Mémoire pour l'obtention de l'Habilitation à Diriger des Recherches

Sorbonne Université

Malika IZABACHENE

Contributions to Fully Homomorphic Encryption:
Constructions on the Torus and Techniques for Functional Evaluation

Soutenance qui se tiendra le 17 avril 2026
devant le jury composé de :

Rapporteur·e·s

Jean-Sébastien CORON	Professeur des universités	Université du Luxembourg
Sihem MESNAGER	Professeure des universités	Université Paris 8
Damien STEHLÉ	Directeur scientifique	CryptoLab

Examinateur·trice·s

Caroline FONTAINE	Directrice de recherche	CNRS
Iryna ANDRIYANOVA	Professeure des universités	CY Cergy Paris Université
Daniele MICCIANCIO	Full Professor	University of California San Diego, USA
Damien VERGNAUD	Professeur des universités	Sorbonne Université

Contents

Acknowledgments

I would like to thank Jean-Sébastien Coron, Sihem Mesnager, and Damien Stehlé for the time and attention they devoted to reading my manuscript. I am deeply honoured that they agreed to evaluate it. I also wish to express my gratitude to all the members of the committee for their commitment to the HDR review process and for the honour of serving on my jury. I extend my sincere thanks to Caroline Fontaine for agreeing to chair the jury.

My thanks go to all of those who have contributed, directly or indirectly, to the work presented in this manuscript.My gratitude goes especially to my co-authors for our discussion and their commitment.

I thank the ETIS laboratory for its support, particularly financial support, which enabled my HDR process to take place under favourable conditions. I also thank Sorbonne Université for giving me the opportunity to defend my HDR here. I thank the Almasty team, particularly Damien Vergnaud and Charles Bouillaguet for their availability and their warm welcome at several team events.

Finally, I wish to thank my family and those I share my life with outside of work for their presence and the moments of joy we share.

1 Introduction

This manuscript presents a survey of my research work after 2010, following the completion of my PhD. It primarily focuses on my contributions to Fully Homomorphic Encryption (FHE). A brief overview of my contribution in otther topics is provided later in the introduction. FHE is an encryption technique that enables computation to be performed directly on encrypted data, without requiring decryption. This property has given rise to new computational models in which sensitive data remain protected while they are processed. Progress in FHE over the last decade has opened the way to a wide range of applications, including privacy-preserving cloud computation and delegated computation, as well as secure inference in artificial intelligence systems, where not only sensitive inputs but also model parameters can be protected.

The concept of FHE was formalized by Rivest, Adleman, and Dertouzos in 1978 [RAD78]. In 2009, Gentry [Gen09a] proposed the first construction of an FHE scheme. Although mainly of theoretical interest, this construction estanlished the foundations of subsquent research and led to the development of more efficient FHE schemes.

Almost all known FHE makes use of lattice-based encryption schemes and their ciphertexts include a noise component. Each homomorphic operation introduces additional noise that grows along the evaluation process and might affect the quality of the FHE ciphertexts so that decryption may fail. Gentry's initial construction addresses this issue through bootstrapping, which is a fundamental homomorphic operation that controls the noise growth and enables further homomorphic operations. Noise growth can also be handled without bootstrapping. In leveled schemes, the maximum supported level is tied to a noise parameter, and this noise is consumed incrementally as homomorphic operations are performed.

Since Gentry's first FHE construction proposed by Gentry, numerous schemes have been proposed to enhance both efficiency and practicality with respect for real-world applications. The BGV [BGV12] and B/FV [Bra12, FV12] schemes introduced more efficient leveled homomorphic operations and support efficient homomorphic computation through SIMD-based packing techniques. The GSW scheme, introduced in [GSW13] by Gentry, Sahai and Waters, proposed a new paradigm to perform homomorphic operations, setting the stage for efficient gate-by-gate bootstrapping. The FHE schemes proposed by Alperin-Sheriff and Peikert [AP14] (AP), by Ducas and Micciancio [DM15] (DM), and the GINX and CGGI schemes introduced in [GINX16, CGGI16a] further achieve fast bootstrapping, enabling low-latency, gate-by-gate homomorphic operations [1].

The CKKS scheme [CKKS17] proposed by Cheon, Kim, Kim and Song extends homomorphic encryption to approximate arithmetic over real numbers and achieves high throughput, and is particularly suitable for machine learning functions expressed in a continuous domain.

[1] Several recent optimizations have been introduced since the introduction of these FHE schemes, including NTRU-based variant of the GSW, DM and CGGI schemes, as proposed in [XZD+23, Klu22, BIP+22]. In this introductory chapter, I focus on a selected subset of originally proposed FHE schemes.

Several approaches are possible to classifying the aforementioned FHE schemes, which could vary according to the context. Some of them could arguably be refined considering conversion techniques between FHE ciphertext format. Although the criteria listed below are not exhaustive, FHE schemes could be classified according to the following:

- their encoding: most of the FHE schemes have LWE ciphertexts of the form:

$$c = (\mathbf{a}, b), \text{ where } \mathbf{as} + b = \mathsf{encode}(\mu) + noise$$

  with $\mathbf{s}$ the secret key and where the noise is small and $\mathsf{encode}(\mu)$ denotes the encoding of the message μ. The encoding can embed the message in the least significant bits as in BGV or in the most significant bits as in the B/FV and CGGI schemes, for example, or via approximate encodings as in [CGGI17] (CKKS).

- the way the homomorphic product is performed: the homomorphic product between (ring) LWE ciphertexts could implemented via relinearization as in B/FV and CKKS for example or using an auxiliary scheme as in the DM and CGGI schemes;

- the latency and throughput of homomorphic operations. For example, the CGGI scheme is optimized for low-latency gate evaluation and supports efficient functional bootstrapping, i.e., a discretized function can be evaluated during the bootstrapping operation, while B/FV and CKKS schemes perform well over packed ciphertexts and provide high throughput. However, this categorization could be refined considering recent results showing that scheme switching techniques could help to leverage the feature of one scheme to another. For example, CKKS has recently been shown to also bootstrap ciphertexts that encode binary data efficiently [BCKS24].

1.1 Overview of the manuscript

In this manuscript, I primarily focus on the FHE scheme TFHE (where 'T' stands for torus), which supports highly efficient bootstrapping. However, in the context of the applications presented in this document, other FHE schemes and techniques that are not specific to TFHE are also considered.

The first chapter introduces the building blocks of the TFHE scheme and highlights selected contributions within this framework. The second chapter focuses on homomorphic techniques for evaluating look-up table functions using TFHE. Several amortization techniques for improving the throughput of the homomorphic look-up table evaluation are also presented. The third chapter is devoted to the design of FHE sanitization algorithms within a framework applicable to TFHE. FHE sanitization is a notion closely related to FHE circuit privacy, capturing the privacy of the evaluated circuit, even when the secret key is known to the adversary. A more detailed overview of the content of each chapter is provided below.

TFHE framework, chapter 1. This introductory chapter gives an overview of the fundamental aspects of the TFHE framework introduced in [GINX16, CGGI16a, CGGI17, CGGI20]. In [GINX16], with the collaboration of Nicolas Gama, Phong Nguyen and Xiang Xie, we introduced a generalization of the Gentry Sahai Waters (GSW) [GSW13] scheme over groups and provided a new decryption circuit using homomorphic multiplexers. This allows us to reformulate the AP bootstrapping in our general framework using multiplexer gates and slightly improve the noise overhead from $\tilde{O}(n^2)$ to $\tilde{O}(n^{1.5})$, where n is the dimension of the input LWE ciphertext. In [CGGI16a], together with Ilaria Chillotti, Nicolas Gama and Mariya Georgieva, we improved the practical performance of the gate bootstrapping based on our previous generalized GSW-based bootstrapping, and on the DM scheme proposed earlier by Ducas and Micciancio [DM15]. In this work, the authors proposed a new method to homomorphically evaluate simple logical gate operations (in particular the NAND gate) on encrypted messages and introduced a new technique to bootstrap ciphertexts. Their solution named FHEW is based on the LWE encryption scheme [Reg05] and the ring GSW FHE scheme [GSW13]. Their prototype implementation runs in less than a second on a personal computer. In [CGGI16a], we further improve the FHEW bootstrapping performance to less than one millisecond. We noticed that the internal product of GSW ciphertexts can be replaced by the more efficient external product between a ring GSW and an ring LWE ciphertext. As in [GINX16], LWE decryption is expressed using homomorphic multiplexers, and consequently we "functionally" rely on binary LWE secret key. But this allows us to reduce the bootstrapping key size from 1GB to 24MB while preserving the same security level, as we do not rely on an additional decomposition with respect to the secret key.

In this chapter, we present the TFHE scheme, which includes the description of the encryption scheme over the torus, the TFHE bootstrapping, and the homomorphic operations that can be performed in this framework and that will be used in the rest of the manuscript. We also discuss the comparison between the bootstrapping having the same blueprint as TFHE. The last section presents the functional bootstrapping within the TFHE framework, which enables the homomorphic evaluation of a unitary discretized function through FHEW-like bootstrappings.

Homomorphic techniques for arbitrary function evaluations, chapter 2. This chapter introduces computational models for the homomorphic evaluation of arbitrary functions. We first recall the method we introduced in [GINX16, CGGI16a] for the evaluation of an arbitrary boolean function over encrypted inputs and acyclic fixed-time deterministic automata, with a noise growth that is linear in the number of inputs. Our generalization framework led us to propose a generic way to homomorphically evaluate binary decision diagrams using a tree of homomorphic multiplexers. We also outline the approach we proposed in [CGGI17] to evaluate arithmetic functions on encrypted binary words using Deterministic Weighted Finite Automata (DWFA). In DWFA, a weight, associated with each transition is accumulated during the computation. The accumulated weight permits the encoding of arithmetic information along the evaluation so that arithmetic operations can be implemented more efficiently than using deterministic automata.

The second section provides an overview of our contribution to the amortization of homomorphic computations. We focus on the optimization techniques proposed for the *functional bootstrapping*. In [CIM19], we introduced the multi-value bootstrapping, which supports the evaluation of arithmetic functions over a larger domain than regular gate bootstrapping. This work demonstrated how several functions can be evaluated on the same input at almost no additional cost compared to the regular bootstrapping. Our implementation of the multi-value bootstrapping run in 1.57 seconds on a 6-to-6 bit arbitrary function. And the evaluation of additional 128 outputs on the same 6 input bits takes only 0.05 more seconds. This section presents the strategy and discusses applications for the multi-value bootstrapping. The second section reviews packing techniques and how they can be used to optimize the homomorphic evaluation of an arbitrary function. We describe the *vertical packing* and *horizontal packing*, which pack a set of LWE ciphertexts of scalar messages into one ring LWE ciphertexts of a polynomial message. In [BI25a], Jean-Philippe Bossuat and I proposed an iterative variant of the recursive packing from [CDKS21] that makes use of homomorphic automorphism evaluations to scale the powers of input message coefficients into one polynomial. At the end of this section, we additionally present our dynamic variant where the input ciphertexts are streamed, i.e. one ciphertext at a time is processed, which allows to decrease the internal memory from N to $\log N$, where N is the polynomial ring degree of the packed ciphertext polynomials.

Circuit privacy, FHE sanitization and applications, chapter 3. This chapter presents our contributions on FHE circuit privacy. This property ensures that no information about the circuit is leaked through the homomorphic evaluation. An FHE sanitization algorithm is related to circuit privacy, as it removes any information contained in a ciphertext except the plaintext. This notion and the first strategy for the construction of an FHE sanitization algorithm were proposed by Ducas and Stehlé [DS16]. They introduced a sanitization strategy that iterates a rerandomization step followed by a bootstrapping. This approach can be applied generically to bootstrappable FHE schemes and requires circular security. Together with Florian Bourse, we proposed an alternative strategy [BI22] to construct an FHE sanitization algorithm by modifying the TFHE bootstrapping internally. Our construction relies on an adaptation of the randomization technique for GSW ciphertexts proposed by Bourse *et al.* [BdPMW16] over rings and the introduction of a public rerandomizer that allows one to simulate the correct distribution of an FHE ciphertext, without knowing the computation history. We also discuss other proposed sanitization algorithms, provide a comparison with our results, and present applications of our construction beyond circuit privacy.

1.2 Selected Fully Homomorphic Encryption Applications

This section reviews selected FHE applications that I have worked on and which will be referred to throughout this manuscript. The first application concerns a solution we proposed for a real-world use case, in which the goal is to construct a private heatmap

over encrypted coordinates specified by latitude and longitude. The goal is for a client to obtain a visualization of location density while individual points remain private. With the collaboration of Ilia Iliashenko, Axel Mertens, and Hilder Vitor Lima Pereira, we proposed two homomorphic methods in [IIMP22] to count the number of encrypted inputs that verify the same property. In the context of a heatmap, this corresponds to the homomorphic counting of encrypted points that lie in the same section of a grid.

With Jean-Philippe Bossuat, we recently proposed an efficient solution in [BI25b] enabling a data scientist to probe large datasets while preserving both the privacy of the probing function (i.e. privacy with respect to the data scientist) and patient's record, except for the information revealed by the homomorphic computation result. In this work, we introduced a composable set of homomorphic operations using the CKKS scheme [CKKS17], which enables large-scale private data analysis.

Homomorphic encryption has also been applied to design e-voting protocols. Together with Véronique Cortier, David Galindo and Stéphane Glondu, we proposed an e-voting protocol [CGGI13b, CGGI13a, CGGI14] that constitutes a variant of the e-voting system Helios [AdMP]. Compared to previous construcions, ours achieves full verifiability of the election result in the presence of a dishonest bulletin board. This means that voters can verify the result without relying on an additional authority even if the bulletin board behaves maliciously. Since my postdoctoral work on Belenios, the protocol has continued to evolve [Bel], with additional features developed by the team at Loria working on it. In [CGGI16b], together with Nicolas Gama, Mariya Georgieva, and Ilaria Chillotti, we proposed a post-quantum analog of *Benelios* using a FHEW-like bootstrappable FHE scheme, where the multiplicative ElGamal encryption is replaced by scale-invariant LWE encryption over the torus. We also adapted the scheme to support distributed decryption between the tally authorities which are in charge of decrypting the election result and to ensure public verifiability of the final result.

1.3 Additional contributions with PhD students

My research has also focused on protecting user data privacy in settings where an adversary may act actively by issuing decryption oracle queries. In [BIL20, BIL21], we studied the design of post-quantum identity-based encryption (IBE) schemes that proved to be secure against active attacks, i.e., an adversary may act actively and has access to decryption oracle queries in the associated security model. While applying the Fujisaki-Okamoto transformation [FO13] to a weakly secure encryption scheme (e.g., one with chosen-plaintext or CPA security) enables the design of a strongly secure scheme, we considered direct strongly secure constructions that do not rely on generic paradigms and the random oracle model.

As part of Lucas Prabel's PhD research work I co-supervised with Pierre-Alain Fouque and Adeline Roux-Langlois, we studied the design and security of post-quantum IBE schemes (i.e. secure against a quantum adversary) using approximate trapdoors. We proposed two-constructions; the first one is built on the paradigm proposed by Agrawal, Boneh and Boyen [ABB10] and relies on the decisional LWE over modules [BGV12, LS15];

the second one relies on a variant of the NTRU assumption [HPS98]. In [ISZ19], we explored the use of FHE to homomorphically evaluate a special class of neural networks called Hopfield networks. Using this neural network family, we demonstrated that by encrypting both the model (the network topology and the weights) and the user's data, the resulting homomorphic evaluation provides reasonable timings. This work was part of Martin Zuber's PhD thesis I co-supervised at the beginning of his PhD when I was working at CEA. As part of the Paola de Perthuis PhD thesis I co-supervised with David Pointcheval when I was working at Cosmian, we investigated in [INdPP22] how to provide proofs attesting to the correctness of homomorphic computations built from the BFV scheme. We designed a low-communication oblivious polynomial evaluation protocol in malicious settings where the computation can be shown to be performed correctly. As an application, our work can be used to demonstrate the correctness of FHE computations in Private Information Retrieval protocols [CGKS95, KO97] and Private Set Intersection protocols [FNP04].

2 Torus Fully Homomorphic Encryption

This chapter introduces the Fully Homomorphic Encryption scheme over the torus, called TFHE together with the homomorphic operations supported within the TFHE framework. We provide a detailed description of the internal building blocks of its fast bootstrapping. As outlined in the introduction, this chapter focuses on TFHE and other bootstrappings that follow a similar blueprint as TFHE and present a comparitive overview between the different schemes proposed in this line of work. This chapter focuses on the design aspects and addresses efficiency considerations at a conceptual level [2]. This chapter is associated with publications [GINX16, CGGI16a, CGGI17, CGGI20].

2.1 Definitions for FHE

In this manuscript, we focus on secret key fully homomorphic encryption, where the same key is used both for encryption and decryption. This is sufficient for many applications. However, our results can be easily generalized to the public key setting if needed by defining a pubic encryption key as a set of encryptions of zero. We provided further details in remark 2.

Definition 1 (Fully Homomorphic Encryption)**.** *A fully homomorphic encryption scheme is given by four polynomial time algorithms,* (KeyGen, Encrypt, Decrypt, Eval) *defined as follows:*

- $\mathsf{KeyGen}(1^\lambda)$*: on input a security parameter* λ*, outputs an evaluation key* EVK *and a secret key* SK*;*
- $\mathsf{Encrypt}(SK, \mu)$*: on input a secret key* SK *and a message* μ*, returns a ciphertext* CT*;*
- $\mathsf{Decrypt}(SK, CT)$*: on input a secret key* SK *and a ciphertext* CT*, returns a message* μ*;*
- $\mathsf{Eval}(EVK, f, CT_1, \ldots, CT_t)$*: on input an evaluation key* EVK*, a function* f *on* t *inputs, and* t *ciphertexts* $CT_1, \ldots, CT_t$*, it returns a ciphertext* CT_f*.*

We denote $\mathcal{M}$ the message space and $\mathcal{C}$ the ciphertext space. For $\mu \in \mathcal{M}$, we define $\mathcal{C}_\mu = \mathsf{Decrypt}(\mathrm{SK}, \cdot)^{-1}(\mu)$, the set of all ciphertexts that decrypt to μ. We say that an FHE scheme is *correct* if, for $(\mathrm{EVK}, \mathrm{SK})$ sampled from $\mathsf{KeyGen}(1^\lambda)$:

- for all messages $\mu \in \mathcal{M}$: $\mathsf{Encrypt}(\mathrm{SK}, \mu) \in \mathcal{C}_\mu$ with overwhelming probability.
- for all functions $f : \mathcal{M}^t \to \mathcal{M}$, $(\mu_1, \ldots, \mu_t) \in \mathcal{M}^t$, $(\mathrm{CT}_1, \ldots, \mathrm{CT}_t) \in \mathcal{C}_{\mu_1} \times \ldots \times \mathcal{C}_{\mu_t}$: $\mathsf{Eval}(\mathrm{EVK}, f, \mathrm{CT}_1, \ldots, \mathrm{CT}_t) \in \mathcal{C}_{f(\mu_1, \ldots, \mu_t)}$ with overwhelming probability;

Remark 1. *FHE ciphetexts being noisy, this definition of correctess for* Eval *is often relaxed to hold for low noise ciphertexts in* $\mathcal{C}$ *in input. However, this definition can still be met by starting* Eval *with a call to the bootstrapping procedure for each input.*

[2] For concrete performance and benchmarks, we refer the reader to existing implementations of TFHE ([Zam22, Hwa23] for example), which include different features and optimizations and are currently still under active development.

We say that an FHE scheme is *compact* if the ciphertexts are of polynomial size. We say that an FHE has *indistinguishability under chosen plaintext attacks (IND-CPA security)* or is *semantically secure* if no polynomial time adversary can have a non-negligible advantage in guessing a bit β given oracle access to the function $(\mu_0, \mu_1) \mapsto \mathsf{Encrypt}(\mathrm{SK}, \mu_\beta)$.

Notations: the set of integers from 1 to n will be denoted $[1, n]$ for convenience. We use lower case bold font, e.g. $\mathbf{a}$, to denote (possibly row) vectors, and upper case bold font, e.g. $\mathbf{A}$, to denote matrices. We write $\mathbf{A}^T$ the transpose of the matrix $\mathbf{A}$. We will use $\otimes$ to denote the Kronecker product of two matrices. We write χ_ϑ for a noise distribution with variance ϑ.

2.2 Encryption over the dicretized torus

In the following, $\mathbb{T}$ denotes the set of real numbers modulo one, and the discretized torus $\hat{\mathbb{Z}}_q = \{0, \frac{1}{q}, \cdots, \frac{q-1}{q}\}$ is $\frac{1}{q}\mathbb{Z} \cap \mathbb{T}$. For N a fixed power of 2, $R := \mathbb{Z}[X] \mod X^N + 1$ denotes the N-th cyclotomic polynomial ring and $\hat{R}_q$ is the set of polynomials with coefficients in $\hat{\mathbb{Z}}_q$ modulo $X^N + 1$.

LWE *encryption.* In order to LWE encrypt a message μ from the message space $\mathcal{M}$ using the secret key $\mathbf{s} \in \{0,1\}^n$, one first defines a noisy encoding for μ as $\mu^\star = \mu + e$ where e is sampled from χ_ϑ and chooses a uniform vector $\mathbf{a}$ in $\hat{\mathbb{Z}}_q^n$. The encryption of μ is then $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mu^\star) \in \hat{\mathbb{Z}}_q^{n+1}$. In order to decrypt $(\mathbf{a}, b)$ using $\mathbf{s}$, one first retrieves the noisy encoding as $\mu^\star := b - \mathbf{a} \cdot \mathbf{s}$ and rounds a rescaling of $\mu^\star$ to the nearest element in $\mathcal{M}$ to retrieve the message μ. The encoding and decoding functions are defined with respect to the message space. Other encodings can be defined. In particular, if the message space is $\hat{\mathbb{Z}}_p$ with $p|q$, the noisy encoding of $\mu \in \hat{\mathbb{Z}}_p$ is defined as before and, to decrypt, the decoding function first retrieves the representative of $\mu \in \mathbb{Z}_p$ and rescales it over $\hat{\mathbb{Z}}_p$. The decisional LWE problem, parameterized by n and χ_ϑ, asks to distinguish the uniform distribution over $\hat{\mathbb{Z}}_q^{n+1}$ from the distribution of fresh LWE encryptions of zero, all encrypted using the same secret key $\mathbf{s} \in \{0,1\}^n$, sampled uniformly at random.

Remark 2. *We described in [CGGI16b] how the TFHE symmetric key encryption scheme can be converted to public key encryption. The public key is defined as a set of m LWE encryptions of 0. In order to ensure that the public key has sufficiently high entropy, we rely on the leftover hash lemma, which implies that m is of the order of $O(n \log q)$. Then, to encrypt a message μ, one takes a random subset sum of the m encryptions of 0 and adds the vector $(0, \cdots, 0, \mu)$. Joye [Joy24] recently presented variants of this method that allow smaller key sizes. In [CGGI16b], we also described a multi-user variant which allows support shared decryption. The construction is used in the context of evoting, where privacy of the votes is guaranteed, and decryption of the election result works as it should, even if all but a subset of the trustees leak their private keys.*

RLWE *encryption.* The RLWE (ring LWE) encryption scheme encrypts a noisy encoding of the message $\mu \in \hat{R}_q$ with the secret key $\tilde{\mathbf{s}}$, sampled uniformly at random over $\mathbb{B}[X]$, is defined as: $(\mathbf{a}, \mathbf{a} \cdot \tilde{\mathbf{s}} + \mu + e)$, where $\mathbf{a}$ is uniformly random in $\hat{R}_q$, e has coefficients sampled from χ_ϑ. $\mathsf{RLWE}_{\tilde{\mathbf{s}},\vartheta}(\mu)$ denotes a fresh RLWE encryption of μ with noise variance ϑ, the variance will be often omitted. Its semantic security relis on the hardness of the LWE over rings [LPR10, SSTX09].

GLWE *encryption.* The GLWE (generalized LWE) encryption scheme [BGV12, LS15] is defined similarly to the RLWE scheme, except that it includes an additional parameter k. The encryption of $\mu \in \hat{R}_q$ with the secret key $\tilde{\mathbf{s}} = (\tilde{\mathbf{s}}_1, \cdots, \tilde{\mathbf{s}}_k)$, where $\tilde{\mathbf{s}}_i$ is sampled uniformly at random over $\mathbb{B}[X]$ for $i \in [1, k]$, is defined as: $(\mathbf{a}, \mathbf{a} \cdot \tilde{\mathbf{s}} + \mu + e)$, where $\mathbf{a} = (\mathbf{a}_1, \cdots, \mathbf{a}_k)$, each $\mathbf{a}_i$ is sampled uniformly at random in $\hat{R}_q$ and e has coefficients sampled from χ_ϑ. RLWE encryption is a special case of GLWE encryptions with $k = 1$. To simplify the presentation and illustrations, we employ RLWE, but all results remain valid when using GLWE encryption instead.

Deterministic gadget decomposition. Gadget decomposition plays a key role in Ring-LWE based homomorphic encryption computations. A gadget vector $\mathbf{g}$ is a fixed vector in $\hat{\mathbb{Z}}_q^\ell$.

A function $\mathsf{dec}_{\mathbf{g}} : \hat{\mathbb{Z}}_q \mapsto \mathbb{Z}^\ell$ is called *gadget decomposition* for $\mathbf{g}$ if for any $a \in \hat{\mathbb{Z}}_q$, $\|\mathsf{dec}_{\mathbf{g}}(a)\|_\infty \leq \delta$ and $\|\langle \mathsf{dec}_{\mathbf{g}}(a), \mathbf{g} \rangle - a\|_\infty \leq \epsilon$, where δ, ϵ defines the quality and precision bounds of the decomposition. The gadget vector is often chosen as $\mathbf{g} = \left(1/B, \cdots, 1/B^\ell\right)$ for a radix B such that B^ℓ divides q. For $a \in \hat{\mathbb{Z}}_q$, we define $\mathsf{dec}_{\mathbf{g}}(a) := (\bar{a}_1, \cdots, \bar{a}_\ell)$ where $\bar{a} = \sum_{j=1}^{\ell} \bar{a}_j B^{-j}$ with $\bar{a}_j \in \left(\frac{-B}{2}, \frac{B}{2}\right]$. With these notations, $\bar{a} \in \frac{1}{B^\ell}\mathbb{Z}$ is the closest multiple of $\frac{1}{B^\ell}$ to a and we can take $\epsilon = \frac{1}{2B^\ell}$ and $\delta = \frac{B}{2}$.

This gadget decomposition can be extended to vectors over $\hat{\mathbb{Z}}_q^d$ for a gadget matrix $\mathbf{G} = I_d \otimes \mathbf{g}$, by decomposing coordinate-by-coordinate with respect to $\mathbf{g}$. In our case, we will consider exact decompositions and denote the gadget decomposition with respect to $\mathbf{g}$ (resp. $\mathbf{G}$) as $\mathbf{g}^{-1}(\cdot)$ (resp. $\mathbf{G}^{-1}(\cdot)$).

The homomorphic product in TFHE is based on the generalization of this decomposition on $\hat{R}_q^{\ d}$: any $\mathbf{v} \in \hat{R}_q^{\ d}$ can be uniquely and deterministically decomposed into a small vector $\mathbf{G}^{-1}(\mathbf{v}) \in R^{d \cdot \ell}$ whose coefficients are integers in $[-B/2, B/2[$ and such that $\mathbf{G}^{-1}(\mathbf{v}) \cdot \mathbf{G} = \mathbf{v}$.

RGSW encryption. Using the ring variant [DM15] of the GSW encryption scheme [GSW13], a fresh RGSW encryption of $\mu \in R$ is defined as:
$\mathsf{RGSW}_{\tilde{\mathbf{s}}}(\mu) := Z + \mu \cdot \mathbf{G} \in (\hat{R}_q^{\ 2})^{2 \cdot \ell}$, where $\mathbf{G} = \begin{pmatrix} 1/B & \cdots & 1/B^\ell & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1/B & \cdots & 1/B^\ell \end{pmatrix}^T$
and Z is a set of 2ℓ RLWE encryptions of zero.

An RGSW ciphertext encrypting a message μ can alternatively be written as (a, b) with:

$$\begin{aligned} a &= \left(\mathsf{RLWE}(-\tilde{s}\cdot\frac{\mu}{B}), \mathsf{RLWE}(-\tilde{s}\cdot\frac{\mu}{B^2}), \cdots, \mathsf{RLWE}(-\tilde{s}\cdot\frac{\mu}{B^\ell})\right), \text{ and} \\ b &= \left(\mathsf{RLWE}(\frac{\mu}{B}), \mathsf{RLWE}(\frac{\mu}{B^2}), \cdots, \mathsf{RLWE}(\mu\frac{1}{B^\ell})\right) \end{aligned}$$

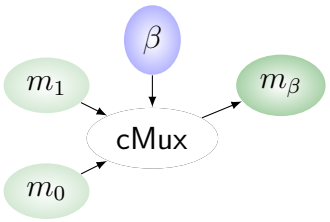
Defining $\text{SK} := (\tilde{s}, -1) \in R^2$, an RGSW ciphertext encrypting μ can then be defined as a matrix whose t-th row encrypts $-\text{SK}_j \frac{\mu}{B^i}$, with $t = i + (j-1)\ell$ and $0 \leq i < \ell$ and $0 \leq j < 2$.

2.3 Homomorphic operations and efficient bootstrapping

External Product. The LWE, RLWE encryption schemes support homomorphic addition and homomorphic internal multiplication as defined in [Bra12, FV12], using an additional relinearization step. In the TFHE framework, the homomorphic product is defined using an external product between RLWE ciphertexts and RGSW ciphertexts. This operation has previously been used in [BP16], and we later formalized it in [CGGI16a].

The external product $\boxdot$ homomorphically evaluates the homomorphic product of an RLWE encryption V of a torus polynomial message v and an RGSW encryption $\mathbf{C}$ of an integer polynomial message μ: $\mathbf{C} \boxdot \mathbf{V} := \mathbf{G}^{-1}(\mathbf{V}) \cdot \mathbf{C}$, where $\mathbf{G}^{-1}(\mathbf{V})$ is the deterministic decomposition with respect to the gadget matrix $\mathbf{G}$. The RGSW scheme also supports addition and homomorphic internal multiplication, defined using $\mathbf{G}^{-1}$ decomposition. The decomposition produces a small (unique) vector $\mathbf{G}^{-1}(\mathbf{V}) \in R^{1\times(d+1)\cdot\ell}$ whose coefficients are integers in $[-B/2, B/2[$ and such that $\mathbf{G}^{-1}(\mathbf{V}) \cdot \mathbf{G} = \mathbf{V}$.

As shown in [GINX16, CGGI16a], the homomorphic product can be used to evaluate homomorphic multiplexers (cMux) over ciphertexts as follows: given two RLWE encryptions $V_0, V_1 \in \hat{R}_q^{\ 2}$ of $\mu_0, \mu_1 \in \hat{R}_p$, an RGSW ciphertext $C_\beta \in (\hat{R}_q^{\ 2})^{2\cdot\ell}$, for $\beta \in \{0, 1\}$, all encrypted with the same secret key $\tilde{s}$, $\mathsf{cMux}(C_\beta, \mathbf{V}_1, \mathbf{V}_0) := \mathbf{V}_0 + \mathbf{C}_\beta \boxdot (\mathbf{V}_1 - \mathbf{V}_0)$. The result is a valid RLWE encryption of μ_β with secret key $\tilde{s}$.



Keyswitching key. Given a radix B', a gadget vector $\mathbf{g}'$ and decomposition $\mathbf{g}^{-1}$, all for keyswitching, a keyswitching key $\mathsf{ksk}_{\mathbf{s}'\mapsto\mathbf{s}}$ from secret key $\mathbf{s}' \in \{0,1\}^{n'}$ to secret key $\mathbf{s} \in \{0,1\}^n$ is a sequence of fresh LWE ciphertexts $\mathsf{ksk}_{i,j} \in \mathsf{LWE}(s_i' B'^{-j})$ for $i \in [0, n']$ and $j \in [1, \ell']$.

Keyswitching. Given an LWE ciphertext $\mathbf{c}' = (\mathbf{a}', b') \in \hat{\mathbb{Z}}_q^{n+1}$ of a message μ and a keyswitching key $\mathsf{ksk}_{\mathbf{s}'\mapsto\mathbf{s}}$, the keyswitching algorithm returns $(\mathbf{0}, b') - \sum_{i=0}^{n'-1} \mathsf{dec}_{\mathbf{g}'}(a_i')^T \cdot \mathsf{ksk}_i$ which is an LWE sample in $\mathcal{C}_\mu$ as long as the noise remains manageable. In [CGGI17], We also extended keyswicthing to take a lipschitzian morphism $f : \mathbb{T}^p \mapsto \hat{R}_q$ and convert a set of p

LWE encryptions of μ_i under secret key $\mathbf{s}$ into a valid RLWE encryption of $f(\mu_1, \cdots, \mu_p)$ under secret key $\tilde{s}$. We call this extended keyswicthing *public (resp. private) functional keyswicthing* when f is a public (resp. secret) function.

Sample extraction. Given as input an RLWE encryption $c = (a, b) \in R_q^2$ of a polynomial message $\mu(X) = \sum_{i=0}^{N-1} \mu_i X^i$ under secret $\tilde{\mathbf{s}}$, with $b = \sum_{i=0}^{N-1} b_i X^i$ and $a = \sum_{i=0}^{N-1} a_i X^i$, $\mathsf{SampleExtract}$ returns $(a_0, -a_{N-1}, \cdots, -a_1, b_0) \in \mathbb{Z}_q^{N+1}$. The output is an LWE encryption of μ_0 under secret key $\mathbf{s} = (\tilde{s}_0, \cdots, \tilde{s}_{N-1}) \in \mathbb{Z}^N$.

BlindRotation. BlindRotation plays a central role in many bootstrapping algorithms. It is defined by the use of an accumulator associated to three operations we detailed below, and illustrated in Figure 1:

(i) Init: it takes as input an integer polynomial $\mathsf{testv} \in \hat{R}_q$, called *test vector* or *test polynomial*, an LWE encryption $(\bar{\mathbf{a}}, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$ under secret key $\mathbf{s} = (s_0, \cdots, s_{n-1})$ and a bootstrapping key, which comprises fresh RGSW encryption of the s_i's under secret key $\tilde{s}$. It first sets up an accumulator as a trivial (i.e., the left part is equal to zero) and noiseless RLWE ciphertext $\mathsf{ACC}_0 := (\mathbf{0}, v(X))$;

(ii) Update: it updates the accumulator as: $\mathsf{ACC}_i := \mathsf{ACC}_{i-1} + (X^{\bar{a}_i} - 1) \cdot (\mathsf{ACC}_{i-1} \boxdot \mathsf{RGSW}_{\tilde{s}}(s_i))$;

(iii) $\mathsf{Extract}$: after applying a chain of sequential cMux, it extracts an LWE encryption of the constant coefficient of $\mathsf{ACC}_n = \mathsf{RLWE}_{\tilde{s}}(v(X) \cdot X^{-\bar{b}+\langle \bar{a}, s \rangle})$ using sample extraction.

Remark 3. *An iteration in the BlindRotation consists of homomorphically adding $\bar{a}_i s_i$ to the accumulator ACC_{i-1} in the exponent. As s_i is binary, we have $s_i \cdot (X^{\bar{a}_i} - 1) + 1 = X^{\bar{a}_i s_i}$. In the encrypted domain, the iteration at step i can be expressed as an external product between ACC_{i-1} and $\mathsf{RGSW}_{\tilde{s}}(X^{\bar{a}_i s_i}) = 1 + (X^{\bar{a}_i} - 1) \cdot \mathsf{RGSW}_{\tilde{s}}(s_i)$. This update operation can alternatively be written as $\mathsf{ACC}_i := \mathsf{ACC}_{i-1} + (X^{\bar{a}_i} - 1) \cdot (\mathsf{ACC}_{i-1} \boxdot \mathsf{RGSW}_{\tilde{s}}(s_i))$.*

Remark 4. *There are alternative ways to homomorphically compute $\sum_{i=0}^{n-1} \bar{a}_i s_i$ in the exponent. For example, the automorphism-based BlindRotation from [LMK+23] combines homomorphic automorphism and external product evaluations with respect to the decomposition of the multiplicative group $\mathbb{Z}_{2N}^{\star}$.*

2.3.1 TFHE Gate Bootstrapping

Given an LWE encryption of a message μ, the purpose of the bootstrapping is to generate a ciphertext of the same message with a noise reset to a controlled level. In [GINX16], we proposed a gate bootstrapping procedure called TFHE bootstrapping, and a library, TFHE [CGGI16c] implementing the homomorphic operations on the torus. which achieves substantially better performance than FHEW [DM15] and its variants [AP14, GINX16]. Any boolean gate can then be homomorphically evaluated based on this bootstrapping. We first present the TFHE gate bootstrapping (Agorithm 1), which homomorphically evaluates the decryption circuit as a sign function. We then highlight the key differences between the bootstrapping procedures following the same blueprint as TFHE. And we show how the gate bootstrapping can be extended to support the evaluation of an arbitrary function.

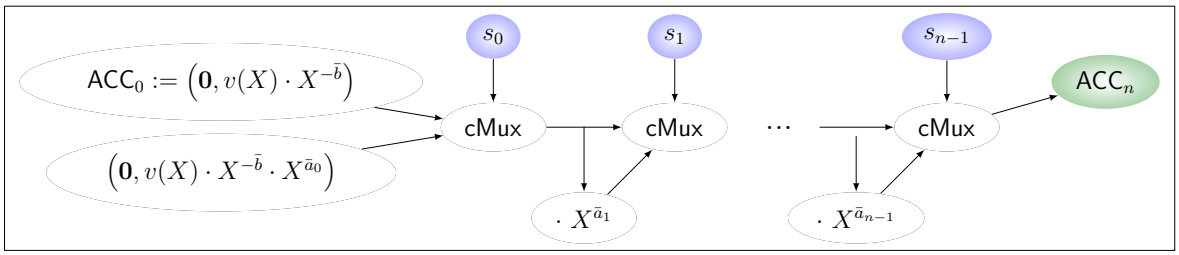


Figure 1: Illustration of the BlindRotation in TFHE: RLWE ciphertexts $\mathsf{ACC}_0 := (\mathbf{0}, v(X) \cdot X^{-\bar{b}})$ and $\mathsf{ACC}_0 := (\mathbf{0}, v(X) \cdot X^{-\bar{b}})$ are both taken as input to the cMux and BK_0 is taken as the controlled bit, then cMux evalutions are performed with inputs ACC_{i-1} and $\mathsf{ACC}_{i-1} \cdot X^{\bar{a}_i}$ and with BK_i as the controlled bit. At the end, the output ACC_{n-1} corresponds to the RLWE encryption of $v(X) \cdot X^{\bar{b}-\langle \bar{\mathbf{a}}, \mathbf{s} \rangle}$.

Algorithm 1 TFHE gate bootstrapping from LWE-to-LWE samples

Require: $(\mathbf{a}, b)$ an LWE encryption of μ, a bootstrapping key $\mathrm{BK} = (\mathrm{BK}_i)_i$, where BK_i is a RGSW encryption of s_i for $i \in [1, n]$, a keyswitching key KS, and two fixed messages $\mu_0 (= 0), \mu_1 (= \frac{1}{2})$.

Ensure: an LWE encryption $(\mathbf{a}', b')$ of μ_0 if $(\mathbf{a}, b) \in \mathcal{C}_0$ and μ_1 if $(\mathbf{a}, b) \in \mathcal{C}_{\frac{1}{2}}$.

1: $\bar{\mu} = \frac{\mu_0 + \mu_1}{2}$ and $\bar{\mu}' = \mu_0 - \bar{\mu}$
2: $\mathsf{testv} = (1 + X + \cdots + X^{N-1}) \cdot X^{-\frac{N}{2}} \cdot \bar{\mu}'$
3: $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a}_i = \lfloor 2Na_i \rceil$
4: $\mathsf{ACC} = (0, \ldots, 0, \mathsf{testv} \cdot X^{-\bar{b}}) \in \hat{R}_q^{\,d+1}$,
5: **for** $i = 1$ **to** $n - 1$
6: $\quad \mathsf{ACC} += \mathrm{BK}_i \boxdot ((X^{\bar{a}_i} - 1)\mathsf{ACC})$
7: $\mathbf{c} = (\mathbf{0}, \bar{\mu}) + \mathsf{Extract}(\mathbf{c}')$
8: Return $\mathsf{KeySwitch}(\mathrm{KS}, \mathbf{c})$

Combining altogether - TFHE Gate Bootstrapping

(i) *Initialization:* This step initializes the public parameters $(n, N, q, \mathbf{g}, \mathbf{g}^{-1}, \mathbf{g}', \mathbf{g}^{-1'})$ where $\mathbf{g}, \mathbf{g}^{-1}$ (resp. $\mathbf{g}', \mathbf{g}^{-1'}$) is the gadget decomposition parameter for the external product (resp. for the keyswicthing). It also includes the variance of the input LWE error and the variance associated to fresh RGSW ciphertexts in the bootstrapping key. The first step additionally initializes a test vector polynomial, $\mathsf{testv} := v(X)$, where $v(X) = \sum_{i=0}^{N-1} v_i X^i \in \hat{R}_p$. We explain in the next paragraph how the coefficients v_i of the test vector polynomial are chosen. The notation testv will be used in the two next chapters.

(ii) *Modulus Switching:* it first rounds each coefficient of the LWE input: $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a}_i = \lfloor 2Na_i \rceil \in \mathbb{Z}_{2N}$ for each $i \in [0, n-1]$. The rounding is done modulo $2N$ as the ciphertext coefficients b, a_i will be encoded in the exponent, and X has order $2N$ in the multiplicative group R. The rounding over $\mathbb{Z}_{2N}$ introduces a cumultae error that is called the *error drift*.

Remark 5. *As this rounding step introduces a rounding error, it could change the result of decryption. In order to take into account this additional rounding error, we define the ciphertext space for ciphertexts that decrypt correctly to a given message. For $\mu \in \hat{\mathbb{Z}}_p$, we define*

$$\mathcal{C}_\mu = \left\{ (\mathbf{a}, b) \in \hat{\mathbb{Z}}_q^{1\times(n+1)}, \left| \lfloor 2Nb \rceil - \sum_{i=1}^{n} s_i \lfloor 2Na_i \rceil - \mu \right| \leq \frac{q}{2p} \right\}.$$

(iii) *BlindRotate:* it applies the BlindRotation algorithm with inputs the test vector $v(X) \in \hat{R}_p$, the rounded LWE ciphertext $\left(\bar{\mathbf{a}}, \bar{b}\right) \in \mathbb{Z}_{2N}^{n+1}$ and the bootstrapping keys $\text{BK}_i \in \mathsf{RGSW}_{\tilde{s}}(s_i)$, for $0 \leq i < n$.

(iv) *Keyswitching:* given an LWE ciphertext under secret key extracted from $\tilde{s}'$ and of dimension N, it outputs an LWE encryption with dimension n of the same message encrypted with secret key $\mathbf{s}$.

Polynomial Rounding: in this paragraph, we explain how the test vector coefficients are chosen. After the $\mathsf{BlindRotate}$, a target LWE ciphertext can be recovered by extracting the constant coefficient of $X^{\bar{b}-\langle\bar{\mathbf{a}},\mathbf{s}\rangle} \cdot v = X^{\bar{b}-\langle\bar{\mathbf{a}},\mathbf{s}\rangle} \cdot \sum_{i=0}^{N-1} v_i X^i$:

a) The constant coefficient of $X^{-j}\sum_{i=0}^{N-1} v_i X^i$ is v_j if $0 \leq j < N$ and $-v_j$ if $N \leq j < 2N$. This comes from the fact that the coefficients of v are rotated by $-j$ positions and $X^{-j} = X^{2N-j} = -X^{N-j}$. This means that N distinct values can be encoded in $\frac{1}{2N}\mathbb{Z}$.

b) As the goal is to obtain an LWE encryption of a message encoded in $\hat{\mathbb{Z}}_p$, the coefficients of v_j are defined by mapping each element in $\frac{1}{2N}\mathbb{Z}$, $\frac{j}{2N}$ to its nearest representative in $\hat{\mathbb{Z}}_p$, i.e., $v_j := \frac{\lfloor \frac{pj}{2N} + \frac{1}{2p} \rfloor \mod p}{p}$

2.3.2 FHEW-like bootstrappings

There are different bootstrapping constructions that follow the same blueprint as in TFHE. All of these methods are based on the use of the GSW encryption schemes. The first approach was proposed in the plain LWE setting by Alperin-Sherif and Peikert [AP14] which was later improved by Ducas and Micciancio [DM15] in the ring setting. Their bootstrapping algorithm, called FHEW put forward the first practical bootstrapping, their proof-of-concept implementation run in less than 1 second. In [GINX16], we proposed a new bootstrapping approach where the decryption circuit is expressed as a sequence of multiplexers implemented using plain GSW ciphertexts. In [CGGI16a], we introduced several improvements to this construction that can be summarized as follows: while [DM15] evaluates the homomorphic product using an internal GSW-GSW product over rings, we make use of the external product [BP16, CGGI16a] between ring GSW ciphertexts and ring LWE ciphertexts. The linear part of the decryption function is then evaluated as in [GINX16] using homomorphic multiplexers. The design of GINX/CGGI bootstrapping is based on the use of binary secret keys, since the secret key bit s_i is used as a selector between two possible messages ACC_{i-1} and $\mathsf{ACC}_{i-1} + X^{\bar{a}_i} \cdot \mathsf{ACC}_{i-1}$.

In [GINX16, CGGI16a], the decryption circuit is expressed as a sequence of homomorphic multiplexers, and the secret key bits are taken as `Mux` wire inputs, the LWE secret key is encrypted bit-by-bit (the first input bit determining the underlying if-then-else condition). It has been adapted to support ternary secret keys in [BIP+22, KDE+21] and any secret key distribution in [MP21, JP22]. Contrary to GINX/CGGI bootstrappings, FHEW [DM15] can support arbitrary secret key distributions, but in that case, the bootstrapping key includes additional decomposition of ciphertext components with respect to a radix basis parametrizing the bootstrapping procedure, which leads to a large amount of bootstrapping key material. Overall, the bootstrapping key in [CGGI16a] is about one hundred times smaller than that of [DM15] and the gate bootstrapping was reduced to 0.052 seconds, representing a twelve-fold speed-up over the procedure of [DM15].

Recent improvements. Other BlindRotation algorithms based on the NTRU problem [BIP+22, Klu22] and homomorphic automorphism evaluations [BDF18, LMK+23, WWL+24a] have also been proposed. Compared to the bootstrappings from [AP14, DM15], the automorphism-based method optimizes the number of keys used, but requires one keyswitching per automorphism evaluation. This number can be reduced [XZD+23, DKMS24] by combining automorphisms in the same loop [XZD+23, DKMS24, BJ25]. Recently, the authors of [BJ25] further optimizes the number of keyswitching and external products, reducing the overall time of the automorphism-based bootstrapping. Improvements on the bootstrapping in [LMSS23, BCL+24] had also been proposed by using block binary keys where specific positions of the secret key are filled with zero values.

2.3.3 Automorphism-based BlindRotation

In this paragraph, we provide a high-level overview of the automorphism-based variant of the BlindRotation introduced in [LMK+23].

Homomorphic automorphism. An automorphism ϕ_α, for α odd, applied to an RLWE ciphertext encrypting $\mu(X)$ under a secret key $\tilde{s}(X)$ maps it to an RLWE ciphertext encrypting $\mu(X^\alpha)$ under the secret key $\phi(\tilde{s})$. Then, by applying a keyswitching using $\mathsf{ksk}_{\tilde{s}(X^\alpha)\mapsto\tilde{s}(X)}$ as the keyswitching key gives an RLWE ciphertext, encrypting $\mu(X^\alpha)$ under secret key $\tilde{s}(X)$. This operation will be written as HomAut_α. If $\alpha = (-1)^{k_0}(5)^{k_1} \mod 2N$, with k_0, k_1 explicit, HomAut_α will be denoted by $\mathsf{HomAut}_{(k_0,k_1)}$. Notice that each automorphism requires a keyswitching operation.

The automorphism-based BlindRotation is based on the observation that $\mathbb{Z}_{2N}^\star \cong \mathbb{Z}_{N/2} \otimes \mathbb{Z}_2$. As the integers 5 and -1 are generators of order $N/2$ and 2 respectively, they both span $\mathbb{Z}_{2N}^*$. Setting $I_\ell^+ := \{i \mid a_i = g^\ell\}$ and $I_\ell^- := \{i \mid a_i = -g^\ell\}$ for $\ell \in \left[0, \frac{N}{2}\right[$.

We recall that the goal is to bootstrapp an LWE ciphertext $(\mathbf{a}, b)$, whose components are already rounded in $\mathbb{Z}_{2N}$. The accumulator is initialized as $\mathsf{ACC}_0 := \left(0, X^{-gb} \cdot v(X^{-g})\right)$ and the bootstrapping key $\mathrm{BK} = (\mathrm{BK}_i)_{0\le i<n}$, with $\mathrm{BK}_i := \mathsf{RGSW}_{\tilde{s}}(X^{s_i})$, and keyswitching keys $\mathsf{ksk}_{\tilde{s}(X^g)\mapsto\tilde{s}(X)}$, $\mathsf{ksk}_{\tilde{s}(X^{-g})\mapsto\tilde{s}(X)}$ for the two automorphisms $X \mapsto X^g$ and $X \mapsto X^{-g}$ respectively.

The automorphism-based BlindRotation relies on the fact that $\sum_i a_i s_i$ can be written as:

$$\left(\sum_{j\in I_0^+} s_j + g\left(\cdots + g\left(\sum_{j\in I_{N/2-1}^+} s_j - g\left(\sum_{j\in I_0^-} s_j + g\left(\cdots + g\left(\sum_{j\in I_{N/2-1}^-}\right)\right)\right)\right)\right)\right)$$

The a_j's are first grouped according to the indices I_j. The BlindRotation first performs an external product between ACC_0 and BK_j for $j \in I_{N/2-1}$ and then applies HomAut_g. The process is repeated:

- for each set of indices I_j^- for $j > 0$, the external product is followed by an evaluation by HomAut_g. And for $j \in I_0^-$, the external product is followed by an evaluation by HomAut_{-g} and;
- for each set of indices I_j^+ for $j > 0$, the external product is followed by an evaluation by HomAut_g.

As automorphisms do exist for odd exponents in $\mathbb{Z}_{2N}$, a solution proposed by the authors consists in rescaling the a_i's by performing the modswitching by rounding to the nearest odd values.

2.4 Circuit bootstrapping

The output of a homomorphic multiplexer is an RLWE ciphertext and if multiplexers are to be connected, the output of a multiplexer cannot be used directly as a selector for a subsequent homomorphic multiplexer. Unfortunately, the previous GSW-based bootstrapping as implemented in [GSW13, AP14, GINX16] does not provide practical solutions. In [CGGI17], we introduced the *circuit bootstrapping* which allows to convert an R/LWE ciphertext with high-noise into an RGSW ciphertext of the same message with reducing the noise. The circuit bootstrapping makes it possible to connect gates whose outputs are of different types, i.e., connecting a cMux gate output, implemented as an RLWE ciphertext into a cMux selector, implemented as an RGSW ciphertexts. Initially, the *circuit bootstrapping* proposed in [CGGI17] had a runtime of 137 ms. Subsequent works [CLOT21, WWL+24b] have significantly improved it. We will first present the original *circuit bootstrapping* algorithm and discuss the optimized algorithms in Section 3.3.1 of chapter 2 as they rely on variants of the bootstrapping that will be presented next. The *circuit bootstrapping* works across three levels level_i of error noise for $i \in [0, 2]$, each of variance ϑ_i. Each level is associated with a set of homomorphic operations. The level 0 corresponds to large noise ciphertexts and tolerates a limited number of linear operations. The ciphertexts at level 1 have medium size parameters and support a relatively large number of leveled operations. Level 2 is associated to ciphertexts with small noise parameters and there is a very little margin for more homomorphic operations, as this level almost reaches the limit of what can be handled with native 64-bit arithmetic. The *circuit bootstrapping* workflow proposed in [CGGI17] is be summarized in the following. We refer the reader to [CGGI17, Section 4.1] for the proof of correctness of the circuit bootstrapping procedure.

1. The first step takes an LWE ciphertext of $\mu \in \{0, \frac{1}{2}\}$ and switches it to a level 0 ciphertext using keyswitching.
2. The second step performs ℓ gate bootstrappings to convert a ciphertext at level 0 in $\mathcal{C}_\mu$ into ℓ ciphertexts at level 2 in $\mathcal{C}_{\frac{\mu}{B^i}}$ for $i \in [1, \ell]$.
3. The last step applies $\mathsf{PrivKeySwitch}^{f_0}_{\overline{\text{SK}}\mapsto\text{SK}}$ and $\mathsf{PrivKeySwitch}^{f_1}_{\overline{\text{SK}}\mapsto\text{SK}}$ to each ciphertext in $\mathcal{C}_{\frac{\mu}{B^i}}$ to convert them into one RGSW ciphertext of μ at level 1, where $f_i(x) := K_i x$ and

$(K_0, K_1) \in R^2$ is the secret key of the RGSW encryption scheme used to generate the bootstrapping key.

2.5 Functional bootstrapping

FHEW-like bootstrappings and automorphism-based variants support *functional* or *programmable* bootstrapping, which means that an arbitrary function can be evaluated on encrypted data using one bootstrapping call. The term functional bootstrapping (resp. programmable) was introduced in [BGGJ20] (resp. [CJP21]) but the idea of evaluating non-binary gates was already suggested in [DM15, Section 7], further investigated in [BR15, CIM19, BGGJ20].

Since then, many improvements on the efficiency and precision of functional bootstrapping (extending domain size) have been proposed, firstly for specific functions
[BST20] (for comparison) and [BMMP18, ISZ19, LMP22] (for sign function) and later on, for arbitrary functions [GBA21, CLOT21, YXS+21, KS23, MHW+24]. [3]. The authors of [LY23] improved the functional bootstrapping by enlarging the input domain by introducing a polynomial morphism to a larger ring. Recently, the authors of [BORLT25] have further improved the extended bootstrapping from [LY23] by reducing the number of external products and refining the modulus switching step. Recent works shows that the functional bootstrapping also applies to other FHE schemes than the FHEW-like bootstrapping, in particular to the CKKS scheme [AKP25, BKSS24].

The regular bootstrapping relies on two consecutive steps: a linear step, which homomorphically recovers $\bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle$ in the exponent, i.e., as $X^{\bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle}$ and a non-linear step, which homomorphically rounds the noisy encoding of the message by homomorphically extracting the constant coefficient of $v(X) \cdot X^{\bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle}$. The gate bootstrapping can be adapted to support arbitrary functions evaluation defined over a discretized space by modifying the second step as follows:

if $f : \hat{\mathbb{Z}}_p \mapsto \hat{\mathbb{Z}}_p$ is a function defined in the rational torus $\hat{\mathbb{Z}}_p$, then setting the test vector $v(X)$ such that $v_j := f\left(\frac{\lfloor \frac{pj}{2N} + \frac{1}{2p} \rfloor \mod p}{p}\right)$.

Substituting j with $\mu^\star := \bar{b} - \langle \bar{\mathbf{a}}, \mathbf{s} \rangle \mod 2N$, the constant term $v(X) \cdot X^{\mu^\star}$ becomes $f(\mu)$ if the error drift noise remains small enough and if $0 \leq \mu^\star < N$, as a result of the anticyclic property of $\hat{R}_p$. In that case, the plaintext space is halved. This technique is named *padding* in [CLOT21] as this comes down to set the most significant bit of the message equal to 0. The negacylclic property of $\hat{R}_p$ can be expressed differently in the case where $p \| q$, $q = 2^k$ (or more generally when p is even). If the function f verifies $f(\mu + \frac{1}{2}) = -f(\mu)$ then the function f can be defined over the full input domain. The *sign* function verifies this property.

Remark 6. *If $f : \mathrm{dom}_f \mapsto \mathrm{im}_f$ is not defined over $\hat{\mathbb{Z}}_p$, an additional encoding and decoding functions will be used $f = \mathsf{decode} \circ \tilde{f} \circ \mathsf{encode}$ with $\mathsf{encode} : \mathrm{dom}_f \to \hat{\mathbb{Z}}_q$, $\tilde{f} : \hat{\mathbb{Z}}_q \to \hat{\mathbb{Z}}_q$ and $\mathsf{decode} : \hat{\mathbb{Z}}_q \to \mathrm{dom}_f$.*

[3] Given the extensive body of literature on FHEW-based functional bootstrapping, we have selected a subset of representative works.

3 Optimizing Homomorphic Evaluation

This chapter presents our contributions to the optimization of homomorphic evaluation of arbitrary functions. We review generic representations of a Boolean function and show how it can be evaluated without bootstrapping, using a sequence of homomorphic multiplexers. We first show to homomorphically evaluate a Binary Decision Tree and a Binary Decision Diagram (BDD), and based on the correspondence between BDD and deterministic finite automata, we further show how Deterministic Automata can be evaluated homomorphically. We extend this technique to the homomorphic evaluation of arithmetic functions by encoding the value into the weights of a Weighted Deterministic Automata.

This chapter introduces amortization techniques for the homomorphic evaluation of arbitrary functions. It presents a technique called multi-value bootstrapping which amortizes the cost of the BlindRotation among different calls of the functional bootstrapping with different functions. We then review packing techniques and their use for the homomorphic evaluation of arbitrary functions. This chapter is associated with publications [GINX16, CGGI16a, CGGI17, CIM19, BI25a].

3.1 Boolean function evaluation

3.1.1 Boolean functions in Disjunctive Normal Form

In this section, we review computational models for the homomorphic evaluation of an arbitrary boolean function $f : \{0,1\}^k \mapsto \{0,1\}$. It has been demonstrated in [AP14] that when a chain of conjunctions is right-associated, with one of the two inputs encrypted as a fresh GSW ciphertext, then the noise grows linearly in the number of conjunctions instead of exponentially. In order to homomorphically evaluate an arbitrary Boolean function, we can rely on its disjunctive normal form (DNF), i.e., by expressing f as a disjoint union of conjunctive terms. f can then be evaluated by homomorphically adding all the encrypted terms, where each term formula is expressed using a chain of AND gates. However, in that case, the resulting noise is exponential in the number of terms in the DNF. In [GINX16], we showed that arbitrary Boolean functions can be evaluated homomorphically in a trivial way using a chain of multiplexer gates instead. In that case, the noise growth is proportional to the square root of the number of inputs.

3.1.2 Binary Decision Diagram representation

The truth table of f can be written as a vector of length 2^k such that $T_j := f(x_0, \cdots, x_{k-1})$, with $j = \sum_{i=0}^{k-1} x_i 2^{k-i-1}$. The full binary decision diagram (BDD) of f can be viewed as a binary tree of multiplexers, of depth k. The bottom level consists of 2^k leaves, each denoted $X_{k,j} := T_j$ for $j \in [0, 2^k - 1]$. The i-th iteration of the homomorphic evaluation can then be implemented using the `Mux` gate as follows: there are 2^i nodes computed as $X_{i,j} := \mathsf{cMux}(\mu_i, X_{i+1,2j+1}, X_{i+1,2j})$ and if $i = 0$, the node $X_{0,0}$ contains $f(\mu_0, \cdots, \mu_{k-1})$. The noise of the output ciphertext is linear in the number of inputs. As the number of nodes is exponential in the number of inputs, the time complexity remains high for large domain Boolean functions. In some cases, identical subtrees can be merged and the full BDD can be

homomorphically evaluated with $\mathcal{N}(f)$ homomorphic cMux gates, where $\mathcal{N}(f)$ is equal to the number of distinct subtrees in the full BDD of f. The BDD associated with the evaluation of the Boolean function f corresponds to deterministic automata that recognize the mirror language of the language of f. With this correspondence, Nerode's partitioning algorithm can then be used to determine the identical subtrees.

3.1.3 Using Deterministic Automata

Following the previous results, we showed in [GINX16] that we are able to evaluate any deterministic finite automata (DFA) using less than $|w|\#Q$ homomorphic multiplexer gates, where $|w|$ is the size of the input word and Q is the number of states. We additionally showed in [CGGI16a] that this number can be reduced by a factor $|w|$ when all valid paths in the automaton, from the initial state to any accepting states have the same length. The method of [GINX16] makes use of RGSW internal products which are replaced with $\mathsf{RLWE\text{-}RGSW}$ external products in [CGGI16a]. We obtained the following theorem proved in [CGGI16a]:

Theorem 1 (Evaluating Deterministic Automata)**.** *Let $\mathcal{A} = (Q, i, T_0, T_1, F)$ be a deterministic automata, where Q is the set of states, $i \in Q$ denotes the initial states, T_0, T_1 are the two transition functions and F is the set of final states. Given s valid RGSW ciphertexts $c_1, \ldots, c_s$ that encrypt bits of a word $\mathbf{w} \in \mathbb{B}^s$, with error noise variance $\vartheta = \max_j \mathsf{Var}(\mathsf{Err}(c_j))$, by evaluating at most $\leq s\#Q$ cMux gates, one can produce an RLWE ciphertext $\mathbf{d}$ that encrypts $\frac{1}{2}$ iff $\mathcal{A}$ accepts $\mathbf{w}$, and 0 otherwise. Assuming the Gaussian heuristic, $\mathsf{Var}(\mathsf{Err}(\mathbf{d})) \leq 2 \cdot (2\ell N\beta^2\vartheta + (N+1)\epsilon^2)$, where ℓ, β, ϵ are decomposition parameters defined in Section 2.2 and N is the ring degree. Furthermore, the number of evaluated cMux can be decreased to $\leq \#Q$. if $\mathcal{A}$ satisfies either of the conditions:*
(i) for all $q \in Q$ (except KO states), all words that connect i to q have the same length;
(ii) $\mathcal{A}$ only accepts words of the same length.

The correctness comes from the following. We initialize $\#Q$ noiseless ciphertexts for each state in Q, encrypting 1 if $q \in F$ and 0 otherwise. Then for each letter of the word encrypted as c_j, the transition is computed as follows: for all $q \in Q$ and $j \in [0, s-1]$, $\mathbf{d}_{q,j-1} := \mathsf{cMux}(c_j, d_{T_1(q),j}, d_{T_0(q),j})$. And finally we output $d_{i,0}$. The automata of this construction are used to evaluate rational Boolean functions in binary words where the image of $(w_0, \cdots, w_{s-1}) \in \{0,1\}^s$ is equal to 1 iff $T_{w_{s-1}}(T_{w_{s-2}}(\cdots(T_{w_1}(i)))) \in F$, and 0 otherwise.

3.2 Using Weighted Deterministic Automata

In [CGGI17], we extended the previous result to handle the homomorphic evaluation of arithmetic operations over binary words more efficiently, by using deterministic weighted finite automata (DWFA). Weighted finite automation are finite automata with a weight added in each transition. The weights serve as an implicit memory and are incrementally added together along the evaluation so that the result can be obtained directly. We refer to [DG09, BGW00] for a formal definition of Weighted Deterministic Automata (WFA).

When using determinsic automata, the output is one bit of information, i.e., 1 if the word is accepted and 0 if rejected. In order to obtain several bits of information as output, as expected after the evaluation of arithmetic operation, a straightforward approach would be to

use a DFA for each output bit of the function. However, the number of possible outputs would be exponential in the size of the inputs. Using DWFA overcomes this blow-up by integrating an implicit memory into the evaluation process.

In this section, we provide an intuition on how the evaluation is processed and refer to [CGGI17, Section 3.2] for more details. WFA are defined over a semi ring $(S, \oplus, \otimes, 0, 1)$, and each transition is labeled by a weight in S. The two laws are used in the following way: (a) when a word is read, the weights along the path are multiplied; (b) if several paths lead to the same word, their paths are summed. As we would like to support homomorphic evaluation, the class of WFA is restricted to deterministic WFA (DWFA), i.e., there is one possible path per word. In addition, we add the condition that all the words are accepted. With these two conditions, the addition law $\oplus$ becomes useless in our case as (b) defined above is never invoked. We use $(S, \otimes) = (\hat{\mathbb{Z}}_q[X], +)$ as the semi ring structure for the DWFA. In [CGGI17], we showed how to use DWFA to homomorphically evaluate a set of arithmetic functions, including multi-addition, multiplication and comparison.

3.3 Amortized Evaluation

3.3.1 Using the multi-value bootstrapping

In [CIM19], we introduced *multi-value bootstrapping* which allows one to evaluate many arbitrary functions on the same ciphertext(s) using one BlindRotation. As mentioned in the previous chapter, when using functional bootstrapping to evaluate an arbitrary function, the function is encoded within the coefficients of the test vector. The idea of the multi-value consists then of defining as many test vectors as there are arbitrary functions, and factorizing the computation by merging the redundant parts. However, if this factorization is implemented naively, the noise growth may increase significantly. The decomposition we choose relies on the observation that the following equation holds in the cyclotomic ring $\mathbb{Z}_N[X]$: $(1 + X + X^2 + \cdots + X^{N-1})(1 - X) = 2 \mod X^N + 1$.

Given many functions $f_i : \hat{\mathbb{Z}}_p \mapsto \hat{\mathbb{Z}}_p$, the test vector $\mathsf{TV}_i(X)$ associated with f_i can be written as $\mathsf{TV}_i(X) = v_0(X)v_i(X)$, where $v_0(X)$ is the polynomial factor shared among all $\mathsf{TV}_i(X)$.

Taking $v_0(X) := \frac{1}{2}\tau(1 + X + \cdots + X^{N-1})$ and $v_i(X) := (1 - X)\tau^{-1}\mathsf{TV}_i$, where τ is a scaling factor set as the least common divisors among the coefficients of all $\mathsf{TV}_i(X)$'s, we have $\mathsf{TV}_i(X) = v_0(X) \cdot v_i(X)$.

The error growth of the multi-value bootstrapping can be expressed as $\|v_i\|_\infty^2 \vartheta_{\mathsf{boot}}$, where $\vartheta_{\mathsf{boot}}$ denotes the error variance of the noise induced by the regular bootstrapping procedure. We refer the reader to Theorem 3.3 and its proof [CIM19] for the argument concerning noise propagation in the multi-value bootstrapping. We give some application examples of the multi-value bootstrapping in the following paragraph.

3.3.2 Applications of the multi-value bootstrapping

- *Optimized arbitrary functions:* assume that f is a *look-up table* $f : \{0,1\}^r \mapsto \{0,1\}^s$, and $\phi_j(x_1, \cdots, x_r)$ is defined as the j-th bit of $f(x_1, \cdots, x_r)$ for $j = 1, \cdots, s$. Then the function f can be homomorphically evaluated by evaluating each ϕ_j independently, by using the functional bootstrapping s times. One can make use of tmulti-value

bootstrapping instead, as the functions ϕ_j, $j = 1, \cdots, s$ are evaluated on the same ciphertexts, encrypting $x_1, \cdots, x_r$, which amounts to one BlindRotation instead of s using the previous method;

- *Optimized tree-based method for the evaluation of an arbitrary function over multiple ciphertexts:* to mitigate the increase in N for large look-up table (LUT) evaluation, the authors of [GBA21] have introduced two methods, the *tree-based* method and the *chaining* method, which combine several look-up tables to evaluate a larger one. We recall the first one. Let $f : B^d \mapsto B$ be a LUT that takes as input $c_0, \cdots, c_{d-1}$ LWE ciphertexts encrypting $m_0, \cdots, m_{d-1} \in \mathbb{Z}_B$ with $m := \sum_{i=0}^{d-1} m_i B^i$. The evaluation of f amounts to the evaluation of the B^{d-1} functions $f_i : \mathbb{Z}_B \mapsto \mathbb{Z}_B$ defined as $f_i(x) = f(i + xB^{d-1})$ for $i \in [1, B^{d-1}]$. The tree-based method first defines B^{d-1} test vectors, each associated with f_i and then performs B^{d-1} functional bootstrappings with the same LWE sample c_{d-1} as input. The B^{d-1} LWE output encrypts $f_i(i + m_{d_1} B^{d-1})$ for $i \in [1, B^{d-1}]$. If $d = 1$ the process ends, otherwise they are combined as B^{d-2} RLWE test vectors using the public functional keyswitching [CGGI17], which was recalled in Chapter 1, Section. Using multi-value bootstrapping instead of regular bootstrapping, the number of BlindRotations could be decreased from exponential in d to linear in d. In that case, the test vector would have to be given in cleartext, or the RLWE test vector ciphertexts would need to be converted into RGSW ciphertexts using circuit bootstrapping to perform the product with the accumulators. Therefore, as explained in [GBA21, Section 3.1], the multi-value bootstrapping is used only at the first step in this context. This optimization has been used in [TCBS23, BBB+25] to speed-up the evaluation of AES in the TFHE framework, more specifically for the evaluation of the Sbox and the multiplications that occur in the MixColumns step.

- *Optimized circuit bootstrapping:* in the second step of the circuit bootstrapping, recalled in Section 2.4, ℓ calls to the bootstrapping take the same LWE ciphertext encrypting μ, where each of them evaluates $f_i : x \mapsto \frac{x}{B^i}$ for $i \in [1, \ell]$. In [CIM19, Section 4.4], we suggest using multi-value bootstrapping so that, instead of calling ℓ times the BlindRotation, only one call to the BlindRotation is made. The idea of amortizing the cost of the BlindRotation was also used and implemented in [CLOT21], where multi-value bootstrapping is replaced with a different technique, called PBSmanyLUT.

  In their solution, the parameters are set a priori and determined how many functions could be evaluated using one bootstrapping call. In the end, contrary to the multi-value-based optimization, the noise growth does not depend on the size of the test vectors, but in that case, the parameters need to support the evaluation of the different LUTs. So both optimizations are working with different trade-offs.

  The multi-value bootstrapping was also used in the optimized circuit bootstrapping construction recently proposed by Wang *et al.* [WWL+24b]. They proposed a new workflow which includes the multi-value bootstrapping without the sample extraction, which is then combined with a homomorphic trace computation and a scheme switching step. We refer to [WWL+24b] for a full description of the workflow. Their report timings show that their circuit bootstrapping algorithm outperforms ours described in Section 2.4 with almost a 10x speedup and 16x key reduction.

3.3.3 Using packing methods

In this section, we review different packing techniques and show how they can be used to optimize the homomorphic evaluation of arbitrary functions. We will further detail our contributions based on the colum method and the row method. Packing refers to the operation of converting several LWE ciphertexts into one RLWE ciphertext and repacking refers to mapping multiple RLWE ciphertexts into a single RLWE ciphertext. We can assume that RLWE ciphertexts are given as LWE ciphertexts, since an RLWE can be efficiently converted into LWE ciphertexts by homomorphic sample extraction. An RLWE ciphertext encrypting $\sum_{i=0}^{N-1} \mu_i X^i$ can be viewed as a structured set C of N LWE ciphertexts $(\mathbf{a}_i, b_i) \in \hat{\mathbb{Z}}_q^N \times \hat{\mathbb{Z}}_q$ encrypting μ_i (up to a format-switching) under the same secret key $\mathbf{s}$. In the following, we review various packing techniques, where the shape of C can vary.

- *Column method:* In [CGGI17], we introduced the *public functional keyswitching*, which can be viewed as a packing technique by taking $f : \hat{\mathbb{Z}}_p^N \mapsto \hat{R}_p$ defined as $f(\mu_0, \cdots, \mu_{N-1}) = \sum_{j=0}^{N-1} \mu_j X^j$. The fact that f is 1-lipschitzian ensures that the noise does not grow too much. In addition, we can move the slots or remove some of them by adjusting f accordingly.

  The column packing method and the improved variants [CGGI17, MS18, BGGJ20, BCK+23] arrange the input LWE ciphertexts as $(\mathsf{coeff}(\mathbf{a}_i), b_i)$, vertically, where coeff is the coefficient embedding map. With this representation, the j-th column of C is $c_j = (a_{0,j}, \cdots, a_{N-1,j})$ for $j \in [0, N-1]$ and $c_{N-1} = (b_0, \cdots, b_{N-1})$. This technique reconstructs the RLWE encryption of $\sum_i \mu_i X^i$, performing a keyswitching from the secret key s_j to the new secret key for each column.

  How we used the column method to optimize look-up table evaluation: we proposed two approaches to optimize the homomorphic evaluation of an arbitrary Boolean function using this packing method, called *horizontal packing* and *vertical packing*. In order to evaluate $f : \{0,1\}^r \mapsto \{0,1\}^s$, a standard approach is to evaluate the s subfunctions that return each digit separately. The *horizontal packing* is based on the observation that, using the cMux-based approach (i.e. a sequence of multiplexer evaluation), each of the subfunction evaluates the same tree using the same selector. So, the idea is to process the evaluation from the left to the right of the BDD tree, but to pack the outputs of the s subfunctions' together. This reduces the total number of multiplexers by a factor s, i.e., the number of cMux gates becomes $\lceil \frac{s}{N} \rceil (2^d - 1)$. This method is beneficial when s is large, but in general $s \leq N$. The *vertical packing* packs the output of each f_j vertically. Assume $N = 2^\delta$ divides 2^s. The look-up table for f_j is a column of 2^s values packed as $\frac{2^s}{N}$ RLWE ciphertexts $d_0, \cdots, d_{2^{s-\delta}-1}$. To retrieve $f_j(x)$, one has to retrieve the ρ-th element (with $\rho = x \mod N$) of the p-th block (with $p = \lfloor \frac{x}{N} \rfloor$). One can use a cMux-tree to select the p-th block and the BlindRotation to rotate the coefficient of d_p by ρ positions and extract the constant coefficient, i.e., $\sigma_{j,x}$. The full algorithm is described in [CGGI17, Algorithm 5].

- *Row packing method:* the row packing introduced in [CDKS21] encodes each row of C as a polynomial ciphertext. But the format of the message is not preserved (except for the constant term scaled by a factor N) and some coefficients appear to be useless. To

remove them except the constant one, the homomorphic trace function can be used. They obtain $N\mu_i$, with μ_i the constant term of the i-th polynomial, i.e., $\mathsf{RLWE}(\mu_i + \star)$ where $\star$ stands for useless terms. The packing algorithm from [CDKS21] is an FFT-like recursive algorithm, which reduces the required key material from $O(N^2)$ to $O(N \log N)$ compared to [CGGI17, MS18]. With Jean-Philippe Bossuat, we recently proposed in [BI25a] an iterative variant of the row packing method and a detailed proof of its correctness. An appealing feature of this iterative variant is that it enables a streamable packing algorithm, which we present in the next paragraph.

How we used the row packing method to optimize arbitrary function evaluation: we considered a two-party scenario where a user would like to obtain the evaluation of $f : \text{dom} \mapsto \text{im}$ over a set of encrypted inputs $\{i_0, \cdots, i_{z-1}\} \subseteq \text{dom}(f)$ that the user would like to keep unknown to the server. The following step is applied for each encrypted input i in parallel. To simplify, we assume that the size of the subset z is less than N but larger sizes can be handled using the *split domain* approach we proposed in [IIMP22]. If the user wants to obtain the evaluation of f in i, it sends the RLWE encryption of X^i. The server precomputes a polynomial representation of the function f, $u_f(X)$ such that the plaintext multiplication with $\mathsf{RLWE}(X^i)$ gives the RLWE encryption of a polynomial, (i) whose constant coefficient is $f(i)$ and (ii) whose other coefficients are consecutive evaluations of the function on the other ($\leq N-1$) inputs in the domain; these other inputs are not targeted by the current query and are omitted from the response. In a second phase, the server applies a repacking procedure as follows: it takes as input the previous N RLWE ciphertexts whose constant coefficients are respectively $f(i_0), \cdots, f(i_{z-1})$. And it reconstructs a new RLWE ciphertext of the polynomial $f(i_0)X^0 + \cdots + f(i_{z-1})X^{z-1} + \star X^z + \cdots + \star X^{N-1}$, where $\star$ can be any element in the message space. Depending on the format agreed, an LWE encryption of $f(i)$ can be homomorphically extracted and sent back to the user or the repacked RLWE ciphertext is sent to the user who decrypts and parses the polynomial by herself in order to retrieve the targeted evaluations. The communication cost on the user side is $z \times 2N \log q$, while that on the server side is $2N \log q$. Hence, our method is benefical in terms of communication when the number of requested points is small, as the response size is the one that is most significantly reduced.

- *Diagonal packing method:* The diagonal method [HS14] views C according to its sub-diagonal. Each of them is multiplied component-wise by the rotated version of the secret key. Summing all intermediate products gives the product between C and the secret key. The diagonal method was extended in [jLHH+21] to support large domain LUT evaluations at the cost of some approximation errors in the result.

3.4 Low-memory dynamic packing

As this variant of the row packing is not yet published, we provide the full description of the algorithm in this section. This is a joint work with Jean-Philippe Bossuat. The goal is to merge $d \leq N/n$ RLWE ciphertexts of degree N of some polynomial messages $\mu_i(Y) = \sum_{j=0}^{n-1} \mu_{i,j} Y^j$ for $Y = X^{N/n}$ and $i \in [0, d-1]$, and to produce a single RLWE ciphertext whose polynomial message is $\mu(X) = \sum_{i=0}^{d-1} \left(\sum_{j=0}^{n-1} \mu_{i,j} Y^j \right) X^i$.

For sake of simplicity, we take $n = 1$ so that $Y = X$ and the number of input ciphertexts equals $d = N$, the degree of the polynomial ring. However, the algorithm easily generalizes to any Y and any number d of inputs by taking zero values for the remaining coefficients if $d \leq N/n$ and by splitting the d ciphertexts into batches of up to N/n ciphertexts and evaluating one repacking per batch if $d > N/n$.

In order to specify how the ciphertext ordering is modified by the dynamic repacking algorithm, we introduce ciphertext hashmap which is aligned with the index-permutation mappings appearing in FFT-like algorithms.

Definition 2 (Ciphertext hashmap)**.** *A ciphertext hashmap is a list of ciphertexts in which each ciphertext is associated with a unique index in* $[0, N-1]$*. The elements of a hashmap can be reindexed by applying a* permutation *over* $[0, N-1]$*. We denote by* $\mathsf{rev}_N(i)$ *the bit-reversal permutation of* $i \in [0, N-1]$ *on* $\log(N)$ *bits.*

We also introduce a register, indexed by an integer i and associated with a state consisting of a ciphertext and a boolean flag. The state serves as a temporary memory to track updates or operations on the corresponding ciphertext in the hashmap during dynamic repacking.

Definition 3 (Register)**.** *A register* REG_i *is associated with an index* i*, and a state* (c_i, b_i)*, where* c_i *is an* RLWE *ciphertext and* b_i *is a boolean flag. An empty ciphertext is denoted by* $\emptyset$*. We define three methods associated to a register:*

- $\mathsf{Init}(i)$*:* $(c_i, b_i) \leftarrow (\emptyset, 0)$
- $\mathsf{combine}(c, c', i)$*:*

$$\begin{cases} c \leftarrow c + c' * X^t + \mathsf{HomAut}_\alpha(c - c' * X^t) & \text{if } c \neq \emptyset \text{ and } c' \neq \emptyset \\ c \leftarrow c + \mathsf{HomAut}_\alpha(c) & \text{if } c \neq \emptyset \text{ and } c' = \emptyset \\ c \leftarrow (c' - \mathsf{HomAut}_\alpha(c')) * X^t & \text{if } c = \emptyset \text{ and } c' \neq \emptyset \\ c \leftarrow \emptyset & \text{if } c = \emptyset \text{ and } c' = \emptyset \end{cases}$$

  where

$$t \leftarrow N/2^{i+1} \text{ and } \alpha \leftarrow \begin{cases} (1, 0) & \text{if } i = 0 \\ (0, 2^{i-1}) & \text{otherwise} \end{cases}$$

- $\mathsf{Add}(i, c)$*:*

$$\begin{cases} (c_i, b_i) \leftarrow (c, 1) & \text{if } b_i = 0 \\ \\ \mathsf{Add}(i+1, \mathsf{combine}(c_i, c, i)) & \text{if } b_i = 1 \\ (c_i, b_i) \leftarrow (\emptyset, 0) \end{cases}$$

We first precent in Algorithm 2 the iterative ring repacking algorithm variant of [CDKS21] using the previous notation, whose proof is detailed in [BI25a].

We then describe the low-memory packing that allows the packing of $d = N/n$ RWLE encryptions into one RLWE encryption using $\log(N)$ registers and at most $\log N$ keyswitching keys. An input ciphertext is taken in a bit-reversed ordering with respect to its assigned

Algorithm 2 RLWE REPACKING

Inputs: a hashmap h containing up to N/n RLWE ciphertexts with unique indexes in the range $[0, N/n)$, where $\mathsf{h}[i]$ encrypts $\mu_i(Y) = \sum_{j=0}^{n-1} \mu_{i,j} Y^j$ for $Y = X^{N/n}$.
Output: an RLWE encryption of $\mu(X) = \sum_{i=0}^{N/n-1} \left(\sum_{j=0}^{n-1} \mu_{i,j} X^{jN/n}\right) X^i$.

for $i \leftarrow 0$ **to** $N-1$ **do**
    if $h[i] \neq \emptyset$ **do**
        $h[i] \leftarrow (N/n)^{-1} \cdot c[i]$

for $i \leftarrow \log(n)$ **to** $\log(N)-1$ **do**
    $t_i \leftarrow N/2^{i+1}$
    if $i = 0$
        $\alpha \leftarrow (1, 0)$
    else
        $\alpha \leftarrow (0, 2^{i-1})$
    for $j \leftarrow 0$ **to** $t_i - 1$
        $\mathsf{h}[j] \leftarrow \mathsf{h}[j] + \mathsf{h}[j+t_i] * X^{t_i} + \mathsf{HomAut}_\alpha(\mathsf{h}[j] - h[j+t_i] * X^{t_i})$
    $i \leftarrow i + 1$
return $\mathsf{h}[0]$

position, so that, in the end, its constant coefficient is placed at the index corresponding to its input position. A permutation can additionally be taken as input if a different ordering is required.

Lemma 1. *Let $c_0, \ldots, c_{N-1}$ be* RLWE *encryptions of $\mu_0(X), \cdots, \mu_{N-1}(X)$, respectively. On input the c_i's, Algorithm 3 returns an* RLWE *encryption of $\mu(X) = \sum_i \mu_i(X)[0] \cdot X^i$, where $\mu_i(X)[j]$ denotes the j-th coefficient of the i-th ciphertext.*

Proof. We will show that Algorithm 3 calls combine on the same sequences of ciphertext pairs as Algorithm 2 when Algorithm 2 inputs are taken in bit-reverse ordering. We consider the intermediate calls to combine() along Algorithm 3 run and we denote $\mathsf{h}^{(i)}$ the hashmap which contains the output of all calls to $\mathsf{combine}(\star, \star, i)$. The proof is based on the two following observations:

1. REG_i calls $\mathsf{combine}(\star, \star, i)$ $\frac{N}{2^{i+1}}$ times;
2. We can prove by induction that for $i \in [0, \log N)$, the calls of REG_i to $\mathsf{combine}(\star, \star, i)$ are such that for all $j \in \left[0, t_i := \frac{N}{2^{i+1}}\right)$:

$$\mathsf{h}^{(i)}[j] = \mathsf{combine}(\mathsf{h}^{(i-1)}[2j+1], \mathsf{h}^{(i-1)}[2j], i); \tag{1}$$

and $\mathsf{h}^{(-1)}[\kappa] = \mathsf{h}[\mathsf{rev}(\kappa)]$.

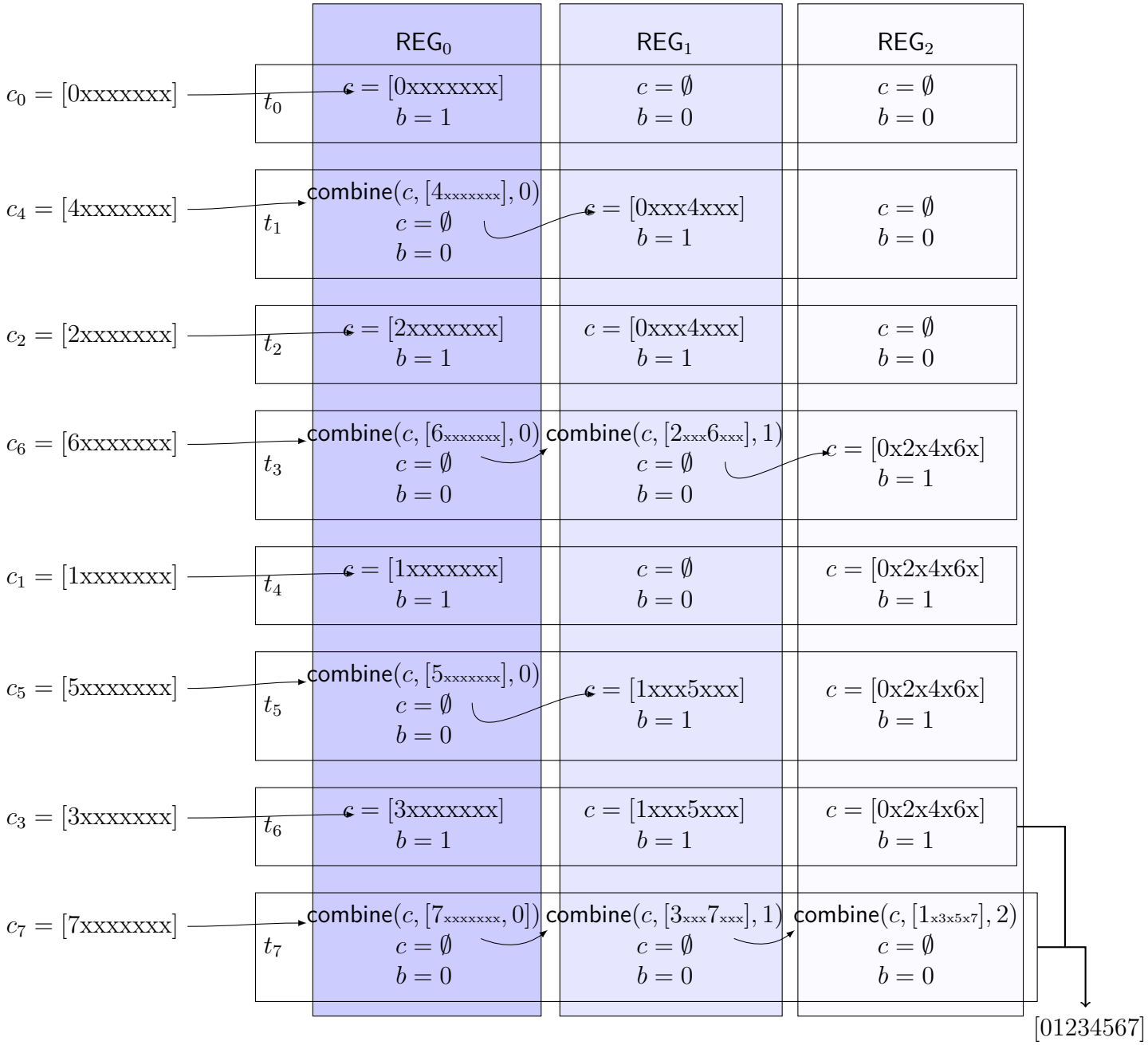


Figure 2: Illustrated example of Algorithm 3 for an input hashmap of $n = 8$ ciphertexts $(c_0, c_1, \ldots, c_7)$ which are RLWE encryptions of $\mu_i(X)$ with $\mu_i[0] = i$. The line t_i indicates the state values of each register associated to the i-th ciphertext. The initialization step creates the register pairs which are initialized to $(\emptyset, 0)$. At state 0, $c_0 = [0\text{xxxxxx}]$ is taken as input and REG_0 becomes actif $(b = 1)$. The next cipherext c_1 is then combined with c_0 (according to rule a). The result is sent to the next register and the local buffer is reinitisalized to $\emptyset$. When the last register (REG_2) calls combine, it outputs the final ciphertext [01234567].

Algorithm 3 RLWE Low-memory Dynamic Repacking

Inputs:
  h : hashmap of N/n ciphertexts, where $\mathsf{h}[i]$ encrypts $\mu_i(Y) = \sum_{j=0}^{n-1} \mu_{i,j} Y^j$ for $Y = X^{N/n}$
Output: an RLWE encryption of $\mu(X) = \sum_{i=0}^{d-1} \left(\sum_{j=0}^{n-1} \mu_{i,j} Y^j\right) X^i$
for $i = \log(n)$ **to** $\log(N) - 1$ **do**
  $\mathsf{Init}(i)$ ▷ creates (c_i, b_i)
end for
for $i = 0$ **to** $N/n - 1$ **do**
  if $\mathsf{h}[\mathsf{rev}_N(i)] \neq \emptyset$ **then**
    $\mathsf{Add}(0, (N/n)^{-1} \cdot \mathsf{h}[\mathsf{rev}_N(i)])$
  else
    $\mathsf{Add}(0, \emptyset)$
  end if
end for
return $c_{\log(N)-1}$

- For $i = 0$: REG_0 calls $\mathsf{combine}$ $\frac{N}{2}$ times since b_0 is initialized to 0 and switched to 1 one half of the time. Thus, we have, for $j \in \left[0, \frac{N}{2}\right)$:

$$\begin{aligned}\mathsf{h}^{(0)}[j] =& \mathsf{combine}(\mathsf{h}^{(-1)}[2j], \mathsf{h}^{(-1)}[2j+1], 0) \text{ if } b_0 = 1\\ =& \mathsf{combine}(\mathsf{h}[\mathsf{rev}(2j)], \mathsf{h}[\mathsf{rev}(2j+1)], 0)\end{aligned}$$

- By induction, $\mathsf{Add}(i-1, \star)$ invokes $\mathsf{combine}$ exactly $\frac{N}{2^i}$ times. Moreover, for each of these invocations, since combinations occur between two outputs produced by REG_{i-1}, REG_i calls $\mathsf{combine}$ $\frac{N}{2^{i+1}}$ times.

  By construction, given $c := \mathsf{combine}(\mathsf{h}^{(i-1)}[2j], \mathsf{h}^{(i-1)}[2j+1], i-1)$ for some $j \in [0, \frac{N}{2^i})$, the number of calls from REG_i to $\mathsf{combine}$ is half less than the ones from REG_{i-1} and we have for all $j \in \left[0, t_i := \frac{N}{2^{i+1}}\right)$:

$$\begin{aligned}\mathsf{h}^{(i)}[j] &= \mathsf{combine}(\mathsf{h}^{(i-1)}[2j], \mathsf{h}^{(i-1)}[2j+1], i)\\ &= \bar{\mathsf{h}}^{(i-1)}[2j] + \bar{\mathsf{h}}^{(i-1)}[j+t_i] * X^{t_i} + \textsc{Auto}_\alpha(\bar{\mathsf{h}}^{(i-1)}[j] - \bar{\mathsf{h}}^{(i-1)}[j+t_i] * X^{t_i})\end{aligned}$$

  where $\bar{\mathsf{h}}^{(i-1)}$ is the hashmap in step i in line 15 of Algorithm 2 as $\mathsf{h}^{(i-1)}[2j+1] \neq \emptyset$ and $\mathsf{h}^{(i-1)}[2j+1] \neq \emptyset$ when $\mathsf{Add}(i, c)$ is invoked.

The output of Algorithm 3 is $c_{\log(n)-1} := \mathsf{h}^{\log -1}[0]$, which is the same as Algorithm 2.

□

The complexity and noise analysis of Algorithm 3 is exactly the same as that of Algorithm 2. Algorithm 3 needs to be modified to be parallelisable, and one instance of Algorithm 3 must be instantiated for each thread, whereas in Algorithm 2, each pair of ciphertexts can be combined independently.

	Complexity	Memory	Noise	Streamable	Parallelisable
Algorithm 2	$(N/n)-1$	N/n	$\vartheta_{\mathsf{ct}}+((N/n)-1)\vartheta_{\mathsf{ks}}$	N	
Algorithm 3	$(N/n)-1$	$\log(N/n)$	$\vartheta_{\mathsf{ct}}+((N/n)-1)\vartheta_{\mathsf{ks}}$	Y	Somewhat

Table 1: Comparison between Algorithm 2 and Algorithm 3.

4 Circuit Privacy

In the homomorphic operations considered so far, without any additional property of the FHE scheme, the user who received the evaluation result may learn information about the circuit that the sender evaluated on their data.

Circuit privacy ensures that the ciphertext returned after homomorphic evaluation does not contain information about the circuit. A related algorithm, called sanitization, allows one to destroy all information about how a ciphertext was obtained, ensuring that the circuit that was evaluated remains secret. In this chapter, we present our contribution [BI22], developed in collaboration with Florian Bourse, on circuit private FHE and sanitization of FHE ciphertexts. In the next paragraph, we explain how these two notions are connected. We will present two approaches for circuit privacy and FHE sanitization that apply to the TFHE scheme, built by modifying its bootstrapping internally. Our strategy also applies to other FHEW-like schemes with slight adjustments. We first recall the definitions of circuit privacy and FHE sanitization, reviewing the prior works, and presenting our contributions. We then discuss concurrent works closely related to ours. Applications of FHE sanitization and related notions are presented in paragraph 4.5.

4.1 Circuit privacy and FHE sanitization definitions

Definition 4 (Circuit privacy)**.** *An FHE scheme is* circuit-private *for a class of functions $\mathcal{F}$ if the homomorphic evaluation of a function $f \in \mathcal{F}$ in encrypted messages does not leak more information than the evaluation result, even given the secret key, i.e. there exists a polynomial time simulator* Sim *such that the following property holds for any $f \in \mathcal{F}$:*

$$\begin{aligned}&\left(\mathsf{Sim}\left(1^\lambda, f(\mu_1, \cdots, \mu_t), (CT_1, \cdots, CT_t), EVK\right), (CT_1, \cdots, CT_t), SK\right)\\ &\approx_s \left(\mathsf{Eval}\left(EVK, f, (CT_1, \cdots, CT_t)\right), (CT_1, \cdots, CT_t), SK\right),\end{aligned}$$

where $CT_i \leftarrow \mathsf{Encrypt}(SK, \mu_i), (EVK, SK) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

Remark 7. *The above definition considers the joint probability distribution by including the secret key* SK*, which allows a distinguisher to exploit the knowledge of* SK*. In contrast, some of the prior definitions of circuit privacy consider only the distribution of the ciphertexts. In that case, the statistical distance must be proved to be small for any secret key* SK*. In particular, the notion of circuit privacy has been introduced by Gentry [Gen09b]. In the original definition, the property holds for any fixed secret key. Both definitions address circuit privacy against an adversary that knows the secret key. However, proving security for any possible* SK *requires the statistical distance to be small, even for maliciously generated secret keys. All our lemmata actually prove statements for any fixed* SK*, which provides stronger results. However, our results assume that the randomness in the ciphertexts* $(CT_1, \ldots, CT_t)$ *or in the evaluation key* EVK*, i.e., on the random coins of* KeyGen*, is honestly generated. Consequently, the property does not have to hold for any possible secret key* SK*.*

Remark 8 (Circuit privacy in the honest-but-curious *vs.* malicious settings)**.** *Our construction lies in the honest-but-curious model i.e., the ciphertexts and keys are sampled correctly. To*

support security in the malicious setting, one can use the techniques of [OPP14] to upgrade our scheme, by using a maliciously circuit-private (possibly non-compact) FHE scheme.

In the following definition of sanitization, as before, we assume that the keys are sampled correctly.

Sanitization of ciphertexts. Intuitively, the goal of a sanitization algorithm is to remove any information contained in a ciphertext beyond the message. A sanitization algorithm must satisfy two requirements: it must be *message-space preserving* and *sanitizing*, as defined below:

$\mathsf{Sanitize}(\text{EVK}, \text{CT})$ on input an evaluation key EVK and a ciphertext CT, it returns a ciphertext CT_s.

We say that $\mathsf{Sanitize}$ is *message-space preserving* if for any $\mu \in \mathcal{M}$, any ciphertext $\text{CT} \in \mathcal{C}_\mu$, $\mathsf{Sanitize}(\text{EVK}, \text{CT}) \in \mathcal{C}_\mu$ with overwhelming probability. We say that $\mathsf{Sanitize}$ is *sanitizing* if there exists a polynomial time simulator Sim such that for any $\mu \in \mathcal{M}$, any ciphertext $\text{CT} \in \mathcal{C}_\mu$, the following holds:

$$\Big(\mathsf{Sim}(1^\lambda, \mu, \text{EVK}), \text{SK}\Big) \approx_s (\mathsf{Sanitize}(\text{EVK}, \text{CT}), \text{SK}),$$

where EVK and SK are sampled honestly.

Note that given a message-space preserving sanitizing algorithm $\mathsf{Sanitize}$, it is easy to construct an FHE scheme that is circuit-private for all functions, by running $\mathsf{Sanitize}$ at the end of the evaluation procedure. In our paper [BI22], we also considered a relaxed variant of circuit privacy and sanitization, in which the required indistinguishability holds computationally rather than statistically.

4.2 Previous strategies

- *Noise flooding approach.* The first circuit-private technique, proposed in [Gen09a] relies on noise flooding. The idea is to add a fresh encryption of zero with a super-polynomially larger noise compared to the ciphertext noise. So adding a very large noise at the end of the computation drowns out the existing noise. This yields a 1-hop circuit-private (i.e., circuit privacy for one evaluation step) scheme for the class of all functions. The noise flooding technique requires starting with a modulus-to-noise ratio that is super-polynomial. Although this is not an issue for schemes like B/FV [Bra12, FV12], BGV [BGV12] or CKKS [CKKS17], noise flooding introduces an additional parameter blow-up for FHEW-like FHE schemes [DM15, CGGI16a], which makes use of smaller parameters.

- *Soak-and-spin approach.* A new approach, proposed in [DS16] avoids this downside and achieves FHE circuit privacy by relying on LWE with a polynomial modulus-to-noise ratio. Their sanitization strategy, called *soak and spin*, consists of iterating two consecutive steps: a $\mathsf{Refresh}$ algorithm which, given a ciphertext as input, reduces its noise to a fixed level by invoking the bootstrapping, and a second step ReRand injects additional noise into the ciphertext to make it closer to a canonical distribution. After λ consecutive iterations of $\mathsf{Refresh}$ and Rerand on any pair of input ciphertexts that

decrypt to the same message, the statistical distance between the two outputs is bounded by $2^{-\lambda}$. Any bootstrapping algorithm can be used as the Refresh procedure. For Rerand, a standard LWE public key containing $O(N \log q)$ encryptions of zero can be used. A random subset of these encryptions, together with a suitably small noise, is added to the ciphertext to rerandomize it.

- *Oblivious evaluation protocols.* Homomorphic encryption can also be used to construct low-communication two-party oblivious evaluation protocols. A folklore and generic technique to achieve circuit privacy in this context is to use noise flooding. Alternative strategies were proposed for efficiency reasons:

  - The authors of [dCJV21] proposed a two-party protocol in which a sender has many affine functions and the receiver learns the evaluation of these functions in x, which is kept private to the sender. Their circuit privacy approach relies on a divide-and-round step, which removes the ciphertext's dependence on the initial noise.
  - The authors of [dCKK$^+$24] proposed an oblivious linear evaluation algorithm for the BFV scheme, with a smaller noise overhead compared to the noise flooding technique. They observe that for a plaintext-ciphertext product of the form $a\mathbf{c}$ with $a \in R_t$, the correctness is preserved if a is replaced by a Gaussian in the coset $a + t\mathbb{Z}_t$. Then, adding a discrete Gaussian of a well-chosen parameter to $a\mathbf{c}$ allows them to prove circuit privacy for linear function evaluation. This technique has been recently extended in [HMS25] to handle not necessarily fresh ciphertext.
  - In [BdPMW16], Bourse *et al.* showed circuit privacy for NC^1 circuits without relying on the circular security assumption. However, their solution was not implemented. Their approach applies to the GSW scheme [GSW13] and its security relies on the plain LWE with polynomial modulus-to-noise ratio. Their construction makes use of a randomized decomposition of GSW ciphertexts to perform the homomorphic product instead of a deterministic decomposition and adds a random Gaussian after each homomorphic product, ensuring that the noise does not leak information about the computation being carried out.
  - *Modifying the bootstrapping internally.* [Klu24] introduced a statistical LWE sanitization procedure by modifying the FHEW-like bootstrapping restricted to binary secret keys. Recently, [HMS25] proposed an efficient sanitization procedure for TFHE, which also extends to other FHEW-like bootstrapping schemes. Since these two approaches are closely related to ours, we include a short comparison in the next paragraph.

4.3 Our strategy

In this paragraph, we outline the main ideas of the sanitization approach introduced in [BI22], which is a work I conducted in collaboration with Florian Bourse.

Randomized decomposition. Our construction relies on the use of a randomized decomposition [MP12, Kle00] instead of using a deterministic gadget decomposition. This allows us to have better control over the distribution when analyzing the homomorphic product noise. We

explain how randomized decomposition works and then how it is used for our circuit-private approach. Given a vector $\mathbf{v}$ and the gadget matrix $\mathbf{G}$, the deterministic decomposition produces a short vector in the coset $\Lambda^{\perp}_{\mathbf{v}} = \{\mathbf{u} \in \mathbb{Z}_q \mid \mathbf{uG} = \mathbf{v} \mod q\}$ while the randomized decomposition finds a discrete Gaussian sample in $\Lambda^{\perp}_{\mathbf{v}}$. We denote r its Gaussian parameter and $\vartheta_{\boxdot} = \frac{r^2}{2\pi q^2}$ its variance. In our work, we extend the decomposition from scalars to vectors of polynomials by viewing the gadget vector $\mathbf{g} = (1, B, \cdots, B^{\ell})$ as a ℓ-dimensional vector of polynomials. Sampling over $\Lambda^{\perp}_{\mathbf{g}}$ with $\mathbf{g} \in R^{\ell}$, then reduces to invoking N times the sampling algorithm over $\Lambda^{\perp}_{\mathbf{g}}$ with $\mathbf{g} \in \mathbb{Z}^{\ell}$ and to reconstructing the polynomials based on the output vectors. We denote $\mathbf{g}_r^{-1}$ (resp. $\mathbf{G}_r^{-1}$) the randomized decomposition with respect to $\mathbf{g}$ (resp. $\mathbf{G}$). LWE ciphertexts can be decomposed with a randomized algorithm into integer vectors in $R^{1\times 2\cdot\ell}$ with a spherical Gaussian distribution on a coset of $\Lambda^{\perp}(\mathbf{G})$. The randomized external product $\boxdot_r$ is defined accordingly.

Rerandomizer. Our procedure for randomizing an RLWE ciphertext relies on a randomized decomposition, and the addition of a randomizing term, which we called rerandomizer and denote it RERAND. We introduced two sanitizing strategies, each using a different rerandomizer that will be explicitly described below. We showed both randomizers satisfy the property of Lemma 2, which will be used to prove the correctness of our sanitization algorithm. In the following, for two random variables X and Y, $X \approx_s Y$ (resp. $X \approx_c Y$) means that X is statistically (resp. computationally close to Y.

Lemma 2. *For any* $\tilde{\mathbf{s}} \in \hat{R}_q^{\,d}$*, any* $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{\,(d+1)\cdot\ell}$ *and any* $\mathbf{v} \in \hat{R}_q^{\,d+1}$*, we have:*

$$\mathit{RERAND} + \Big(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\Big) \boxdot_r \mathbf{v} \approx \mathit{RERAND} + \Big(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\Big) \boxdot_r \mathbf{0}, \tag{2}$$

where $\mathbf{A}$ *is uniform over* $\hat{R}_q^{\,(d+1)\cdot\ell\times d}$.

We will now explain how randomized decomposition and a rerandomizer with the property of Lemma 2 are used to construct a circuit-private BlindRotation procedure. We will then prove Lemma 2 for each rerandomizer in Section 4.3.1 and 4.3.2, respectively.

We made two main modifications compared to the regular BlindRotation in TFHE gate bootstrapping: deterministic randomization is replaced by randomized decomposition and a rerandomizer is added at the last step of the BlindRotation.

Lemma 3. *For any* $\mathsf{testv} \in \hat{R}_q$*, the encryption scheme of Section 2.2 is circuit private for the class of functions* $\Big(\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\Big((\bar{\mathbf{a}}, \bar{b}), \cdot, \mathit{RERAND}\Big)\Big)_{(\bar{\mathbf{a}},\bar{b})}$*, if* RERAND *is a fresh rerandomizer which satisfies property (2).*

The proof of Lemma 3 is given in [BI22, Section 4.1]. We recall it for the sake of readability.

Proof. For $0 \le t < n$, ACC_t denotes the accumulator ACC after t iterations in Algorithm, and ACC^* the value of ACC after adding the rerandomizer RERAND, i.e., after the last iteration.

Algorithm 4 Private computation of a LWE encryption of $\mathsf{testv} \cdot X^{\bar{\varphi}}$ where $(\bar{\mathbf{a}}, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$, and $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} s_i \bar{a}_i$

Require: $(\bar{\mathbf{a}}, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$, a bootstrapping key $\mathrm{BK} = (\mathrm{BK}_i)_i$, where BK_i is a RGSW encryption of s_i for $i \in [1, n]$, a fresh rerandomizer RERAND.

Ensure: $\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\bar{\mathbf{a}}, \bar{b}), \mathrm{BK}, \mathrm{PK}\right)$: a RLWE encryption of $\mathsf{testv} \cdot X^{\bar{\varphi}}$ where $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} s_i \bar{a}_i$ and whose distribution is statistically close to a distribution independent from $(\bar{\mathbf{a}}, \bar{b})$, except for the message part.

1: $\mathsf{ACC} = (0, \ldots, 0, \mathsf{testv} \cdot X^{-\bar{b}}) \in \hat{R}_q^{\,d+1}$,
2: **for** $i = 1$ **to** $n - 1$
3: $\quad \boxed{\mathsf{ACC} += \mathrm{BK}_i \boxdot_r ((X^{\bar{a}_i} - 1)\mathsf{ACC})}$
4: $\boxed{\mathsf{ACC} += \mathrm{BK}_n \boxdot_r ((X^{\bar{a}_n} - 1)\mathsf{ACC}) + \mathrm{RERAND}}$
5: Return ACC

After the t-th iteration, for $0 \leq t \leq n$, we have:

$$\begin{aligned}
\mathsf{ACC}_t =& \mathrm{BK}_t \boxdot_r \left(\left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}\right) + \mathsf{ACC}_{t-1} \\
=& (\mathrm{BK}_t - s_t \mathbf{G} + s_t \mathbf{G}) \boxdot_r \left(\left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}\right) + \mathsf{ACC}_{t-1} \\
=& (\mathrm{BK}_t - s_t \mathbf{G}) \boxdot_r \left(\left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}\right) + \left(1 + \left(X^{\bar{a}_t} - 1\right) s_t\right) \mathsf{ACC}_{t-1}
\end{aligned}$$

Since $1 + (X^{\bar{a}_t} - 1)s_t = X^{\bar{a}_t \cdot s_t}$, we have:

$$\mathsf{ACC}_t = X^{\bar{a}_t s_t} \mathsf{ACC}_{t-1} + (\mathrm{BK}_t - s_t \mathbf{G}) \boxdot_r \left(\left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}\right) \tag{3}$$

We prove by induction that for t from n down to 0, the following holds:

$$\mathsf{ACC}^* \approx X^{\sum_{i=t+1}^{n} s_i \bar{a}_i} \mathsf{ACC}_t + \sum_{j=t+1}^{n} (\mathrm{BK}_j - s_j \mathbf{G}) \boxdot_r \mathbf{0} + \mathrm{RERAND} \tag{4}$$

For $t = n$, we have $\mathsf{ACC}^* = \mathsf{ACC}_n + \mathrm{RERAND}$. Now assume that (4) holds at step $1 \leq t \leq n$. Using (3), (4) can be rewritten:

$$\begin{aligned}
\mathsf{ACC}^* \approx & X^{\sum_{i=t+1}^{n-1} s_i \bar{a}_i} \left(X^{\bar{a}_t s_t} \mathsf{ACC}_{t-1} + (\mathrm{BK}_t - s_t \mathbf{G}) \boxdot_r \left(\left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}\right)\right) \\
& + \sum_{j=t+1}^{n-1} (\mathrm{BK}_j - s_j \mathbf{G}) \boxdot_r \mathbf{0} + \mathrm{RERAND}
\end{aligned}$$

For any $\mathbf{v} \in \hat{R}_q^{\,d+1}$, $X^i \Lambda^{\perp}(\mathbf{G}) = \Lambda^{\perp}(\mathbf{G})$, and $\left\|X^{-i} \cdot \mathbf{v}\right\|_2 = \|\mathbf{v}\|_2$, because they have the same coefficients modulo $X^{2N} - 1$ in absolute value. So for any parameter $r \in \mathbb{R}$, $X^i \mathbf{G}_r^{-1}(\mathbf{v}) = \mathbf{G}_r^{-1}(X^i \cdot \mathbf{v})$. Taking $\mathbf{v} = \left(X^{\bar{a}_t} - 1\right) \mathsf{ACC}_{t-1}$, this implies that

$$X^{\sum_{i=t+1}^{n-1} s_i \bar{a}_i} \left((\mathrm{BK}_t - s_t \mathbf{G}) \boxdot_r \mathbf{v}\right) = (\mathrm{BK}_t - s_t \mathbf{G}) \boxdot_r \left(X^{\sum_{i=t+1}^{n} s_i \bar{a}_i} \cdot \mathbf{v}\right)$$

Now, using Lemma 2, we obtain:

$$\mathsf{ACC}^* \approx X^{\sum_{i=t}^{n-1} s_i \bar{a}_i} \mathsf{ACC}_{t-1} + \sum_{j=t}^{n-1} (\mathrm{BK}_j - s_j \mathbf{G}) \boxdot_r \mathbf{0} + \mathrm{RERAND}$$

Finally, when $t = 0$, $X^{-\bar{b}} \cdot X^{\sum_{i=0}^{n-1} s_i \bar{a}_i} = X^{\bar{\varphi}}$ where $\bar{\varphi} := \sum_{i=t}^{n-1} s_i \bar{a}_i - \bar{b}$ and we have:

$$\mathsf{ACC}^* \approx \left(\mathbf{0}, \mathsf{testv} \cdot X^{\bar{\varphi}}\right) + \sum_{j=1}^{n-1} (\mathrm{BK}_j - s_j \mathbf{G}) \boxdot_r \mathbf{0} + \mathrm{RERAND},$$

which is independent of $\bar{a}$ and $\bar{b}$, except for the result $\mathsf{testv} \cdot X^{\bar{\varphi}}$.

The following simulator $\mathsf{Sim}_{cp}\left(\mathsf{testv} \cdot X^{\bar{\varphi}}, \mathrm{BK}, \mathrm{RERAND}\right)$ thus correctly simulates the output of $\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\bar{\mathbf{a}}, \bar{b}), \mathrm{BK}, \mathrm{RERAND}\right)$:

$$\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\mathbf{0}, 0), \mathrm{BK}, \mathrm{RERAND}\right) + \left(\mathbf{0}, \mathsf{testv} \cdot \left(X^{\bar{\varphi}} - 1\right)\right)$$

□

4.3.1 First approach

In this approach, we take $\mathrm{RERAND} := (\mathbf{0} \mid y)$, where each coefficient of $y \in \frac{1}{q}R$ is a Gaussian sample of variance $\vartheta_{\boxdot}$, and we also require that the decomposition parameter ℓ is large. In this section, we use the notation $(\mathbf{a} \mid b)$ to denote a sample $(\mathbf{a}, b)$, as we may handle the left part and the right part separately. We wanted to show that RERAND satisfies the property of Lemma 2. For any $\tilde{\mathbf{s}} \in \hat{R}_q^{\,d}$, any $\mathbf{e} \in \hat{R}_q^{\,(d+1)\cdot\ell}$, and any $\mathbf{v} \in \hat{R}_q^{\,d+1}$, we have:

$$(\mathbf{0}, y) + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{v} = \left(\mathbf{A} \boxdot_r \mathbf{v} \mid (\mathbf{A} \boxdot_r \mathbf{v})\tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y\right),$$

where $\mathbf{A}$ is uniform over $\hat{R}_q^{\,(d+1)\cdot\ell \times d}$.

In order to prove Lemma 2, we dealt with the left part and the right part separately: (a) we proved that the left part $\mathbf{A} \boxdot_r \mathbf{v}$ conditioned on the knowledge of $\mathbf{e} \boxdot_r \mathbf{v} + y$ is statistically close to a uniform sample $\mathbf{u} \in \hat{R}_q^{\,d}$; (b) $\mathbf{e} \boxdot_r \mathbf{v} + y$ is statistically close to a Gaussian sample, whose distribution is independent of $\mathbf{v}$.

In order to prove (b), we generalized the Gaussian lemmata over plain LWE from [BdPMW16] to rings. In order to prove (a), we need a regularity lemma over $\hat{R}_q$ for conditional distributions to show:

$$\left(\mathbf{A} \boxdot_r \mathbf{v} \mid \mathbf{e} \boxdot_r \mathbf{v} + y\right) \approx_s \left(\mathbf{u} \mid \mathbf{e} \boxdot_r \mathbf{v} + y\right) \tag{5}$$

However, $\mathbf{A} \boxdot_r \mathbf{v}$ is not uniform because if there is a line $\mathbf{a}$ of $\mathbf{A}$, whose coefficients are all non invertible, then $\mathbf{a} \boxdot_r \mathbf{v}$ cannot be invertible. In that case, $(\mathbf{a}, \mathbf{a} \boxdot_r \mathbf{v})$ is not uniform, since a pair $(\mathbf{a}, \mathbf{a} \boxdot_r \mathbf{v})$ with all coefficients of $\mathbf{a}$ non invertible and $\mathbf{a} \boxdot_r \mathbf{v}$ invertible couldn't appear. In order to have $\mathbf{A} \boxdot_r \mathbf{v}$ close to uniform in that case, we made this probability that this event occurs negligibly small, by taking the decomposition parameter ℓ sufficiently large.

We showed that this approach, which can be viewed as a direct adaptation of [BdPMW16] over rings works but does not lead to an efficient FHE circuit-private algorithm for standard TFHE parameters. Now, using (a) and (b), we obtained:

$$\left(\mathbf{A} \boxdot_r \mathbf{v} \mid (\mathbf{A} \boxdot_r \mathbf{v})\tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y\right) \approx_s (\mathbf{u} \mid \mathbf{u} \cdot \tilde{\mathbf{s}} + e'),$$

where $\mathbf{u}$ is uniform over $\hat{R}_q^{1\times d}$ and $e' \leftarrow \mathcal{D}_{\frac{1}{q}R,\Gamma}$ for some Γ independent of $\mathbf{v}$. We can conclude by invoking (2) in the two directions.

4.3.2 Second approach

In [BI22], we introduced an alternative approach that makes use of a public key encryption scheme $(\mathsf{PkGen}, \mathsf{PkEnc})$ with tailored properties on its distribution. In particular, for any secret key $\tilde{\mathbf{s}}$, a public key encryption of zero $\mathsf{PkEnc}(\text{PK})$ has the following property:

$$(\text{PK}, \mathsf{PkEnc}(\text{PK})) \approx (\text{PK}, (\mathbf{u}, \mathbf{u} \cdot \tilde{\mathbf{s}} + e_u)), \tag{6}$$

where $\mathbf{u}$ is uniform and e_u has a variance bounded by the norms of a honestly generated public key $\text{PK} \leftarrow \mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$ and the noise of $\mathsf{PkEnc}(\text{PK})$.

In [BI22, Section 3.4], we showed that the following construction, which is a slight variant of the public key encryption scheme proposed by Linder and Peikert [LP11], satisfies the property (6). For our purposes, it suffices to take $d' = 1$ but we proved it for an arbitrary d'.

- $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$: on input the security parameter λ, and a secret key $\tilde{\mathbf{s}} \in R^d$,

  $$\text{PK} = (\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}) \in \hat{R}_q^{d'\times d+1},$$

  where $\mathbf{A}$ is uniform over $\hat{R}_q^{d'\times d}$, and each coefficient of $\mathbf{e} \in \hat{R}_q^{d'}$ has coefficients sampled from $\chi_{\vartheta_\mathbf{e}}$, for a noise variance $\vartheta_\mathbf{e}$.

- $\mathsf{PkEnc}(\text{PK})$: on input public key PK, returns

  $$\mathbf{r} \cdot \text{PK} + (\mathbf{e}' \mid e'') \in \hat{R}_q^{1\times d+1},$$

  with $\mathbf{r} \leftarrow \mathcal{D}_{R^{1\times d'}, q\sqrt{2\pi\vartheta_\mathbf{r}}}$, $\mathbf{e}' \leftarrow \mathcal{D}_{\frac{1}{q}R^{1\times d}, \sqrt{2\pi\vartheta_{\mathbf{e}'}}}$ and $e'' \leftarrow \mathcal{D}_{\frac{1}{q}R, \sqrt{2\pi\vartheta_{e''}}}$,

where $\mathcal{D}_{C,\sigma}$ denotes a spherical Gaussian distribution over C of parameter $\sigma I_{\dim}$, where $\dim = Nd'$ (for $\mathbf{r}$) or $\dim = Nd$ (for $\mathbf{e}'$).

The challenge of proving that this public key construction satisfies property (6) was dealing with the fact that the encryption secret key is known, which introduces dependencies between the random variables. We therefore showed that the joint distributions of these dependent variables are close to the target distribution. We proved this result under the decisional Ring LWE assumption with Gaussian secret key and Gaussian noise, by adapting the Gaussian regularity lemma from [BdPMW16, KLSS23].

Now, taking $\text{RERAND} := (\mathbf{0}, y) + \mathsf{PkEnc}(\text{PK})$, with y sampled as before and PK is honestly generated from $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, Lemma 2 can be proven as before, but now $\approx$ holds in a computational sense in the FHE circuit privacy condition. We refer to [BI22, Section 4.1] for the

detailed proof.

From circuit privacy to sanitization. In the sanitization algorithm we proposed, the BlindRotation is replaced by the circuit private BlindRotation. The simulator for sanitization works exactly as the sanitization except that it computes the output of $\mathbf{c}' = \mathsf{Sim}_{cp}\left(\mathsf{testv} \cdot X^{\bar{\varphi}}, \mathrm{BK}, \mathrm{RERAND}\right)$ instead of calling the circuit private BlindRotation. The result follows from the the fact that the output of $\mathsf{Extract}$ with $\mathbf{c}'$ input on one hand and $\mathsf{Sim}_{cp}\left(\mathsf{testv} \cdot X^{\lfloor 2N\mu \rceil}, \mathrm{BK}, \mathrm{RERAND}\right)$ on the other hand are exactly the same. We refer to [BI22, Section 4.2] for the detailed proof.

Overall result. Our contribution in this work can thus be summarized as follows:

- generalizing the result on convolution lemmas Gaussian over rings and on arbitrary lattices;
- introducing a more compact rerandomizer, which can also be used in other sanitization approaches and showing that it satisfies the requirement 6 under the decisional RLWE assumption. This approach has recently been used by [HMS25] where discrete Gaussians are replaced by rounded Gaussians;
- specializing the regularity lemma over rings for power-of-two moduli. We do not develop this result in this manuscript and refer to [BI22, Appendix C] for the proof.

4.4 Comparisons with other sanitization algorithms

Comparison with the methodology used in [BdPMW16]: to deal with dependencies between polynomial coefficients and random variables, we proceed by induction over the $\mathsf{BlindRotate}$ circuit size as in [BdPMW16]. However, instead of using a forward induction, we use a backward induction that shows that the rerandomizer only needs to be added in the final step of $\mathsf{BlindRotate}$, rather than adding it at each step. As a side effect, the computation time and output noise are reduced assuming the same decryption failure probability compared to the output noise that would be obtained by applying directly [BdPMW16] to rings.

In the following, we compare our sanitization algorithm with the existing ones, which follow the same strategy as ours i.e., which modify the bootstrapping internally.

- [Klu24] introduced a statistical LWE sanitization procedure by modifying the FHEW-like bootstrapping restricted to binary secret keys. Although statistical sanitization is stronger, our computational variant considerably reduces the size of the public key. For comparison purposes, in [Klu24], the public sanitization key is composed of $2^{11.5}$ to $2^{12.3}$ fresh LWE samples of zero in dimension $N = 2^{11}$, while in our case it is made of one RLWE encryption of zero in dimension $N = 2^{11}$. Our strategy can also be adapted to the FHEW setting. However, FHEW bootstrapping uses only certain components of the bootstrapping key, with the specific component selected depending on the input ciphertext. Hiding which components were actually used would require adding dummy computations, leading to a significant drop in efficiency. By contrast, TFHE bootstrapping used all components of the bootstrapping key, removing the need to hide key-dependent accesses.

- Recently, the authors of [HMS25] significantly improved our sanitization method, they proposed an efficient sanitization procedure for TFHE. Their solution builds on our idea of using a public rerandomizer and also relies on the decisional RLWE assumption. Their proof strategy for the rerandomizer follows the same blueprint as ours, except that they replace discrete Gaussians with rounded Gaussian samples. In addition, their sanitization algorithm avoids randomized decomposition and instead appies randomization before and after the bootstrapping. In contrast, our method introduces a slight internal modification to the bootstrapping procedure, and the proof relies on a generalization of the noise analysis from [BdPMW16] to the ring setting. This allows us to achieve multi-hop circuit privacy for branching programs over rings without relying on bootstrapping as opposed to [Klu24] and, consequently, without requiring the circular security assumption. [HMS25] may provide multi-hop circuit privacy for branching programs with slight modification but the authors did not mention whether their construction achieves it directly without requiring the circular security assumption.

- The implementation of [Klu24, HMS25] shows better timings than those obtained with our sanitization approach. The first reason is that, like the method from [Klu24], our method is highly based on Gaussian sampling, which is challenging to implement efficiently. The second reason is that combining conditions to reach property (2) with the requirement on the parameter of $\mathbf{G}_r^{-1}$ scheme imposes restrictive constraints on the parameters. As a result, the optimal parameters for the regular bootstrapping do not apply in our case. We estimate the efficiency of our method by comparing the relative cost between the non-sanitized TFHE bootstrapping and the sanitized TFHE bootstrapping. Our experiments show that our sanitization approach induces an overhead of around 25% compared to the cost of the non-sanitizing TFHE bootstrapping for the same set of parameters, assuming Gaussian samples are precomputed.

4.5 Applications of circuit privacy and sanitization

In this section, we review contexts in which circuit privacy and sanitization could be useful. A natural application occurs when the function to be evaluated needs to be protected. However, the techniques selected to design a circuit-private solution may vary depending on the circuit to be evaluated and the noise-to-modulus ratio of the FHE scheme. In [INdPP22], we proposed a verifiable oblivious polynomial evaluation protocol implemented using the BFV scheme, where we relied on noise flooding techniques and the hiding property of the underlying commitment scheme to prove the privacy of the sender. In [BI25a], we proposed a method that allows a receiver to privately query the evaluation of points. In this two-party evaluation protocol, the circuit consists of a sum of polynomial multiplications. And circuit privacy can be achieved by performing an appropriate modulus switch on the server side after the repacking before sending back the result to the client. In the following, we mention possible applications of our sanitization approach, including those related to its building blocks.

- *Simulatablity of FHE ciphertext and threshold FHE.* In threshold FHE [AJL+12, BGG+18, BS23, PS24], a secret key is shared among n parties so that they can collaboratively and correctly decryt an FHE ciphertext, even if up to t of the n parties are corrupted. Such requirements may arise in applications such as evoting, where a set of the decrypted

servers may be unavailable or compromised. In that case, the tally can still be correctly computed. In [CGGI13a], we described a fully distributed threshold cryptosystem suitable for partial decryption of ElGamal ciphertexts, where each tally partially decrypts and the decryption shares can then be publicly aggregated to retrieve the election result. One of the challenging problems in designing secure threshold FHE is that, after partially decrypting the FHE ciphertext, the noise may carry information on the computation and possibly on the secret key. The decrypting parties could add large noise to mask the "b" part of the ciphertext, which implies that the modulus has to be large. This could be a problem if the starting modulus is much smaller than the one required to apply the noise flooding techniques.

Last year, [PS24] proposed a new threshold FHE schemes, where the server provides some help to securely decrypt, i.e. where the server randomizes the ciphertext before each user partially decrypts it. This assumes that the server does not collude with another party and acts as an honest party. In their solution, the server applies a double flood and a randomized rounding operation from a large to a small modulus so that the user applies partial decryption at a small modulus. Recently, Smart and Walter [SW25] extended this method to TFHE, and more generally to FHE schemes with small modulus. In order to prove their result, they showed that our construction provides error simulatability, which means that there exists a simulation algorithm such that for all ciphertexts $c \in \mathcal{C}_\mu$, there exists a simulator Sim such that $\mathsf{Sim}(\mu, \mathrm{PK}, \mathrm{SK}) \approx (\mathsf{Sanitize}(c), \mathrm{PK}, \mathrm{SK})$ with overwhelming probability over the choice of $(\mathrm{PK}, \mathrm{SK})$ produced by $\mathsf{KeyGen}(1^\lambda)$.

- *FuncCPA security for client-aided outsourcing protocols.* This FuncCPA security notion was introduced in [AGHV22] to address security in client-aided outsourcing protocols. In such a protocol, part of the server's computation can be outsourced to a client, who decrypts, evaluates a function, and then re-encrypts the result before sending it back to the server. Hence, to model privacy with respect to the client in this scenario, the server is provided access to a re-encryption oracle that inputs a function. The adversary provides a function and an arbitrary ciphertext and receives the corresponding re-encryption.

  The authors of [AGHV22] showed that IND-CPA security is not sufficient to reach funcCPA security for client-aided outsourcing protocols and introduced a generic way to design a funcCPA secure FHE scheme from an IND-CPA secure FHE scheme and a sanitization algorithm. But, as noticed by the authors of [SW25], the client has to run the sanitization algorithm for each re-encryption request, which could make the client's work as inefficient as if the server does it by itself. The authors of [SW25] proposed a more efficient solution, which takes advantage of the fact that the client knows the message. The client runs the simulation algorithm instead of the sanitization algorithm and the sanitization property guaranties that the output of Sim carries no information about the input ciphertext.

- *Oblivious transfer protocol with a proxy.* In an earlier version of [BI22], we showed that a public key encryption scheme $(\mathsf{PkGen}, \mathsf{PkEnc})$ defined over rings, satisfying condition (6) and this additional property:

$$(\mathrm{PK}, X^j \mathsf{PkEnc}(\mathrm{PK})) \approx (\mathrm{PK}, \mathsf{PkEnc}(\mathrm{PK})) \text{ for any } \mathrm{PK} \text{ and } j \in \mathbb{Z}_{2N}$$

can be used as a building block for constructing a protocol in which a (honest-but-curious) proxy randomizes ciphertexts. The public key encryption scheme from Section 4.3.2 satisfies property (4.5) since all the errors are spherical Gaussians.

The idea of having a proxy that filters data and rerandomizes ciphertexts can be found in Access Control Encryption [DHO16]. By adding a message to the $(d+1)$-th coordinate of $\mathsf{PkEnc}(\text{PK})$, we obtain a public key encryption scheme for polynomials with the useful following property: it coefficient-wise packs N messages into a single polynomial ciphertext and rotates the messages across the different slots. An additional property is that multiplying by X does not change the distribution of the ciphertext, even for an adversary who knows the secret key. We illustrate the idea through a protocol between $\mathcal{A}$ (the sender) , $\mathcal{B}$ (the proxy), and $\mathcal{C}$ (the receiver), which works as follows:

1. $\mathcal{A}$ honestly generates the secret key $\tilde{\mathbf{s}}$ and the public key $\text{PK} = \mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$. It keeps $\tilde{\mathbf{s}}$ secret and broadcasts PK.
2. $\mathcal{B}$ encrypts N messages $(x_1, \ldots, x_N)$ as $\text{CT} = \mathsf{PkEnc}(\text{PK}) + \left(\mathbf{0}, \sum_{i=1}^{N} x_i X^i\right)$ and sends the resulting ciphertext to $\mathcal{C}$;
3. $\mathcal{C}$ blindly selects the j-th message and sends $\mathsf{Extract}(X^{-j}\text{CT})$ to $\mathcal{A}$;
4. $\mathcal{A}$ retrieves x_j, while no information about j or about any x_i with $i \neq j$ is revealed.

5 Perspectives

Fully Homomorphic Encryption has emerged as a powerful cryptographic technique with applications in an increasingly large number of domains. Interestingly, FHE is being now applies in contexts that were not originally considered as part of its primary use case. Recently, FHE has been proposed to design confidential smart contracts, enabling to protect sensitive information on-chain data [SWA23, ABC+25]. In addition, the large scale deployments of Artificial Intelligence algorithms introduces both efficiency constraints and privacy challenges, for which FHE offers a promising privacy-preserving solution. Recently, there has been growing interest within the community regarding the definitional aspects and security ntions of FHE, in order to gain a deeper understanding of the security guarantees that FHE schemes can provide and the levels of security tailored to sepcific applications. This section summarizes some of the research challenges in FHE I would like to inestigate.

Efficiency and Practical Performance. Recently, a growing number of works have focused on accelerating homomorphic computations by exploiting hardware mechanisms ([GVP+23, vBDTV23, PBT+24, BPV25] for example), such as SIMD instructions and multicore processors. These works highlight the fact that achieving practical performance also relies on algorithmic improvements and careful integration on suitable architectures. Since algorithmic improvements have the potential to be translated to even more substantial performance gains when using specialized hardware, designing efficient homomorphic algorithms remains a highly relevant question. An additional motivation stems from the fact that real-world applications require optimizations tailored to particular settings, since data representation and corresponding operations differ among contexts. Futhermore, some applications require processing large volumes of encrypted data, which may adversely affect performance, especially when the employed FHE scheme offers limited throughput.

Composability of the Techniques. A large number of homomorphic optimizations have been proposed for FHEW-like bootstrappings since the first practical FHEW-like schemes have been proposed. However, many of these techniques are not directly composable. For example, variants of the bootstrapping proposed in [LMSS23, BCL+24] achieve computational and memory speedups for the TFHE gate bootstrapping by introducing new secret key distributions. However, it appears not direct to compose these techniques with the extended bootstrapping [LY23] as the later is tied to the use of a ring morphism defining the domain size of the function to be evaluated through the functional bootstrapping. A natural approach to enable optimization composability will be to explore how efficient "bridges" between different FHE schemes can be adapted to these specific settings. It would be interesting to explore whether tailored modifications could enable the integration of other optimizations.

Scheme Switching. As discussed in the introduction of the manuscript, there are different families of FHE schemes, each being better suited to a specific message representations and computational requirements. Consequently, no single scheme is universally optimal for all use cases. Scheme switching techniques aim to bridge this gap by enabling ciphertexts encrypted under one scheme to be transformed into ciphertexts compatible with another scheme.

Existing works have shown that scheme switching is a promising approach both for optimizing performance and for enabling new functionalities. Switching techniques can also provide a unified perspective on the different schemes, which contributes to provide a comprehensive view of different optimization strategies. In this context, a systematic framework could offer valuable insights into both practical implementations and design choices.

Memory Usage. In addition to time complexity, memory usage has become increasingly relevant to optimize as FHE is being considered for applications involving large datasets. Several approaches can be explored to reduce memory usage. A first approach consists in decreasing the size of evaluation keys, such as bootstrapping and keyswitching keys, which often dominate the memory cost. Depending on the use case, this can be addressed by updating the evaluation keys during the execution of the protocol rather than storing a large set of precomputed keys. This approach has been proposed by [LLKN23] for the design of a hierarchical key management system, where the computation server owns a (small) set of master evaluation keys and generates new keys when required. This method reduces memory by shifting "a memory part" to the online computation phase. This approach was applied in the context of homomorphic rotation operations, but might be useful in other multi-user settings where a master key is generated and user-specific key material can be publicly updated, possibly with auxiliary information provided by the client.

Another approach to reduce the size of the keys is to integrate amortization techniques. For example, part of the computation—such as decomposition of ciphertexts—can be performed once, and subsequently reused to perform multiple homomorphic operations, such as several homomorphic rotations for instance. It would be interesting to investigate whether this method could be applied to further optimize memory usage in recent variants of the TFHE gate bootstrapping.

Preprocessing. The computational models commonly considered in FHE, including those on which my prior work has focused, assume that FHE computations are compiled into a circuit. While this model is well-suited for many applications, it becomes less efficient in some scenarios. For example, in FHE-based *Private Information Retrieval* (PIR) protocols, a client asks an encrypted index to a server holding a database, the server answers the query so that the client retrieves the item at the queried index in the database. The communication should be sublinear in the database size; otherwise, the server could send all the database contents in clear. However, when the database is very large, the server processes a large volume of data. In particular, answering a query requires accessing all entries in the database to ensure that no information on the index is leaked to the server. To address this limitation, one approach consists in introducing a client-side pre-processing phase, during which the client computes auxiliary information, referred to as hints, that can later be used by the server to reduce the amount of data processed. Schemes following this paradigm, with or without a client-side preprocessing phase, are called doubly efficient PIR (DPIR).

The authors of [LMW23] proposed the first client-independent preprocessing based on standard assumption (as opposed to non-standard assumptions as ideal obfuscation), i.e., on the RLWE assumption relying on Algebraic Somewhat Homomorphic Encryption (ASHE), which provides a ring structure on ciphertexts. A year later, the authors of [OPPW25] improved the DPIR constructions from [LMW23, OPPW24] relying on (among other technical

contributions) a relaxed notion of ASHE.

A related primitive is Oblivious RAM (*random access machine*) (ORAM) protocol . In [LMW23], the authors showed how to construct an FHE-based RAM from their DPIR scheme. In FHE oblivious RAM, an arbitrary program P is homomorphically evaluated on a client's encrypted input and a server's preprocessed plaintext input, with an evaluation time proportional to the execution time of the RAM program.

Such preprocessing-based techniques are particularly appealing in the context of FHE, as they enable sublinear time complexity, which is otherwise unattainable in the circuit model. However, there is still a gap between concrete implementations and sublinear generic constructions, as the latter do not necessarily yield practically efficient schemes. These observations motivate an investigation into whether the computational cost of oblivious RAM constructions—still significant for large databases [CCR19]—can be reduced. A possible direction is to explore improvements to existing protocols, as well as the use of amortization techniques to support handle multiple queries simultaneously.

Threshold FHE. Current FHE schemes can be extended as multi-party FHE in different ways. For example, FHE schemes can be extended as a multi-key FHE ([LTV12, CM15, MW16, CCS19, KKL$^+$23, KMS24] for example), where each party holds its own secret key and evaluates a circuit collaboratively, or as a threshold FHE ([AJL$^+$12, BD10, BGG$^+$18, LMK$^+$23] for example), where a subset of parties (whose number is less than a threshold) must be honest to decrypt correctly. In this scenario, splitting the decryption task among several parties mitigates dishonest behaviors and ensures correct decryption even if the secret key is lost by one (resp. a small set) of the parties.

Several methods have already been proposed to achieve multi-party FHE schemes. Some of these protocols could achieve active security, which means that one or more parties can be corrupted, or strong security requirements such as dishonest majority which tolerate a large number of corruptions. In multi-party FHE schemes, the setup can be single-hop, where the list of parties has to be known before the computation starts, or multi-hop, where additional parties can join the computation dynamically. However, achieving all these properties and features simultaneously remains hard. For example, Boudgoust and Scholl [BS23] proposed a threshold FHE construction that applies to schemes with polynomially small modulus, but the security of their scheme is proved in the game-based selective model, i.e. the adversary is allowed to ask queries before receiving the challenge ciphertext. Last year, Passeguèle and Stehlé [PS24] proposed a simulation-secure threshold FHE scheme with low communication complexities, applicable to schemes with super-polynomially large modulus for the input ciphertext. This raises the interesting question of how the security of threshold FHE schemes with polynomially small moduli could be strengthened.

Publication list

2026 Plug-and-play sanitization for TFHE, with Florian Bourse. preprint, https://hal.science/hal-05507283v1/document.

2026 Large Domain Homomorphic Evaluation in Leveled Mode, with Jean-Philippe Bossuat. Designs, Codes and Cryptography, Published Feb. 2026, DOI:10.1007/s10623-025-01782-x, presented at FHE.org 2024.

2025 TETRIS: Composing FHE Techniques for Private Functional Exploration Over Large Datasets. with Jean-Philippe Bossuat, Proceedings on Privacy Enhancing Technologies, PoPETs volume 2025.

2024 Efficient Post-Quantum Pattern Matching on Encrypted Data, with Anis Bkakria, CIC, IACR Communications in Cryptology, Volume 1, Issue 2 2024.

2023 Identity-Based Encryption from Lattices Using Approximate Trapdoors, with Lucas Prabel and Adeline Roux-Langlois. ACISP 2023.

2022 Homomorphically counting elements with the same property, with Ilia Iliashenko, Axel Mertens and Hilder V. L. Pereira. PoPETS 2022, Volume 4.

2022 MyOPE: Malicious securitY for Oblivious Polynomial Evaluation, with Paola de Perthuis, Anca Nitulescu and David Pointcheval. SCN 2022.

2021 Secure Hybrid Encryption in the Standard Model from Hard Learning Problems, with Xavier Boyen and Qinyi Li. PQCrypto 2021.

2020 TFHE Fast Fully Homomorphic Encryption over the Torus, with Ilaria Chillotti, Nicolas Gama and Mariya Georgieva. Journal of Cryptology 2020.

2020 A Simple and Efficient CCA-Secure Lattice KEM in the Standard Model, with Xavier Boyen and Qinyi Li. SCN 2020.

2019 New Techniques for Multi-value Input Homomorphic Evaluation, with Sergiu Carpov and Victor Mollimard. CT-RSA 2019.

2019 Privacy-preserving k-means clustering: an application to driving style recognition, with Othmane El Omri, Aymen Boudguiga and Witold Klaudel. NSS 2019.

2019 Practical fully homomorphic encryption for fully masked neural networks, with Renaud Sirdey and Martin Zuber. CANS 2019.

2017 Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE, with Ilaria Chillotti, Nicolas Gama and Mariya Georgieva. Asiacrypt 2017.

2016 Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 seconds, with Ilaria Chillotti, Nicolas Gama and Mariya Georgieva. Asiacrypt 2016.

2016 Structural Lattice Reduction: Worst-Case to Average-Case Reduction. for (Almost) All Lattices, with Nicolas Gama, Phong Nguyen and Xiang Xie. Eurocrypt 2016.

2016 An LWE-based Homomorphic e-voting Scheme, with Ilaria Chillotti, Nicolas Gama and Mariya Georgieva. PQCrypto 2016.

2014 Election Verifiability for Helios under Weaker Trust Assumptions, with Véronique Cortier, David Galindo and Stéphane Glondu. ESORICS 2014.

2013 Distributed ElGamal a la Pedersen - Application to Helios, with Véronique Cortier, David Galindo and Stéphane Glondu. WPES 2013.

2012 Divisible e-cash in the standard model, with Benoît Libert. Pairing 2012.

2011 Block-wise P-Signatures and Non-Interactive Anonymous Credentials with Efficient Attributes, with Benoît Libert and Damien Vergnaud. IMA International Conference 2011.

2010 Mediated Traceable Anonymous Encryption, with David Pointcheval et Damien Vergnaud. Latincrypt 2010.

2010 Batch Groth Sahai, with Olivier Blazy, Amandine Jambert, Georg Fuchsbauer, Hervé Sibert and Damien Vergnaud. ACNS 2010.

2009 New Definitions for Identity-Based Encryption and Application, with David Pointcheval. Formal to Practical Security 2009.

2008 Anonymous and Transparent Gateway-based Password-Authenticated Key Exchange, with Michel Abdalla and David Pointcheval. CANS 2008.

2008 New Definitions for Identity-Based Encryption and Application, with David Pointcheval. SCN 2008.

2007 An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentification, with Julien Bringer, Hervé Chabanne, David Pointcheval, Qiang Tang and Sébastien Zimmer. ACISP 2007.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010.

[ABC+25] Daniel Aronoff, Adithya Bhat, Panagiotis Chatzigiannis, Mohsen Minaei, Srinivasan Raghuraman, Robert M. Townsend, and Nicolas Xuan-Yi Zhang. SoK: Fully-homomorphic encryption in smart contracts. Cryptology ePrint Archive, Report 2025/527, 2025.

[AdMP] Ben Adida, Olivier de Marneffe, and Olivier Pereira. Helios voting system. http://heliosvoting.org.

[AGHV22] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable CCA2 relaxation for homomorphic encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 70–99. Springer, Cham, November 2022.

[AJL+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 David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Berlin, Heidelberg, April 2012.

[AKP25] Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using CKKS. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 304–337. Springer, Cham, August 2025.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Berlin, Heidelberg, August 2014.

[BBB+25] Sonia Belaïd, Nicolas Bon, Aymen Boudguiga, Renaud Sirdey, Daphné Trama, and Nicolas Ye. Further improvements in AES execution over TFHE. *CiC*, 2(1):39, 2025.

[BCK+23] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In Helena Handschuh and Anna Lysyanskaya, editors,

CRYPTO 2023, Part IV, volume 14084 of *LNCS*, pages 37–69. Springer, Cham, August 2023.

[BCKS24] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, and Damien Stehlé. Bootstrapping bits with CKKS. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 94–123. Springer, Cham, May 2024.

[BCL+24] Loris Bergerat, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, Adeline Roux-Langlois, and Samuel Tap. New secret keys for enhanced performance in (T)FHE. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2547–2561. ACM Press, October 2024.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 201–218. Springer, Berlin, Heidelberg, February 2010.

[BDF18] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18*, volume 10831 of *LNCS*, pages 217–251. Springer, Cham, May 2018.

[BdPMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89. Springer, Berlin, Heidelberg, August 2016.

[Bel] Verifiable online voting system. `http://www.belenios.org`.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.

[BGGJ20] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. CHIMERA: combining ring-lwe-based fully homomorphic encryption schemes. *J. Math. Cryptol.*, 14(1):316–338, 2020.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BGW00] Allan L. Buchsbaum, Michael R. Giancarlo, and Jonathan R. Westbrook. On the determinization of weighted finite automata. *SIAM Journal on Computing*, 30(6):1839–1854, 2000.

[BI22] Florian Bourse and Malika Izabachène. Plug-and-play sanitization for TFHE. Cryptology ePrint Archive, Report 2022/1438, 2022.

[BI25a] Jean-Philippe Bossuat and Malika Izabachene. Large domain homomorphic evaluation for bfv-like schemes via ring repacking. *Designs, Codes and Cryptography, to appear*, 2025.

[BI25b] Jean-Philippe Bossuat and Malika Izabachène. TETRIS: Composing FHE techniques for private functional exploration over large datasets. *PoPETs*, 2025(2):23–38, April 2025.

[BIL20] Xavier Boyen, Malika Izabachène, and Qinyi Li. A simple and efficient CCA-secure lattice KEM in the standard model. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 321–337. Springer, Cham, September 2020.

[BIL21] Xavier Boyen, Malika Izabachène, and Qinyi Li. Secure hybrid encryption in the standard model from hard learning problems. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 399–418. Springer, Cham, 2021.

[BIP+22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 188–215. Springer, Cham, December 2022.

[BJ25] Olivier Bernard and Marc Joye. Bootstrapping (T)FHE ciphertexts via automorphisms: Closing the gap between binary and gaussian keys. *IACR Cryptol. ePrint Arch.*, page 163, 2025.

[BKSS24] Youngjin Bae, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. Bootstrapping small integers with CKKS. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 330–360. Springer, Singapore, December 2024.

[BMMP18] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 483–512. Springer, Cham, August 2018.

[BORLT25] Loris Bergerat, Jean-Baptiste Orfila, Adeline Roux-Langlois, and Samuel Tap. Accelerating TFHE with sorted bootstrapping techniques. Cryptology ePrint Archive, Paper 2025/2214, 2025.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 190–213. Springer, Berlin, Heidelberg, August 2016.

[BPV25] Jonas Bertels, Hilder V. L. Pereira, and Ingrid Verbauwhede. FINAL bootstrap acceleration on FPGA using DSP-free constant-multiplier NTTs. *IACR TCHES*, 2025(3):293–316, 2025.

[BR15] Jean-François Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 119–135. Springer, Cham, August 2015.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Berlin, Heidelberg, August 2012.

[BS23] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Singapore, December 2023.

[BST20] Florian Bourse, Olivier Sanders, and Jacques Traoré. Improved secure integer comparison via homomorphic encryption. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 391–416. Springer, Cham, February 2020.

[CCR19] Hao Chen, Ilaria Chillotti, and Ling Ren. Onion ring ORAM: Efficient constant bandwidth oblivious RAM from (leveled) TFHE. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 345–360. ACM Press, November 2019.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 446–472. Springer, Cham, December 2019.

[CDKS21] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 2021, Part I*, volume 12726 of *LNCS*, pages 460–479. Springer, Cham, June 2021.

[CGGI13a] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed elgamal à la pedersen: Application to helios. In Ahmad-Reza Sadeghi and Sara Foresti, editors, *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 131–142. ACM, 2013.

[CGGI13b] Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachene. A generic construction for voting correctness at minimum cost - application to helios. Cryptology ePrint Archive, Report 2013/177, 2013.

[CGGI14] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 327–344. Springer, Cham, September 2014.

[CGGI16a] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Berlin, Heidelberg, December 2016.

[CGGI16b] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based E-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 245–265. Springer, Cham, 2016.

[CGGI16c] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Fast fully homomorphic encryption library over the torus, 2016. `https://github.com/tfhe/tfhe`.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 377–408. Springer, Cham, December 2017.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995.

[CIM19] Sergiu Carpov, Malika Izabachène, and Victor Mollimard. New techniques for multi-value input homomorphic evaluation and applications. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 106–126. Springer, Cham, March 2019.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017.

[CLOT21] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 670–699. Springer, Cham, December 2021.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Berlin, Heidelberg, August 2015.

[dCJV21] Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. Fast vector oblivious linear evaluation from ring learning with errors. In *WAHC '21: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Virtual Event, Korea, 15 November 2021*, pages 29–41. WAHC@ACM, 2021.

[dCKK+24] Leo de Castro, Duhyeong Kim, Miran Kim, Keewoo Lee, Seonhong Min, and Yongsoo Song. More efficient lattice-based OLE from circuit-private linear HE with polynomial overhead. Cryptology ePrint Archive, Report 2024/1534, 2024.

[DG09] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. In Manfred Droste, Werner Kuich, and Hans Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs in Theoretical Computer Science, pages 149–214. Springer, Berlin, Heidelberg, 2009. Formal definition and theory of weighted finite automata.

[DHO16] Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 547–576. Springer, Berlin, Heidelberg, October / November 2016.

[DKMS24] Gabrielle De Micheli, Duhyeong Kim, Daniele Micciancio, and Adam Suhl. Faster amortized FHEW bootstrapping using ring automorphisms. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14604 of *LNCS*, pages 322–353. Springer, Cham, April 2024.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Berlin, Heidelberg, April 2015.

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 294–310. Springer, Berlin, Heidelberg, May 2016.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Berlin, Heidelberg, May 2004.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[GBA21] Antonio Guimarães, Edson Borin, and Diego F. Aranha. Revisiting the functional bootstrap in TFHE. *IACR TCHES*, 2021(2):229–253, 2021.

[Gen09a] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GINX16] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 528–558. Springer, Berlin, Heidelberg, May 2016.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013.

[GVP+23] Robin Geelen, Michiel Van Beirendonck, Hilder V. L. Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios D. Dimou, Ingrid Verbauwhede, Frederik Vercauteren, and David W. Archer. BASALISC: Programmable hardware accelerator for BGV fully homomorphic encryption. *IACR TCHES*, 2023(4):32–57, 2023.

[HMS25] Intak Hwang, Seonhong Min, and Yongsoo Song. Practical circuit privacy/sanitization for TFHE. *IACR Cryptol. ePrint Arch.*, page 216, 2025.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2014.

[Hwa23] Intak Hwang. TFHE-go. Online: `https://github.com/sp301415/tfhe-go`, 2023.

[IIMP22] Ilia Iliashenko, Malika Izabachène, Axel Mertens, and Hilder V. L. Pereira. Homomorphically counting elements with the same property. *PoPETs*, 2022(4):670–683, October 2022.

[INdPP22] Malika Izabachène, Anca Nitulescu, Paola de Perthuis, and David Pointcheval. MyOPE: Malicious SecuritY for oblivious polynomial evaluation. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 663–686. Springer, Cham, September 2022.

[ISZ19] Malika Izabachène, Renaud Sirdey, and Martin Zuber. Practical fully homomorphic encryption for fully masked neural networks. In Yi Mu, Robert H. Deng, and Xinyi Huang, editors, *CANS 19*, volume 11829 of *LNCS*, pages 24–36. Springer, Cham, October 2019.

[jLHH+21] Wen jie Lu, Zhicong Huang, Cheng Hong, Yiping Ma, and Hunter Qu. PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In *2021 IEEE Symposium on Security and Privacy*, pages 1057–1073. IEEE Computer Society Press, May 2021.

[Joy24] Marc Joye. TFHE public-key encryption revisited. In Elisabeth Oswald, editor, *CT-RSA 2024*, volume 14643 of *LNCS*, pages 277–291. Springer, Cham, May 2024.

[JP22] Marc Joye and Pascal Paillier. Blind rotation in fully homomorphic encryption with extended keys. In *Proceedings of the International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 1–18. Springer, 2022.

[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, Report 2021/691, 2021.

[KKL+23] Taechan Kim, Hyesun Kwak, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 726–740. ACM Press, November 2023.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *11th SODA*, pages 937–941. ACM-SIAM, January 2000.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.

[Klu22] Kamil Kluczniak. NTRU-v-um: Secure fully homomorphic encryption from NTRU with small modulus. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1783–1797. ACM Press, November 2022.

[Klu24] Kamil Kluczniak. Circuit privacy for FHEW/TFHE-style fully homomorphic encryption in practice. *CiC*, 1(4):33, 2024.

[KMS24] Hyesun Kwak, Seonhong Min, and Yongsoo Song. Towards practical multi-key TFHE: Parallelizable, key-compatible, quasi-linear complexity. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14604 of *LNCS*, pages 354–385. Springer, Cham, April 2024.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997.

[KS23] Kamil Kluczniak and Leonard Schild. FDFB: Full domain functional bootstrapping towards practical fully homomorphic encryption. *IACR TCHES*, 2023(1):501–537, 2023.

[LLKN23] Joon-Woo Lee, Eunsang Lee, Young-Sik Kim, and Jong-Seon No. Rotation key reduction for client-server systems of deep neural network on fully homomorphic encryption. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 36–68. Springer, Singapore, December 2023.

[LMK+23] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 227–256. Springer, Cham, April 2023.

[LMP22] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 130–160. Springer, Cham, December 2022.

[LMSS23] Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. Faster TFHE bootstrapping with block binary keys. In Joseph K. Liu, Yang Xiang, Surya Nepal, and Gene Tsudik, editors, *ASIACCS 23*, pages 2–13. ACM Press, July 2023.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608. ACM Press, June 2023.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Berlin, Heidelberg, February 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[LY23] KangHoon Lee and Ji Won Yoon. Discretization error reduction for high precision torus fully homomorphic encryption. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 33–62. Springer, Cham, May 2023.

[MHW+24] Shihe Ma, Tairong Huang, Anyu Wang, Qixian Zhou, and Xiaoyun Wang. Fast and accurate: Efficient full-domain functional bootstrap and digit decomposition for homomorphic computation. *IACR TCHES*, 2024(1):592–616, 2024.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012.

[MP21] Daniele Micciancio and Yuriy Polyakov. Bootstrapping in fhew-like cryptosystems. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Virtual Event, Korea, 15 November 2021*, pages 17–28. ACM, 2021.

[MS18] Daniele Micciancio and Jessica Sorrell. Ring packing and amortized FHEW bootstrapping. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl, July 2018.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Berlin, Heidelberg, May 2016.

[OPP14] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 536–553. Springer, Berlin, Heidelberg, August 2014.

[OPPW24] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. In Jeremy Clark and Elaine Shi, editors, *FC 2024, Part II*, volume 14745 of *LNCS*, pages 264–282. Springer, Cham, March 2024.

[OPPW25] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. On algebraic homomorphic encryption and its applications to doubly-efficient PIR. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VI*, volume 15606 of *LNCS*, pages 34–64. Springer, Cham, May 2025.

[PBT+24] David Du Pont, Jonas Bertels, Furkan Turan, Michiel Van Beirendonck, and Ingrid Verbauwhede. Hardware acceleration of the prime-factor and rader NTT for BGV fully homomorphic encryption. In *31st IEEE Symposium on Computer Arithmetic, ARITH 2024, Malaga, Spain, June 10-12, 2024*, pages 1–8. IEEE, 2024.

[PS24] Alain Passelègue and Damien Stehlé. Low communication threshold fully homomorphic encryption. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 297–329. Springer, Singapore, December 2024.

[RAD78] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *In Foundations of Secure Computation*, pages 169–180, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Berlin, Heidelberg, December 2009.

[SW25] Nigel P. Smart and Michael Walter. Error-simulatable sanitization for TFHE and applications. *IACR Commun. Cryptol.*, 2(3):1, 2025.

[SWA23] Ravital Solomon, Rick Weber, and Ghada Almashaqbeh. smartFHE: Privacy-preserving smart contracts from fully homomorphic encryption. In *2023 IEEE European Symposium on Security and Privacy*, pages 309–331. IEEE Computer Society Press, July 2023.

[TCBS23] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. A homomorphic AES evaluation in less than 30 seconds by means of TFHE. In Michael Brenner, Anamaria Costache, and Kurt Rohloff, editors, *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023*, pages 79–90. ACM, 2023.

[vBDTV23] Michiel van Beirendonck, Jan-Pieter D'Anvers, Furkan Turan, and Ingrid Verbauwhede. FPT: A fixed-point accelerator for torus fully homomorphic encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 741–755. ACM Press, November 2023.

[WWL+24a] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. Cryptology ePrint Archive, Report 2024/323, 2024.

[WWL+24b] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 342–372. Springer, Cham, May 2024.

[XZD+23] Binwu Xiang, Jiang Zhang, Yi Deng, Yiran Dai, and Dengguo Feng. Fast blind rotation for bootstrapping FHEs. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 3–36. Springer, Cham, August 2023.

[YXS+21] Zhaomin Yang, Xiang Xie, Huajie Shen, Shiying Chen, and Jun Zhou. TOTA: Fully homomorphic encryption with smaller parameters and stronger security. Cryptology ePrint Archive, Report 2021/1347, 2021.

[Zam22] Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. `https://github.com/zama-ai/tfhe-rs`.