^{++} protocol execution of $\mathcal{C}$ achieves soundness κ_{ZK}, that the blinding function $B_{\boldsymbol{\nu},\boldsymbol{\eta}}$ is hiding, and that the cryptographic commitment $\text{Com}(\cdot)$ is hiding and binding. Then, our solution achieves integrity as defined in Section 5.2.3, as it ensures with soundness κ_{ZK} that the output $\hat{m}$ is the result of the computation on the user's data.

Proof. Let us consider a cheating user as defined in Section 5.2.2. She wants to cheat the service provider in obtaining from the public SIMD circuit $f(\cdot)$ a result that is not consistent with the certified data. The public function evaluated by the service provider is $f(\cdot)$ and returns $\hat{m}$ on the series $\{msg_i\}$ of data signed by the data source with the signature scheme $\sigma(\cdot)$. We interchangeably denote by $f(\cdot)$ the public function in the plaintext and ciphertext domains. By the property of the CKKS scheme, the ciphertext $\mathbf{ct}'$ can be decrypted correctly using the secret key $\mathbf{sk}$. As stated in [GMO16] adapted to [CDG$^+$17], the binding property of the commitments used during the MPC-in-the-head

guarantees that the proof Π contains the information required to reconstruct $\mathbf{View}_e$ and $\mathbf{View}_{e+1}$. Given three accepting transcripts (*i.e.,* one for each challenge), the verifier can traverse the decomposition of the circuit from the outputs to the inputs, check every gate and reconstruct the input. By surjectivity of the ZKB^{++} decomposition function, the verifier can reconstruct x' s.t. $f(x') = y$ proving the 3-special soundness property (see proof of Proposition 4.2 in [GMO16]). The completeness property of the ZKCE evaluation follows directly from the construction of the (2,3)-decomposition of the circuit. Thus, from a correct execution of τ iterations of the protocol (parameterized by the security parameter κ_{ZK}), a user attempting to cheat the ZKB^{++} execution will get caught by the service provider with probability at least $1-2^{-\kappa_{\text{ZK}}}$. Hence, a malicious but rational user can only cheat by tampering with the data before they are input to the circuit, *i.e.,* the input messages or the encryption parameters. As the user is rational, she samples proper noise for the BDLOP commitment; otherwise, she would lose either privacy or utility: not sampling noise from $\chi^{(\kappa_{\text{sis}}+\lambda_{\text{lwe}}+1)}$ would lead to an improperly formatted commitment, and thus, a potential loss in utility, as the service provider would reject it using a proof of opening of the commitment. By the collision-resistance property of the hash function, it is computationally infeasible for the user to find a collision and thus a tampered message yielding the same hash.

By lemma 10 from Baum *et al.* [BD16], the bound proof is correct and offers special soundness: the service provider will detect a cheating user that samples malicious noises to distort the encryption, with probability at least $1 - 2^{-\kappa_{\text{ZK}}}$. Note that in the case of an abort, the protocol is simply re-executed.

Finally, during the release protocol, the integrity of the computation's output $\hat{m}$ is protected by the hiding property of commitment C_0, the hiding property of the blinding function (seen as a one-time-pad shift cipher in $\mathbb{Z}_q$ which achieves perfect secrecy of $\nu{\cdot}x$, *i.e.,* it is impossible for the user to find ν and blind another result as $\hat{m}_B$), and the binding property of C_1 [GMO16]. Therefore, in CRISP users can only cheat with probability at most $2^{-\kappa_{\text{ZK}}}$. □

C.4 Approximate Homomorphic Encryption

Cheon *et al.* recently introduced the CKKS cryptosystem [CKKS17] (improved in [CHK$^+$18b]), an efficient and versatile leveled homomorphic scheme for approximate arithmetic operations. An approximate homomorphic encryption scheme enables the execution of approximate additions and multiplications on ciphertexts without requiring decryption. It uses an isomorphism between complex vectors and the plaintext space $\mathcal{R}_q{=}\mathbb{Z}_q[X]/\langle X^N{+}1\rangle$, where q is a large modulus, and N is a power-of-two integer. The decryption of a ciphertext yields the input plaintext in $\mathcal{R}_q$ with a small error. This small error can be seen as an approximation in fixed-point arithmetic.

In CKKS, given a ring isomorphism between $\mathbb{C}^{N/2}$ and $\mathbb{R}[X]/\langle X^N+1\rangle$, a complex vector $\mathbf{z}\in\mathbb{C}^{N/2}$ can be encoded into a polynomial $\mathbf{p}$ denoted by a vector $\vec{\mathbf{p}}$ of its coefficients $\in\mathbb{R}^N$ as

$$\vec{\mathbf{p}}=\frac{1}{N}(\bar{\mathbf{U}}^T\cdot\mathbf{z}+\mathbf{U}^T\cdot\bar{\mathbf{z}}),$$

where $\mathbf{U}$ denotes a $(N/2)\times N$ portion of the Vandermonde matrix generated by the $2N$-th root of unity $\zeta_j=e^{5^j\pi i/N}$. This transformation is extended to $\mathcal{R}_q$ by a quantization. Then, considering a maximum number of levels L, a ring modulus $q=\prod_{i=0}^{L-1}q_i$ is chosen with $\{q_i\}$ a set of number theoretic transform (NTT)-friendly primes such that $\forall i\in[0,L-1]$, $q_i=1 \mod 2N$.

Let $\chi_{\text{err}}, \chi_{\text{enc}}$, and χ_{key}, be three sets of small distributions over $\mathcal{R}_q$. Then, for an encoded plaintext $\mathbf{p}\in\mathcal{R}_q$, the scheme works as follows:

KeyGen(λ, N, L, q): for a security parameter λ and a number of levels L, generate $\mathbf{sk}_{\text{HE}}=(1,\mathbf{s})$ with $\mathbf{s}\leftarrow\chi_{\text{key}}$, $\mathbf{pk}_{\text{HE}}=(\mathbf{b},\mathbf{a})$ with $\mathbf{a}\leftarrow\mathcal{R}_q$, $\mathbf{b}=-\mathbf{a}\cdot\mathbf{s}+\mathbf{e} \mod q$, and $\mathbf{e}\leftarrow\chi_{\text{err}}$. Additional keys which are useful for the homomorphic computations (*i.e.,* rotation, evaluation keys, etc.) are denoted by $\mathbf{evk}_{\text{HE}}$. We refer the reader to [HK20] for further details.

Encryption$(\mathbf{p};\mathbf{pk_{HE}})$: For a message $\mathbf{m}\in\mathbb{Z}_t^N$ encoded as a polynomial $\mathbf{p}$, for $\mathbf{r}_0\leftarrow\chi_{\text{enc}}$ and $\mathbf{e}_0,\mathbf{e}_1\leftarrow\chi_{\text{err}}$, output the pair of polynomials $\mathbf{ct}=(\mathbf{ct}_0,\mathbf{ct}_1)=\mathbf{r}_0\cdot\mathbf{pk}_{\text{HE}}+(\mathbf{p}+\mathbf{e}_0,\mathbf{e}_1) \mod q$.

Decryption$(\mathbf{ct};\mathbf{sk_{HE}})$: Output $\hat{\mathbf{m}}=\langle\mathbf{ct},\mathbf{sk}_{\text{HE}}\rangle \mod q_l$, where $\langle\cdot,\cdot\rangle$ denotes the canonical scalar product in $\mathcal{R}_{q_l}$ and l the current level of the ciphertext.

For brevity, we denote the above three operations as HE.KeyGen(λ, N, L, q), HE.Enc$(\mathbf{p};\mathbf{pk}_{\text{HE}})$, and HE.Dec$(\mathbf{ct};\mathbf{sk}_{\text{HE}})$, respectively. The scheme's parameters are chosen according to the security level required (see [ACC+18]) to protect the inputs and *privacy.*

Bibliography

[23a19] 23andMe. DNA Genetic Testing & Analysis, 2019. https://www.23andme.com/en-int/.

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. https://doi.org/10.1038/s41586-019-1666-5.

[ABC+15] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. *Journal of Cryptology*, 28(2):351–395, 2015. https://doi.org/10.1007/s00145-014-9182-0.

[ABF+18] Toshinori Araki, Assi Barak, Jun Furukawa, Marcel Keller, Yehuda Lindell, Kazuma Ohara, and Hikaru Tsuchida. Generalizing the SPDZ Compiler For Other Protocols. In *ACM Conference on Computer and Communications Security (CCS)*, pages 880–895, 2018. https://doi.org/10.1145/3243734.3243854.

[ACC+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. Technical report, HomomorphicEncryption.org, Toronto, Canada, 2018. http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf.

[ADMC17] Muhammad Rizwan Asghar, György Dán, Daniele Miorandi, and Imrich Chlamtac. Smart meter data privacy: A survey. *IEEE Communications Surveys & Tutorials*, 19(4):2820–2835, 2017. https://doi.org/10.1109/COMST.2017.2720195.

[AFS18] Adi Akavia, Dan Feldman, and Hayim Shaul. Secure search on encrypted data via multi-ring sketch. In *ACM Conference on Computer and Communications Security (CCS)*, pages 985–1001, 2018. https://doi.org/10.1145/3243734.3243810.

[AGHV22] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable CCA2 relaxation for homomorphic encryption. In *Theory of Cryptography – TCC*, pages 70–99. Springer, 2022. https://doi.org/10.1007/978-3-031-22365-5_3.

[AGJS13] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for CPU based attestation and sealing. In *International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013. https://www.intel.com/content/dam/develop/external/us/en/documents/hasp-2013-innovative-technology-for-attestation-and-sealing-413939.pdf.

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

[AHHK18] Ulrich Matchi Aïvodji, Kévin Huguenin, Marie-José Huguet, and Marc-Olivier Killijian. Sride: A privacy-preserving ridesharing system. In *ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, pages 40–50, 2018. https://dl.acm.org/doi/10.1145/3212480.3212483.

[AHWC19] Asma Aloufi, Peizhao Hu, Harry WH Wong, and Sherman SM Chow. Blindfolded evaluation of random forests with multi-key homomorphic encryption. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2019. https://doi.org/10.1109/TDSC.2019.2940020.

[AJLA+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*, pages 483–501. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_29.

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

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography – TCC*, pages 137–156. Springer, 2007. https://doi.org/10.1007/978-3-540-70936-7_8.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *Advances in Cryptology – CRYPTO*, pages 470–499. Springer, 2020. https://doi.org/10.1007/978-3-030-56880-1_17.

[Ama21] Amazon Science. Machine learning models that act on encrypted data. Online: https://www.amazon.science/blog/machine-learning-models-that-act-on-encrypted-data, October 2021.

[AMBG+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*, 2016. https://doi.org/10.1007/978-3-319-29485-8_20.

[ARHR13] Erman Ayday, Jean Louis Raisaro, Jean-Pierre Hubaux, and Jacques Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *ACM Workshop on privacy in the electronic society (WPES)*, pages 95–106, 2013. https://dl.acm.org/doi/10.1145/2517840.2517843.

[AS16] Asmaa Abdallah and Xuemin Sherman Shen. A Lightweight Lattice-Based Homomorphic Privacy-Preserving Data Aggregation Scheme for Smart Grid. *IEEE Transactions on Smart Grid*, 9(1):396–405, 2016. https://doi.org/10.1109/TSG.2016.2553647.

[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)*, pages 4901–4906. IEEE, 2021. https://doi.org/10.1109/CDC45484.2021.9683401.

[BBB+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)*, pages 315–334. IEEE, 2018. https://doi.org/10.1109/SP.2018.00020.

[BBFR15] 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)*, pages 271–286. IEEE, 2015. https://doi.org/10.1109/SP.2015.24.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of computer and system sciences*, 37(2):156–189, 1988. https://doi.org/10.1016/0022-0000(88)90005-0.

[BCC+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*, pages 327–357. Springer, 2016. https://doi.org/10.1007/978-3-662-49896-5_12.

[BCD+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)*, pages 325–343. Springer, 2009. https://doi.org/10.1007/978-3-642-03549-4_20.

[BCFK21] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In *Public-Key Cryptography – PKC*, pages 528–558. Springer, 2021. https://doi.org/10.1007/978-3-030-75248-4_19.

[BCG+23] Mariya Georgieva Belorgey, Sergiu Carpov, Nicolas Gama, Sandra Guasch, and Dimitar Jetchev. Revisiting key decomposition techniques for FHE: Simpler, faster and more generic. *Cryptology ePrint Archive*, 2023. https://ia.cr/2023/771.

[BCIV17] Joppe W Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Privacy-friendly forecasting for the smart grid using homomorphic encryption and the group method of data handling. In *Progress in Cryptology – AFRICACRYPT*, 2017. https://doi.org/10.1007/978-3-319-57339-7_11.

[BCOS20] 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*, pages 247–267. Springer, 2020. https://doi.org/10.1007/978-3-030-44223-1_14.

[BCY91] Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, 1991. https://doi.org/10.1016/0304-3975(91)90259-5.

[BD16] Carsten Baum and Ivan Damgard. Efficient Commitments and Zero-Knowledge Protocols from Ring-SIS with Applications to Lattice-based Threshold Cryptosystems. *Cryptology ePrint Report 2016/997*, 2016. https://eprint.iacr.org/archive/2016/997/20161020:181306.

[BDL+18a] 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)*, pages 368–385. Springer, 2018. https://doi.org/10.1007/978-3-319-98113-0_20.

[BDL+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*, pages 368–385. Springer, 2018. https://doi.org/10.1007/978-3-319-98113-0_20.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT*, pages 169–188. Springer, 2011. https://doi.org/10.1007/978-3-642-20465-4_11.

[BDS+21] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty:

Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021. https://arxiv.org/abs/2112.02918.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology – CRYPTO*, volume 576, pages 420–432. Springer, 1991. https://dx.doi.org/10.1007/3-540-46766-1_34.

[BEHZ17] 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*, pages 423–442. Springer, 2017. https://doi.org/10.1007/978-3-319-69453-5_23.

[Ben86] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret sharing. In *Advances in Cryptolog – CRYPTO*, volume 263, pages 251–260. Springer, 1986. https://doi.org/10.1007/3-540-47721-7_19.

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

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology – EUROCRYPT*, pages 149–168. Springer, 2011. https://doi.org/10.1007/978-3-642-20465-4_10.

[BFH+20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: a new optimized sublinear iop. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2025–2038, 2020. https://doi.org/10.1145/3372297.3417893.

[BFR13] Michael Backes, Dario Fiore, and Raphael M Reischuk. Verifiable delegation of computation on outsourced data. In *ACM SIGSAC conference on Computer & Communications Security (CCS)*, pages 863–874, 2013. https://dl.acm.org/doi/10.1145/2508859.2516681.

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

[BGBE19] Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In *International Conference on Machine Learning (ICML)*, pages 812–821. PMLR, 2019. http://proceedings.mlr.press/v97/brutzkus19a.

[BGG+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*, pages 565–596. Springer, 2018. https://doi.org/10.1007/978-3-319-96884-1_19.

[BGH+13] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J Wu. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security (ACNS)*, pages 102–118. Springer, 2013. https://doi.org/10.1007/978-3-642-38980-1_7.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Advances in Cryptology – CRYPTO*, pages 111–131. Springer, 2011. https://doi.org/10.1007/978-3-642-22792-9_7.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. https://dl.acm.org/doi/10.1145/2090236.2090262.

[BH20] Eszter Bokányi and Anikó Hannák. Understanding inequalities in ride-hailing services through simulations. *Scientific reports*, 10(1):6500, 2020. https://www.nature.com/articles/s41598-020-63171-9.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2129–2146, 2019. https://doi.org/10.1145/3319535.3363229.

[BJSV15] 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)*, pages 227–234. Springer, 2015. https://doi.org/10.1007/978-3-662-47854-7_14.

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

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In *Advances in Cryptology – CRYPTO*, pages 441–469. Springer, 2020. https://doi.org/10.1007/978-3-030-56880-1_16.

[BLNS21] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. More efficient amortization of exact zero-knowledge proofs for LWE. In *Computer Security – ESORICS*, pages 608–627. Springer, 2021. https://doi.org/10.1007/978-3-030-88428-4_30.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *Advances in Cryptology – CRYPTO*, pages 176–202. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_7.

[BMC+19] Annalisa Buniello, Jacqueline A L MacArthur, Maria Cerezo, et al. The NHGRI-EBI GWAS catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic acids research*, 2019. https://www.ebi.ac.uk/gwas/home.

[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*, pages 495–526. Springer, 2020. https://doi.org/10.1007/978-3-030-45374-9_17.

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

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In *Advances in Cryptology – CRYPTO*, pages 190–213. Springer, 2016. https://doi.org/10.1007/978-3-662-53018-4_8.

[BR18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018. https://doi.org/10.1145/3243734.3264418.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO*. Springer, 2012. https://doi.org/10.1007/978-3-642-32009-5_50.

[BS21] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security Symposium*, 2021. https://www.usenix.org/system/files/sec21-bagdasaryan.pdf.

[BSBHR18] 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)*, 2018. http://drops.dagstuhl.de/opus/volltexte/2018/9018.

[BSBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable Zero Knowledge with No Trusted Setup. In *Advances in Cryptology – CRYPTO*, pages 701–732. Springer, 2019. https://doi.org/10.1007/978-3-030-26954-8_23.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology – CRYPTO*, pages 90–108. Springer, 2013. https://doi.org/10.1007/978-3-642-40084-1_6.

[BSCR+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*, pages 103–128. Springer, 2019. https://doi.org/10.1007/978-3-030-17653-2_4.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a Von Neumann architecture. In *USENIX Security Symposium*, pages 781–796, 2014. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson.

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

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2011. https://doi.ieeecomputersociety.org/10.1109/FOCS.2011.12.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology – CRYPTO*, pages 505–524. Springer, 2011. https://doi.org/10.1007/978-3-642-22792-9_29.

[BVH+18] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018. https://arxiv.org/abs/1807.00459.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In *Advances in Cryptology – ASIACRYPT*, pages 446–472. Springer, 2019. https://doi.org/10.1007/978-3-030-34621-8_16.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM SIGSAC conference on Computer and Communications Security (CCS)*, pages 1825–1842, 2017. https://doi.org/10.1145/3133956.3133997.

[CDKS19] 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)*, pages 395–412, 2019. https://doi.org/10.1145/3319535.3363207.

[CDSG+21] Seung Geol Choi, Dana Dachman-Soled, S Dov Gordon, Linsheng Liu, and Arkady Yerukhimovich. Compressed oblivious encoding for homomorphically

encrypted search. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2277–2291, 2021. https://doi.org/10.1145/3460120.3484792.

[CEBC22] Jeffrey Chen, Manaswitha Edupalli, Bonnie Berger, and Hyunghoon Cho. Secure and federated linear mixed model association tests. *bioRxiv*, 2022. https://doi.org/10.1101/2022.05.20.492837.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic macs for arithmetic circuits. In *Advances in Cryptology – EUROCRYPT*, pages 336–352, 2013. https://doi.org/10.1007/978-3-642-38348-9_21.

[CFC+22] 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. *bioRxiv*, 2022.

[CFGN14] Dario Catalano, Dario Fiore, Rosario Gennaro, and Luca Nizzardo. Generalizing homomorphic macs for arithmetic circuits. In *Public-Key Cryptography – PKC*, pages 538–555. Springer, 2014. https://doi.org/10.1007/978-3-642-54631-0_31.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 253–270. IEEE, 2015. https://doi.org/10.1109/SP.2015.23.

[CFN18] Dario Catalano, Dario Fiore, and Luca Nizzardo. On the security notions for homomorphic signatures. In *Applied Cryptography and Network Security (ACNS)*, pages 183–201. Springer, 2018. https://doi.org/10.1007/978-3-319-93387-0_10.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *Advances in Cryptology – CRYPTO*, pages 371–389. Springer, 2014. https://doi.org/10.1007/978-3-662-44371-2_21.

[CGBH+18] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11:3–12, 2018. https://doi.org/10.1186/s12920-018-0397-z.

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

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds.

In *Advances in Cryptology – ASIACRYPT*, 2016. https://doi.org/10.1007/978-3-662-53887-6_1.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 2020. https://doi.org/10.1007/s00145-019-09319-x.

[CGHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In *Advances in Cryptology – EUROCRYPT*, pages 3–33. Springer, 2022. https://doi.org/10.1007/978-3-031-07085-3_1.

[Cha84] David Chaum. Blind signature system. In *Advances in Cryptology – CRYPTO*, pages 153–153. Springer, 1984. https://doi.org/10.1007/978-1-4684-4730-9_14.

[CHH+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. *IEEE access*, 6:24325–24339, 2018. https://ieeexplore.ieee.org/abstract/document/8325268.

[CHK+18a] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 360–384. Springer, 2018. https://doi.org/10.1007/978-3-319-78381-9_14.

[CHK+18b] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography – SAC*, pages 347–368. Springer, 2018. https://doi.org/10.1007/978-3-030-10970-7_16.

[CHK+19] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography – SAC*. Springer, 2019. https://doi.org/10.1007/978-3-030-10970-7_16.

[CJLL17] Jung Hee Cheon, Jinhyuck Jeong, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In *Financial Cryptography and Data Security (FC)*, pages 53–74. Springer, 2017. https://doi.org/10.1007/978-3-319-70278-0_4.

[CKK+19] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In *Advances in Cryptology – ASIACRYPT*, pages 415–445. Springer, 2019. https://doi.org/10.1007/978-3-030-34621-8_15.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT*, pages 409–437. Springer, 2017. https://doi.org/10.1007/978-3-319-70694-8_15.

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *Advances in Cryptology – EUROCRYPT*, 2021. https://doi.org/10.1007/978-3-030-77883-5_9.

[CKP+23] Sylvain Chatel, Christian Knabenhans, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. Verifiable encodings for secure homomorphic analytics. *arXiv preprint arXiv:2207.14071*, 2023. https://arxiv.org/abs/2207.14071.

[CKR+20] Hao Chen, Miran Kim, Ilya Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously secure matrix multiplication with applications to private deep learning. In *Advances in Cryptology – ASIACRYPT*, pages 31–59. Springer, 2020. https://doi.org/10.1007/978-3-030-64840-4_2.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology – CRYPTO*, pages 483–501. Springer, 2010. https://doi.org/10.1007/978-3-642-14623-7_26.

[CLL22] Wei Chen, Lu Liu, and Guo-Ping Liu. Privacy-preserving distributed economic dispatch of microgrids: A dynamic quantization-based consensus scheme with homomorphic encryption. *IEEE Transactions on Smart Grid*, 2022. https://doi.org/10.1109/TSG.2022.3189665.

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

[CMS+23] Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. PELTA–shielding Multiparty-FHE against malicious adversaries. *Cryptology ePrint Archive*, 2023. https://ia.cr/2023/642.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *Advances in Cryptology – EUROCRYPT*, pages 769–793. Springer, 2020. https://dx.doi.org/10.1007/978-3-030-45721-1_27.

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

[CPTPH21a] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Privacy and integrity preserving computations with CRISP. In *USENIX Security Symposium*, pages 2111–2128, 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/chatel.

[CPTPH21b] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Sok: Privacy-preserving collaborative tree-based model learning. *Proceedings on Privacy Enhancing Technologies*, 2021. https://petsymposium.org/2021/files/papers/issue3/popets-2021-0043.pdf.

[CPW10] Ann Cavoukian, Jules Polonetsky, and Christopher Wolf. Smartprivacy for the smart grid: embedding privacy into the design of electricity conservation. *Identity in the Information Society*, 3:275–294, 2010. https://doi.org/10.1007/s12394-010-0046-y.

[CSC+23] Weikeng Chen, Katerina Sotiraki, Ian Chang, Murat Kantarcioglu, and Raluca Ada Popa. HOLMES: a platform for detecting malicious inputs in secure collaborative computation. In *USENIX Security Symposium*, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/chang.

[CSS+22] 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, 2022. https://eprint.iacr.org/2022/1625.

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

[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)*, pages 597–627. Springer, 2017. https://doi.org/10.1007/978-3-319-70503-3_20.

[Dan15] Quynh H. Dang. Secure Hash Standard. Technical report, National Institute of Standards and Technology, July 2015. https://doi.org/10.6028/NIST.FIPS.180-4.

[DBN+01] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (AES), 2001-11-26 2001. https://doi.org/10.6028/NIST.FIPS.197.

[DCFT13] Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. Secure genomic testing with size- and position-hiding private substring matching. In *ACM Workshop*

on privacy in the electronic society (WPES), pages 107–118, 2013. https://dl.acm.org/doi/10.1145/2517840.2517849.

[DDC14] George Danezis and Emiliano De Cristofaro. Fast and private genomic testing for disease susceptibility. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 31–34, 2014. https://dl.acm.org/doi/10.1145/2665943.2665952.

[DK21] Shlomi Dolev and Arseni Kalma. Verifiable computing using computation fingerprints within fhe. In *IEEE International Symposium on Network Computing and Applications (NCA)*, pages 1–9. IEEE, 2021. https://doi.org/10.1109/NCA53618.2021.9685831.

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

[DL21] Oualid Demigha and Ramzi Larguet. Hardware-based solutions for trusted cloud computing. *Computers & Security*, 103:102117, 2021. https://doi.org/10.1016/j.cose.2020.102117.

[DNA19] DNAfit | US. Accurate DNA Test For Diet, Fitness, Health & Wellness, 2019. https://www.dnafit.com/us/.

[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)*, pages 574–591, 2018. https://doi.org/10.1145/3243734.3243852.

[DPLS19] Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for FHE and ring-LWE ciphertexts. In *Public-Key Cryptography – PKC*, 2019. https://doi.org/10.1007/978-3-030-17253-4_12.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO*, pages 643–662. Springer, 2012. https://doi.org/10.1007/978-3-642-32009-5_38.

[Dwo06] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, 2006. https://doi.org/10.1007/11787006_1.

[EKR+18] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3):70–246, 2018. https://www.cs.virginia.edu/~evans/pragmaticmpc/pragmaticmpc.pdf.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 1985. https://doi.org/10.1109/TIT.1985.1057074.

[EN14] Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014. https://www.nature.com/articles/nrg3723.

[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*, pages 259–288. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_9.

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

[ESLL19a] 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*, pages 115–146. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_5.

[ESLL19b] 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*, pages 115–146. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_5.

[ESS+19] 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)*, pages 67–88. Springer, 2019. https://doi.org/10.1007/978-3-030-21568-2_4.

[ETPLPG13] Zekeriya Erkin, Juan Ramón Troncoso-Pastoriza, Reginald L Lagendijk, and Fernando Pérez-González. Privacy-preserving data aggregation in smart metering systems: An overview. *IEEE Signal Processing Magazine*, 30(2):75–86, 2013. https://doi.org/10.1109/MSP.2012.2228343.

[FG12] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM conference on Computer and Communications Security (CCS)*, pages 501–512, 2012. https://doi.org/10.1145/2382196.2382250.

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

[FKDL13] Cedric Fournet, Markulf Kohlweiss, George Danezis, and Zhengqin Luo. ZQL: A compiler for privacy-preserving data processing. In *USENIX Security Symposium*, pages 163–178, 2013. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/fournet.

[FKSW19] Susanne Felsen, Ágnes Kiss, Thomas Schneider, and Christian Weinert. Secure and private function evaluation with intel SGX. In *ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW)*, pages 165–181, 2019. https://doi.org/10.1145/3338466.3358919.

[FL14] Matthew Fredrikson and Benjamin Livshits. ZØ: An optimizing distributing Zero-Knowledge compiler. In *USENIX Security Symposium*, pages 909–924, 2014. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson.

[FMM+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. SAFELearn: secure aggregation for private federated learning. In *IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021. https://doi.org/10.1109/SPW53761.2021.00017.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *Advances in Cryptology – ASIACRYPT*, pages 499–530. Springer, 2016. https://doi.org/10.1007/978-3-662-53890-6_17.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In *Public-Key Cryptography – PKC*, pages 124–154. Springer, 2020. https://doi.org/10.1007/978-3-030-45388-6_5.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO*, volume 263, pages 186–194. Springer, 1986. https://doi.org/10.1007/3-540-47721-7_12.

[FTPP+21] 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. *Proceedings on Privacy Enhancing Technologies*, 2021(2):323–347, 2021. https://doi.org/10.2478/popets-2021-0030.

[FTPR+21] 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. *Nature communications*, 12(1):1–10, 2021. https://doi.org/10.1038/s41467-021-25972-y.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012. https://eprint.iacr.org/2012/144.

[FV23] Ramsès Fernàndez-València. Verifiable encodings in multigroup fully homomorphic encryption. *arXiv preprint arXiv:2303.08432*, 2023. https://arxiv.org/abs/2303.08432.

[FYDX21] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security vulnerabilities of SGX and countermeasures: A survey. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021. https://doi.org/10.1145/3456631.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009. https://crypto.stanford.edu/craig/craig-thesis.pdf.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009. https://doi.org/10.1145/1536414.1536440.

[GFS+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*, 2012. https://doi.org/10.1007/978-3-642-33027-8_30.

[GGG17] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud. *Advances in Neural Information Processing Systems (NIPS)*, 30, 2017. https://dl.acm.org/doi/10.5555/3294996.3295220.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology – CRYPTO*, pages 465–482. Springer, 2010. https://doi.org/10.1007/978-3-642-14623-7_25.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology – EUROCRYPT*, pages 626–645. Springer, 2013. https://doi.org/10.1007/978-3-642-38348-9_37.

[GH11] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology – EUROCRYPT*, volume 6632, pages 129–148. Springer, 2011. https://doi.org/10.1007/978-3-642-20465-4_9.

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P Smart. Ring switching in BGV-style homomorphic encryption. In *Security and Cryptography for Networks: 8th International Conference (SCN)*, pages 19–37. Springer, 2012. https://doi.org/10.1007/978-3-642-32928-9_2.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography – PKC*, pages 1–16. Springer, 2012. https://doi.org/10.1007/978-3-642-30057-8_1.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology – EUROCRYPT*, volume 7237, pages 465–482. Springer, 2012. https://doi.org/10.1007/978-3-642-29011-4_28.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology – CRYPTO*, pages 850–867. Springer, 2012. https://doi.org/10.1007/978-3-642-32009-5_49.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In *Advances in Cryptology – CRYPTO*, pages 698–728. Springer, 2018. https://doi.org/10.1007/978-3-319-96878-0_24.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology – CRYPTO*, pages 536–553. Springer, 2013. https://doi.org/10.1007/978-3-642-40084-1_30.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015. https://dl.acm.org/doi/pdf/10.1145/2699436.

[GLL+20] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. VeriFL: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security*, 16:1736–1751, 2020. https://doi.org/10.1109/TIFS.2020.3043139.

[GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21, 2012. https://doi.org/10.1007/978-3-642-37682-5_1.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. In *USENIX Security Symposium*, volume 16, 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1985. https://doi.org/10.1145/22145.22178.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. Comput.*, 18(1):186–208, 1989. https://doi.org/10.1137/0218012.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1987. https://doi.org/10.1145/28395.28420.

[GNSV21] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: Snarks for ring arithmetic. *Cryptology ePrint Archive, Report 2021/322*, 2021. https://ia.cr/2021/322.

[Goo18] Google. Google cloud encrypted bigquery client. Online: https://github.com/google/encrypted-bigquery-client, February 2018.

[Goo20] Google Cloud. Encryption at rest in google cloud. Online: https://cloud.google.com/security/encryption/default-encryption, July 2020.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT*, volume 6477, pages 321–340. Springer, 2010. https://doi.org/10.1007/978-3-642-17373-8_19.

[Gro11] Jens Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *Advances in Cryptology – ASIACRYPT*, volume 7073, pages 431–448. Springer, 2011. https://doi.org/10.1007/978-3-642-25385-0_23.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT*, pages 305–326. Springer, 2016. https://doi.org/10.1007/978-3-662-49896-5_11.

[GVBP+22] Robin Geelen, Michiel Van Beirendonck, Hilder VL Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios Dimou, Ingrid Verbauwhede, Frederik Vercauteren, et al. Basalisc: Programmable asynchronous hardware accelerator for bgv fully homomorphic encryption. *Cryptology ePrint Archive*, 2022. https://ia.cr/2022/657.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 469–477, 2015. https://doi.org/10.1145/2746539.2746576.

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In *Advances in Cryptology-ASIACRYPT*, pages 301–320. Springer, 2013. https://doi.org/10.1007/978-3-642-42045-0_16.

[HB11] Ragib Hasan and Randal C. Burns. Where have you been? Secure Location Provenance for Mobile Devices. *CoRR*, 2011. http://arxiv.org/abs/1107.1821.

[Her18] Alex Hern. Fitness tracking app Strava gives away location of secret US army bases. *The Guardian*, 2018. https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases.

[HK20] Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In *Topics in Cryptology–CT-RSA*, pages 364–390. Springer, 2020. https://doi.org/10.1007/978-3-030-40186-3_16.

[HKKH21] Changhee Hahn, Hodong Kim, Minjae Kim, and Junbeom Hur. Versa: Verifiable secure aggregation for cross-device federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2021. https://doi.org/10.1109/TDSC.2021.3126323.

[HLP+13] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. *HASP ISCA*, 11(10.1145):2487726–2488370, 2013. https://doi.org/10.1145/2487726.2488370.

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

[HTGW18] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018. https://doi.org/10.1515/popets-2018-0024.

[HWA21] Hanieh Hashemi, Yongqin Wang, and Murali Annavaram. DarKnight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 212–224, 2021. https://doi.org/10.1145/3466752.3480112.

[IBM21a] IBM. Helib v2.2.1. Online: https://github.com/homenc/HElib, October 2021.

[IBM21b] IBM Security. IBM security homomorphic encryption services. Online: https://www.ibm.com/security/services/homomorphic-encryption, October 2021.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing*, January 2009. https://doi.org/10.1137/080725398.

[Int19a] Garmin International. Garmin Connect | Free Online Fitness Community, 2019. https://connect.garmin.com/.

[Int19b] International Genome. 1000 Genomes | A Deep Catalog of Human Genetic Variation, 2019. https://www.internationalgenome.org/.

[JMSW02] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Topics in Cryptology – CT-RSA*, pages 244–262. Springer, 2002. https://doi.org/10.1007/3-540-45760-7_17.

[JOB+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)*, 2018. https://doi.org/10.1109/SP.2018.00057.

[JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, pages 1651–1669, 2018. https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

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

[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*, pages 173–192. Springer, 2014. https://doi.org/10.1007/978-3-662-45608-8_10.

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

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 525–537, 2018. https://doi.org/10.1145/3243734.3243805.

[KL15] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. In *BMC medical informatics and decision making*, pages 1–12. BioMed Central, 2015. https://doi.org/10.1186/1472-6947-15-S5-S3.

[KLB+19] Pardeep Kumar, Yun Lin, Guangdong Bai, Andrew Paverd, Jin Song Dong, and Andrew Martin. Smart grid metering networks: A survey on security, privacy and open research issues. *IEEE Communications Surveys & Tutorials*, 21(3):2886–2927, 2019. https://doi.org/10.1109/COMST.2019.2899354.

[KLSW21] Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. A unified framework of homomorphic encryption for multiple parties with non-interactive setup. *Cryptology ePrint Archive*, 2021. https://eprint.iacr.org/2021/1412.

[KMRR16] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016. https://arxiv.org/abs/1610.02527.

[KPS18] Ahmed Kosba, Charalampos Papamanthou, and Elaine Shi. xJsnark: A framework for efficient verifiable computation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 944–961. IEEE, 2018. https://doi.org/10.1109/SP.2018.00018.

[KPZ21] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields. In *Advances in Cryptology – ASIACRYPT*, pages 608–639. Springer, 2021. https://doi.org/10.1007/978-3-030-92078-4_21.

[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*, volume 5350, pages 372–389. Springer, 2008. https://doi.org/10.1007/978-3-540-89255-7_23.

[KZG10] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology – ASIACRYPT*, pages 177–194. Springer, 2010. https://doi.org/10.1007/978-3-642-17373-8_11.

[Lai17] Kim Laine. Simple encrypted arithmetic library 2.3. 1. *Microsoft Research*, 2017. https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf.

[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)*, pages 1219–1234, 2012. https://doi.org/10.1145/2213977.2214086.

[LDPW14] Junzuo Lai, Robert H Deng, HweeHwa Pang, and Jian Weng. Verifiable computation on outsourced encrypted data. In *Computer Security – ESORICS*, pages 273–291. Springer, 2014. https://doi.org/10.1007/978-3-319-11203-9_16.

[LDS20] LDS. CRISP's Implementation, 2020. https://github.com/ldsec/CRISP.

[LDS21] LDS. VERITAS's Implementation, 2021. https://github.com/ldsec/veritas.

[LDS23] LDS. PELTA's Implementation, 2023. https://github.com/ldsec/pelta.

[LGC+20] Meng Li, Jianbo Gao, Yifei Chen, Jingcheng Zhao, and Mamoun Alazab. Privacy-preserving ride-hailing with verifiable order-linking in vehicular networks. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 599–606. IEEE, 2020. https://doi.org/10.1109/TrustCom50675.2020.00085.

[LH10] Wanying Luo and Urs Hengartner. VeriPlace: A Privacy-aware Location Proof Architecture. In *SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 23–32, 2010. https://doi.org/10.1145/1869790.1869797.

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. *Advances in Cryptology – CRYPTO*, pages 1–17, 2013. https://doi.org/10.1007/978-3-642-40084-1_1.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography – TCC*, pages 169–189. Springer, 2012. https://doi.org/10.1007/978-3-642-28914-9_10.

[Lip19] Robert Lipe. GPSBabel: convert, upload, download data from GPS and Map programs v1.6, 2019. https://www.gpsbabel.org/index.html.

[LKO+21] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. PLATYPUS: Software-based power side-channel attacks on X86. In *IEEE Symposium on Security and Privacy (S&P)*, pages 355–371. IEEE, 2021. https://doi.org/10.1109/SP40001.2021.00063.

[LKS17] Wenjie Lu, Shohei Kawasaki, and Jun Sakuma. Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. *Annual Network And Distributed System Security Symposium (NDSS)*, 2017. http://dx.doi.org/10.14722/ndss.2017.23119.

[LL12] Fengjun Li and Bo Luo. Preserving data integrity for smart grid data aggregation. In *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 366–371. IEEE, 2012. https://doi.org/10.1109/SmartGridComm.2012.6486011.

[LLL10] Fengjun Li, Bo Luo, and Peng Liu. Secure information aggregation for smart grids using homomorphic encryption. In *IEEE international conference on smart grid communications*. IEEE, 2010. https://doi.org/10.1109/SMARTGRID.2010.5622064.

[LLNW18] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *Advances in Cryptology – CRYPTO*, pages 700–732. Springer, 2018. https://doi.org/10.1007/978-3-319-96881-0_24.

[LMN16] Atul Luykx, Bart Mennink, and Samuel Neves. Security analysis of BLAKE2's modes of operation. *IACR Transactions on Symmetric Cryptology*, 2016. https://ia.cr/2016/827.

[LMSS22] Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. Securing approximate homomorphic encryption using differential privacy. In *Advances in Cryptology – CRYPTO*, 2022. https://doi.org/10.1007/978-3-031-15802-5_20.

[LN17] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *Advances in Cryptology – EUROCRYPT*, pages 293–323. Springer, 2017. https://doi.org/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*, pages 71–101. Springer, 2022. https://doi.org/10.1007/978-3-031-15979-4_3.

[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)*, pages 1051–1070, 2020. https://doi.org/10.1145/3372297.3417894.

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

[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*, pages 611–640. Springer, 2021. https://doi.org/10.1007/978-3-030-84245-1_21.

[LNSW13] 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*, pages 107–124. Springer, 2013. https://doi.org/10.1007/978-3-642-36362-7_8.

[LPJY13] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. *Advances in Cryptology – CRYPTO*, 2013. https://doi.org/10.1007/978-3-642-40084-1_17.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology – EUROCRYPT*, 2010. https://doi.org/10.1007/978-3-642-13190-5_1.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. https://doi.org/10.1007/s10623-014-9938-4.

[LWX22] Shimin Li, Xin Wang, and Rui Xue. Toward both privacy and efficiency of homomorphic MACs for polynomial functions and its applications. *The*

Computer Journal, 65(4):1020–1028, 2022. https://doi.org/10.1093/comjnl/bxab042.

[LWZ18] Shimin Li, Xin Wang, and Rui Zhang. Privacy-preserving homomorphic MACs with efficient verification. In *Web Services–ICWS*, pages 100–115. Springer, 2018. https://doi.org/10.1007/978-3-319-94289-6_7.

[LYS15] Wen-Jie Lu, Yoshiji Yamada, and Jun Sakuma. Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. In *BMC medical informatics and decision making*, volume 15, pages 1–8. Springer, 2015. https://doi.org/10.1186/1472-6947-15-S5-S1.

[LZY+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. *IEEE Access*, 2019. https://doi.org/10.1109/ACCESS.2019.2913943.

[MAB+13] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. *HASP ISCA*, 10(1), 2013. https://doi.org/10.1145/2487726.2488368.

[MBH22] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for RLWE-based multiparty homomorphic encryption. *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/780.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2111–2128, 2019. https://dl.acm.org/doi/10.1145/3319535.3339817.

[MCP+23] Christian Mouchet, Sylvain Chatel, Apostolos Pyrgelis, Jean-Pierre Hubaux, and Carmela Troncoso. Helium: a multiparty FHE-based system for MPC under churn. *preprint*, 2023. to appear.

[Mic21] Microsoft Azure. Homomorphic encryption with SEAL. Online: https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/homomorphic-encryption-seal, October 2021.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics (AISTATS)*, pages 1273–1282. PMLR, 2017. http://proceedings.mlr.press/v54/mcmahan17a.html.

[Mon85] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 1985. https://doi.org/10.1090/S0025-5718-1985-0777282-X.

[MPC+18] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *IEEE Symposium on Security and Privacy (S&P)*, pages 279–296. IEEE, 2018. https://doi.org/10.1109/SP.2018.00045.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 691–706. IEEE, 2019. https://doi.org/10.1109/SP.2019.00029.

[MSM+22] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank HP Fitzek, and Najwa Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 2022. https://doi.org/10.1109/JPROC.2022.3205665.

[MSS20] Abbass Madi, Renaud Sirdey, and Oana Stan. Computing neural networks with homomorphic encryption and verifiable computing. In *Applied Cryptography and Network Security Workshops (ACNS)*, pages 295–317. Springer, 2020. https://doi.org/10.1007/978-3-030-61638-0_17.

[MTPBH21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies*, 2021:291–311, 2021. https://doi.org/10.2478/popets-2021-0071.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology – EUROCRYPT*, pages 735–763. Springer, 2016. https://doi.org/10.1007/978-3-662-49896-5_26.

[MW22] Samir Jordan Menon and David J Wu. Spiral: Fast, high-rate single-server PIR via FHE composition. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022. https://doi.org/10.1109/SP46214.2022.9833700.

[NIS15] NIST. Special publication 800-78-4: Cryptographic algorithms and key sizes for personal identity verification, 2015. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf.

[NLDD21] 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. *Cryptology ePrint Archive*, 2021. https://ia.cr/2021/1603.

[NPR19] Moni Naor, Benny Pinkas, and Eyal Ronen. How to (not) share a password: Privacy preserving protocols for finding heavy hitters with adversarial behavior. In *ACM SIGSAC conference on Computer and Communications Security (CCS)*, pages 1369–1386, 2019. https://doi.org/10.1145/3319535.3363204.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 129–139, 1999. https://doi.org/10.1145/336992.337028.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE symposium on security and privacy (S&P)*, pages 739–753. IEEE, 2019. https://doi.org/10.1109/SP.2019.00065.

[NWT+20] Chaoyue Niu, Fan Wu, Shaojie Tang, Shuai Ma, and Guihai Chen. Toward verifiable and privacy preserving machine learning prediction. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1703–1721, 2020. https://doi.org/10.1109/TDSC.2020.3035591.

[Ogi23] Tabitha Ogilvie. Differential privacy for free? harnessing the noise in approximate homomorphic encryption. *Cryptology ePrint Archive*, 2023. https://ia.cr/2023/701.

[PAD+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 *USENIX Security Symposium*, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/polychroniadou.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT*, pages 223–238. Springer, 1999. https://doi.org/10.1007/3-540-48910-X_16.

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

[PDE+17] Anh Pham, Italo Dacosta, Guillaume Endignoux, Juan Ramon Troncoso Pastoriza, Kévin Huguenin, and Jean-Pierre Hubaux. ORide: A privacy-preserving yet accountable ride-hailing service. In *USENIX Security Symposium*, pages 1235–1252, 2017. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/pham.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In *International workshop on post-quantum cryptography – PQCrypto*. Springer, 2014. https://doi.org/10.1007/978-3-319-11659-4_12.

[PFA22] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2429–2443, 2022. https://doi.org/10.1145/3548606.3560557.

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

[PHB+15] Anh Pham, Kévin Huguenin, Igor Bilogrevic, Italo Dacosta, and Jean-Pierre Hubaux. SecureRun: Cheat-proof and private summaries for location-based activities. *IEEE Transactions on Mobile Computing*, 15(8):2109–2123, 2015. https://doi.org/10.1109/TMC.2015.2483498.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 238–252, 2013. https://doi.org/10.1109/SP.2013.47.

[PKY+21] Rishabh Poddar, Sukrit Kalra, Avishay Yanai, Ryan Deng, Raluca Ada Popa, and Joseph M Hellerstein. Senate: A maliciously-secure mpc platform for collaborative analytics. In *USENIX Security Symposium*, pages 2129–2146, 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/poddar.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography (TCC)*, pages 145–166. Springer, 2006. https://doi.org/10.1007/11681878_8.

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

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

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography – TCC*, volume 7785, pages 222–242. Springer, 2013. https://doi.org/10.1007/978-3-642-36594-2_13.

[PVC18] Christian Priebe, Kapil Vaswani, and Manuel Costa. EnclaveDB: A secure database using SGX. In *IEEE Symposium on Security and Privacy (S&P)*, pages 264–278. IEEE, 2018. https://doi.org/10.1109/SP.2018.00025.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978. https://people.csail.mit.edu/rivest/pubs/RAD78.pdf.

[Ran21] Jaak Randmets. An overview of vulnerabilities and mitigations of intel SGX applications. Technical Report D-2-116, Cybernetica AS, [Online], 2021.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the ACM Symposium on Theory of Computing*, STOC '05, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. https://dl.acm.org/doi/10.1145/1568318.1568324.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978. https://doi.org/10.1145/359340.359342.

[RSWP23] 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)*, pages 1573–1591, 2023. https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00090.

[RTPM+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. *IEEE/ACM transactions on computational biology and bioinformatics*, 2018. https://doi.org/10.1109/TCBB.2018.2854776.

[RVVV17] Sujoy Sinha Roy, Frederik Vercauteren, Jo Vliegen, and Ingrid Verbauwhede. Hardware assisted fully homomorphic function evaluation and encrypted search. *IEEE Transactions on Computers*, 66(9):1562–1572, 2017. https://doi.org/10.1109/TC.2017.2686385.

[SA15] Markku-Juhani O. Saarinen and Jean-Philippe Aumasson. The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). RFC 7693, 2015. https://www.rfc-editor.org/rfc/rfc7693.

[San19] Sanitas. Sanitas Active app | Sanitas health insurance, 2019. https://www.sanitas.com/en/private-customers/contact-help/customer-portal-and-apps/active-app.html.

[SBB19] Lucas Schabhüser, Denis Butin, and Johannes Buchmann. Context hiding multi-key linearly homomorphic authenticators. In *Topics in Cryptology–CT-RSA*, pages 493–513. Springer, 2019. https://doi.org/10.1007/978-3-030-12612-4_25.

[SBTP+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. *Patterns*, 3(5):100487, 2022. https://doi.org/10.1016/j.patter.2022.100487.

[SCF+15] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. VC3: Trustworthy data analytics in the cloud using SGX. In *IEEE Symposium on Security and Privacy (S&P)*, pages 38–54. IEEE, 2015. https://doi.org/10.1109/SP.2015.10.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. https://doi.org/10.1145/322217.322225.

[SEA18] Microsoft SEAL (release 3.0). http://sealcrypto.org, October 2018. Microsoft Research, Redmond, WA.

[Sei18] Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-lwe lattice cryptography. *Cryptology ePrint Archive*, 2018. https://ia.cr/2018/039.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology – CRYPTO*, pages 704–737. Springer, 2020. https://doi.org/10.1007/978-3-030-56877-1_25.

[SFK+22] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Nathan Manohar, Nicholas Genise, Srinivas Devadas, Karim Eldefrawy, Chris Peikert, and Daniel Sanchez. Craterlake: a hardware accelerator for efficient unbounded computation on encrypted data. In *International Symposium on Computer Architecture (ISCA)*, 2022. https://doi.org/10.1145/3470496.3527393.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. https://doi.org/10.1145/359168.359176.

[SNU19] SNUcrypto. HEAAN, 2019. https://github.com/snucrypto/HEAAN.

[SPTP+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. *Annual Network And Distributed System Security Symposium (NDSS)*, 2021. http://dx.doi.org/10.14722/ndss.2021.24119.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology – CRYPTO*, volume 773, pages 13–21. Springer, 1993. https://doi.org/10.1007/3-540-48329-2_2.

[SV10] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography – PKC*, volume 6056, pages 420–443. Springer, 2010. https://doi.org/10.1007/978-3-642-13013-7_25.

[SV14] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71:57–81, 2014. https://doi.org/10.1007/s10623-012-9720-4.

[SW09] Stefan Saroiu and Alec Wolman. Enabling new mobile applications with location proofs. In *Workshop on Mobile Computing Systems and Applications*, pages 1–6, 2009. https://doi.org/10.1145/1514411.1514414.

[TAD16] Fatih Turkmen, Muhammad Rizwan Asghar, and Yuri Demchenko. iGenoPri: Privacy-preserving genomic data processing with integrity and correctness proofs. In *IEEE Annual Conference on Privacy, Security and Trust (PST)*, pages 407–410. IEEE, 2016. https://doi.org/10.1109/PST.2016.7906964.

[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)*, pages 605–616, 2016. https://dl.acm.org/doi/abs/10.1145/2897845.2897892.

[TSSJ+22] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 2779–2792. Association for Computing Machinery, 2022. https://doi.org/10.1145/3548606.3560554.

[UKP19] UKPN. SmartMeter Energy Consumption Data in London Households - London Datastore, 2019. https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households.

[Uni18] United Kingdom Department of Business Energy and Industrial Strategy. 2017/0350/UK - notification of smart metering technical specifications for data communications company (DCC) system release 2, 2018. [Online].

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT*, volume 6110, pages 24–43. Springer, 2010. https://doi.org/10.1007/978-3-642-13190-5_2.

[VDSKK18] Jennifer Valentino-DeVries, Natasha Singer, Michael Keller, and Aaron Krolik. Your apps know where you were last night, and they're not keeping it secret - the new york times, 2018. [Online].

[Via23] Alexander Viand. Heco: Fully homomorphic encryption compiler alexander viand, patrick jattke, miro haller, anwar hithnawi eth zurich. In *USENIX Security Symposium*, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/viand.

[VJH21] Alexander Viand, Patrick Jattke, and Anwar Hithnawi. Sok: Fully homomorphic encryption compilers. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1092–1108, 2021. https://doi.org/10.1109/SP40001.2021.00068.

[VKH23] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. *arXiv preprint arXiv:2301.07041*, 2023. https://arxiv.org/abs/2301.07041.

[VSBW13] Victor Vu, Srinath Setty, Andrew J Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 223–237. IEEE, 2013. https://doi.org/10.1109/SP.2013.48.

[WCGFJ12] 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*, 2012. https://apps.dtic.mil/sti/pdfs/ADA602806.pdf.

[WCP+17] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A Gunter. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2421–2434, 2017. https://doi.org/10.1145/3133956.3134038.

[WL13] Wenye Wang and Zhuo Lu. Cyber security in the smart grid: Survey and challenges. *Computer networks*, 57(5):1344–1371, 2013. https://doi.org/10.1016/j.comnet.2012.12.017.

[WPZM16] Xinlei Wang, Amit Pande, Jindan Zhu, and Prasant Mohapatra. STAMP: Enabling Privacy-Preserving Location Proofs for Mobile Users. *IEEE/ACM Transactions on Networking*, 24(6):3276–3289, 2016. https://doi.org/10.1109/TNET.2016.2515119.

[WTS+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)*, pages 926–943, 2018. https://doi.org/10.1109/SP.2018.00060.

[WW23] Hoeteck Wee and David J Wu. Succinct vector, polynomial, and functional commitments from lattices. In *Advances in Cryptology – EUROCRYPT*, pages 385–416. Springer, 2023. https://doi.org/10.1007/978-3-031-30620-4_13.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1074–1091. IEEE, 2021. https://doi.ieeecomputersociety.org/10.1109/SP40001.2021.00056.

[WYX+21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications

to machine learning. In *USENIX Security Symposium*, pages 501–518, 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/weng.

[WYXZ20] Rui Wen, Yu Yu, Xiang Xie, and Yang Zhang. LEAF: A faster secure search algorithm via localization, extraction, and reconstruction. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1219–1232, 2020. https://doi.org/10.1145/3372297.3417237.

[WZD+16] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. HEALER: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, 32(2):211–218, 2016. https://doi.org/10.1093/bioinformatics/btv563.

[XCP15] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE Symposium on Security and Privacy (S&P)*, pages 640–656. IEEE, 2015. https://doi.org/10.1109/SP.2015.45.

[XHX+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. *IEEE Transactions on Dependable and Secure Computing*, 2022. https://doi.org/10.1109/TDSC.2022.3218793.

[XHZ17] Shuaijianni Xu, Yan He, and Liang Feng Zhang. Cryptanalysis of Tran-Pang-Deng verifiable homomorphic encryption. In *ICISC*, 2017. https://doi.org/10.1007/978-3-319-78556-1_4.

[XLG+23] Guowen Xu, Guanlin Li, Shangwei Guo, Tianwei Zhang, and Hongwei Li. Secure decentralized image classification with multiparty homomorphic encryption. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. https://doi.org/10.1109/TCSVT.2023.3234278.

[XLL+19] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security (TIFS)*, 15:911–926, 2019. https://doi.org/10.1109/TIFS.2019.2929409.

[XLR+20] Guowen Xu, Hongwei Li, Hao Ren, Jianfei Sun, Shengmin Xu, Jianting Ning, Haomiao Yang, Kan Yang, and Robert H Deng. Secure and verifiable inference in deep neural networks. In *Annual Computer Security Applications Conference*, pages 784–797, 2020. https://doi.org/10.1145/3427228.3427232.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science (SFCS)*, pages 162–167. IEEE, 1986. https://doi.org/10.1109/SFCS.1986.25.

[YAZ+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*, pages 147–175. Springer, 2019. https://doi.org/10.1007/978-3-030-26948-7_6.

[YKHK18] Satoshi Yasuda, Yoshihiro Koseki, Ryo Hiromasa, and Yutaka Kawai. Multi-key homomorphic proxy re-encryption. In *International Conference on Information Security*, pages 328–346, 2018. https://doi.org/10.1007/978-3-319-99136-8_18.

[YWZ+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. *Mathematics*, 2022. https://doi.org/10.3390/math10244678.

[YZW+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. *Cell Systems*, 2022. https://doi.org/10.1016/j.cels.2022.08.001.

[ZC11] Zhichao Zhu and Guohong Cao. APPLAUS: A privacy-preserving location proof updating system for location-based services. In *Proceedings IEEE INFOCOM*, pages 1889–1897. IEEE, 2011. https://doi.org/10.1109/INFCOM.2011.5934991.

[ZFW+20] Xianglong Zhang, Anmin Fu, Huaqun Wang, Chunyi Zhou, and Zhenzhu Chen. A privacy-preserving and verifiable federated learning scheme. In *IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020. https://doi.org/10.1109/ICC40277.2020.9148628.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *IEEE Symposium on Security and Privacy (S&P)*, pages 863–880. IEEE, 2017. https://doi.org/10.1109/SP.2017.43.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. https://arxiv.org/abs/1906.08935.

[ZLX+20] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX ATC*, 2020. www.usenix.org/conference/atc20/presentation/zhang-chengliang.

[ZPGS19] Wenting Zheng, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Helen: Maliciously secure coopetitive learning for linear models. In *IEEE Symposium on Security and Privacy (S&P)*, 2019. https://doi.org/10.1109/SP.2019.00045.

Sylvain Chatel

EDUCATION

Swiss Federal Institute of Technology Lausanne (EPFL) 2018 - 2023
Ph.D. in Computer Science (Applied Cryptography) *Lausanne, Switzerland*

Georgia Institute of Technology (Georgia Tech) 2016 - 2017
M.Sc. in Electrical and Computer Engineering *Atlanta, USA*

CentraleSupélec (Supélec) 2014 - 2018
M.Sc. and B.Sc. in Engineering *France*

EXPERIENCE

Swiss Federal Institute of Technology Lausanne (EPFL) 2018 - 2023 · 5 years
Research Assistant *Lausanne, Switzerland*

· My research focuses on providing secure fully homomorphic encryption and secure multiparty computations in the presence of malicious adversaries. I design practical systems to provide security and privacy by working with cutting-edge techniques such as zero-knowledge proofs. Graduation is expected in Fall 2023.

Darktrace 2018 - 8 months
Cybersecurity Consultant *Paris, France*

· Darktrace is a leading company in Machine Learning for corporate cybersecurity. My role involved providing cybersecurity and machine learning expertise, running threat analysis, and supervising architecture deployments.

Airbus Defence and Space - Advanced Telecom Payload Products Team 2017 · 6 months
Research Intern *Toulouse, France*

· Research on satellite telecommunication payloads with active antenna
· Modelisation, benchmarks, and optimizations using Matlab and Python

Automatic Control team of IETR 2016 · 2 months
Research Intern *Rennes, France*

· Preliminary security analysis of the distributed model predictive control (DMPC) model.

TEACHING EXPERIENCE

EPFL COM-402 Information Security and Privacy co-head TA, Fall 2021, 2020, 2019
EPFL COM-405 Mobile Networks TA, Spring 2021, 2022
EPFL CS-523 Advanced topics on privacy enhancing technologies TA, Spring 2020

SELECTED PUBLICATIONS & PROJECTS

* denotes equal contribution

S. Chatel, et al., *"PELTA – Shielding Multiparty-FHE against Malicious Adversaries"*, ePrint 2023/642. Accepted and to appear at CCS 2023

S. Chatel, et al., *"Verifiable Encodings for Secure Homomorphic Analytics"*, ArXiv 2022. Under submission

C. Troncoso, D. Bogdanov, E. Bugnion, **S. Chatel** et al., *"Deploying decentralized, privacy-preserving proximity tracing"*, Commun. ACM 2022, 65, 9, 48–57

S. Chatel, A. Pyrgelis, J-R. Troncoso-Pastoriza, and J-P. Hubaux, *"Privacy and Integrity Preserving Computations with CRISP"*, USENIX Security 2021

S. Chatel, A. Pyrgelis, J-R. Troncoso-Pastoriza, and J-P. Hubaux, *"SoK: Privacy-Preserving Collaborative Tree-based Model Learning"*, PoPETs 2021

C. Troncoso et al., *"Decentralized Privacy-Preserving Proximity Tracing"*, ArXiv 2020

A. Devos*, **S. Chatel***, Grossglauser M., *"Reproducing Meta-learning with differentiable closed-form solvers"*, RML ICLR 2019 and ReScience journal C, 2019.