Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds

Ilaria Chillotti[1], Nicolas Gama[2,1], Mariya Georgieva[3], and Malika Izabachène[4]

[1] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
[2] Inpher, Lausanne, Switzerland
[3] Gemalto, 6 rue de la Verrerie 92190, Meudon, France
[4] CEA LIST, Point Courrier 172, 91191 Gif-sur-Yvette Cedex, France

Abstract. In this paper, we revisit fully homomorphic encryption (FHE) based on GSW and its ring variants. We notice that the internal product of GSW can be replaced by a simpler external product between a GSW and an LWE ciphertext.
We show that the bootstrapping scheme FHEW of Ducas and Micciancio [11] can be expressed only in terms of this external product. As a result, we obtain a speed up from less than 1 second to less than 0.1 seconds. We also reduce the 1GB bootstrapping key size to 24MB, preserving the same security levels, and we improve the noise propagation overhead by replacing exact decomposition algorithms with approximate ones.
Moreover, our external product allows to explain the unique asymmetry in the noise propagation of GSW samples and makes it possible to evaluate deterministic automata homomorphically as in [13] in an efficient way with a noise overhead only linear in the length of the tested word.
Finally, we provide an alternative practical analysis of LWE based scheme, which directly relates the security parameter to the error rate of LWE and the entropy of the LWE secret key.
Keywords: Fully Homomorphic Encryption, Bootstrapping, Lattices, LWE, GSW

1 Introduction

Fully homomorphic encryption (FHE) allows to perform computations over encrypted data without decrypting them. This concept has long been regarded as an open problem until the breakthrough paper of Gentry in 2009 [15] which demonstrates the feasibility of computing any function on encrypted data. Since then, many constructions have appeared involving new mathematical and algorithmic concepts and improving efficiency.

In homomorphic encryption, messages are encrypted with a noise that grows at each homomorphic evaluation of an elementary operation. In a somewhat encryption scheme, the number of homomorphic operations is limited, but can be made asymptotically large using bootstrapping [15]. This technical trick introduced by Gentry allows to evaluate arbitrary circuits by essentially evaluating

the decryption function on encrypted secret keys. This step has remained very costly until the recent paper of Ducas and Micciancio [11], which presented a very fast bootstrapping procedure running in around 0.69 second, making an important step towards practical FHE for arbitrary NAND circuits. In this paper, we further improve the bootstrapping procedure.

We first provide an intuitive formalization of LWE/RingLWE on numbers or polynomials over the real torus, obtained by combining the Scale-Invariant-LWE problem of [9] or the LWE normal form of [10] with the General-LWE problem of Brakerski-Gentry-Vaikutanathan [5]. We call TLWE this unified representation of LWE ciphertexts, which encode polynomials over the Torus. Its security relies either on the hardness of general or ideal lattice reduction, depending on the choice of dimensions. Using the same formalism, we extend the GSW/RingGSW ciphertexts to TGSW, which is the combined analogue of Gentry-Sahai-Water's ciphertexts from [16, 3], and which can also instantiate the ring version used in Ducas-Micciancio scheme [11] in the FHEW cryptosystem. Similarly, a TGSW ciphertext encodes an integer polynomial message, and depending on the choice of dimesions, its security is also based on (worst-case) generic or ideal lattice reduction algorithms. TLWE and TGSW are basically dual to each other, and the main idea of our efficiency result comes from the fact that these two schemes can directly be combined together to map the external product of their two messages into a TLWE sample. Since a TGSW sample is essentially a matrix whose individual rows are TLWE samples, our external product TGSW times TLWE is much quicker than the usual internal product TGSW times TGSW used in previous work. This could mostly be understood as comparing the speed of the computation of a matrix-vector product to a matrix-matrix product. As a result, we obtain a significant improvement (12 times faster) of the most efficient bootstrapping procedure [11]; it now runs in less than 0.052s.

We also analyze the case of leveled encryption. Using an external product means that we lose some composability properties in the design of homomorphic circuits. This corresponds to circuits where boolean gates have different kinds of wires that cannot be freely interconnected. Still, we show that we maintain the expressiveness of the whole binary decision diagram and automata-based logic, which was introduced in [13] with the GSW-GSW internal product, and we tighten the analysis. Indeed, while it was impractical (10 transitions per second in the ring case, and impractical in the non-ring case), we show that the TGSW-TLWE external product enables to evaluate up to 5000 transitions per second, in a leveled homomorphic manner. We also refine the mapping between automata and homomorphic gates, and reduce the number of homomorphic operations to test a word with a deterministic automata. This allows to compile and evaluate constant-time algorithms (i.e. with data-independent control flow) in a leveled homomorphic manner, with only sub-linear noise overhead in the running time.

We also propose a new security analysis where the security parameter is directly expressed as a function of the entropy of the secret and the error rate. For the parameters that we propose in our implementation, we predict 188-bits of security for both the bootstrapping key and the keyswitching key.

Roadmap. In Section 2, we give mathematical definitions and a quick overview of the classical version of LWE-based schemes. In Section 3, we generalize LWE and GSW schemes using a *torus representation* of the samples. We also review the arithmetic operations over the torus and introduce our main theorem characterizing the new morphism between TLWE and TGSW. As a proof of concept, we present two main applications in Section 4 where we explain our fast bootstrapping procedure, and in Section 5, we present efficient leveled evaluation of deterministic automata, and apply it on a constant-time algorithm with logarithmic memory. Finally, we provide a practical security analysis in Section 6.

2 Background

Notation. In the rest of the paper we will use the following notations. The security parameter will be denoted as λ. The set $\{0,1\}$ (without any structure) will be written $\mathbb{B}$. The real Torus $\mathbb{R}/\mathbb{Z}$, called $\mathbb{T}$ set of real numbers modulo 1. $\mathfrak{R}$ denotes the ring of polynomials $\mathbb{Z}[X]/(X^N+1)$. $\mathbb{T}_N[X]$ denotes $\mathbb{R}[X]/(X^N+1)$ mod 1. Finally, we note by $\mathcal{M}_{p,q}(E)$ the set of matrices $p \times q$ with entries in E.

This section combines some algebra theory, namely abelian groups, commutative rings, R-modules, and on some metrics of the continuous field $\mathbb{R}$.

Definition 2.1 (R-module). *Let $(R,+,\times)$ be a commutative ring. We say that a set M is a R-module when $(M,+)$ is an abelian group, and when there exists an external operation $\cdot$ which is bi-distributive and homogeneous. Namely, $\forall r,s \in R$ and $x,y \in M$, $1_R \cdot x = x$, $(r+s)\cdot x = r\cdot x + s\cdot x$, $r\cdot(x+y) = r\cdot x + r\cdot y$, and $(r\times s)\cdot x = r\cdot(s\cdot x)$.*

Any abelian group is by construction a $\mathbb{Z}$-module for the iteration (or exponentiation) of its own law. In this paper, one of the most important abelian group we use is the real torus $\mathbb{T}$, composed of all reals modulo 1 ($\mathbb{R} \mod 1$). The torus is not a ring, since the real internal product is not compatible with the modulo 1 projection (expressions like $0 \times \frac{1}{2}$ are undefined). But as an additive group, it is a $\mathbb{Z}$-module, and the external product $\cdot$ from $\mathbb{Z}\times\mathbb{T}$ to $\mathbb{T}$, like in $0\cdot\frac{1}{2} = 0$, is well defined. More importantly, we recall that for all positive integers N and k, $(\mathbb{T}_N[X]^k, +, \bullet)$ is a $\mathfrak{R}$-module.

A R-module M shares many arithmetic operations and constructions with vector spaces: vectors M^n or matrices $\mathcal{M}_{n,m}(M)$ are also R-modules, and their left dot product with a vector in R^n or left matrix product in $\mathcal{M}_{k,n}(R)$ are both well defined.

Gaussian Distributions Let $\sigma \in \mathbb{R}^+$ be a parameter and $k \geq 1$ the dimension. For all $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^k$, we note $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \exp(-\pi \left\|\boldsymbol{x}-\boldsymbol{c}\right\|^2 / \sigma^2)$. If $\boldsymbol{c}$ is omitted, then it is implicitly 0. Let S be a subset of $\mathbb{R}^k$, $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{S})$ denotes $\sum_{x\in S}\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})$ or $\int_{x\in S}\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x}).d\boldsymbol{x}$. For all closed (continuous or discrete) additive subgroup $M \subseteq \mathbb{R}^k$, then $\rho_{\sigma,\boldsymbol{c}}(M)$ is finite, and defines a *(restricted) Gaussian Distribution* of parameter σ, standard deviation $\sqrt{2/\pi}\sigma$ and center $\boldsymbol{c}$ over M, with the density

function $\mathcal{D}_{M,\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})/\rho_{\sigma,\boldsymbol{c}}(M)$. Let L be a discrete subgroup of M, then the *Modular Gaussian distribution* over M/L exists and is defined by the density $\mathcal{D}_{M/L,\sigma,\boldsymbol{c}}(\boldsymbol{x}) = \mathcal{D}_{M,\sigma,\boldsymbol{c}}(\boldsymbol{x}+L)$. Furthermore, when $\mathrm{span}(M) = \mathrm{span}(L)$, then M/L admits a uniform distribution of constant density $\mathcal{U}_{M/L}$. In this case, the *smoothing parameter* $\eta_{M,\varepsilon}(L)$ of L in M is defined as the smallest $\sigma \in \mathbb{R}$ such that $\sup_{x\in M} |\mathcal{D}_{M/L,\sigma,\boldsymbol{c}}(\boldsymbol{x}) - \mathcal{U}_{M/L}| \leq \varepsilon \cdot \mathcal{U}_{M/L}$. If M is omitted, it implicitely means $\mathbb{R}^k$.

Subgaussian Distributions A distribution X over $\mathbb{R}$ is σ-subgaussian iff it satisfies the Laplace-transformation bound: $\forall t \in \mathbb{R}, \mathbb{E}(\exp(tX)) \leq \exp(\sigma^2 t^2/2)$. By Markov's inequality, this implies that the tails of X are bounded by the Gaussian function of standard deviation σ: $\forall x > 0, \mathbb{P}(|X| \geq x) \leq 2\exp(-x^2/2\sigma^2)$. As an example, the Gaussian distribution of standard deviation σ (i.e. parameter $\sqrt{\pi/2}\sigma$), the equi-distribution on $\{-\sigma, \sigma\}$, and the uniform distribution over $[-\sqrt{3}\sigma, \sqrt{3}\sigma]$, which all have standard deviation σ, are σ-subgaussian[5]. If X and X' are two independent σ and σ'-subgaussian variables, then for all $\alpha, \beta \in \mathbb{R}$, $\alpha X + \beta X'$ is $\sqrt{\alpha^2\sigma^2 + \beta^2\sigma'^2}$-subgaussian.

Distance and Norms We use the standard $\|\cdot\|_p$ and $\|\cdot\|_\infty$ norms for scalars and vectors over the real field or over the integers. By extension, the norm $\|P(X)\|_p$ of a real or integer polynomial $P \in \mathbb{R}[X]$ is the norm of its coefficient vector. If the polynomial is modulo $X^N + 1$, we take the norm of its unique representative of degree $\leq N-1$.

By abuse of notation, we write $\|\boldsymbol{x}\|_p = \min_{\boldsymbol{u}\in\boldsymbol{x}+\mathbb{Z}^k}(\|u\|_p)$ for all $\boldsymbol{x} \in \mathbb{T}^k$. It is the p-norm of the representative of $\boldsymbol{x}$ with all coefficients in $]-\frac{1}{2}, \frac{1}{2}]$. Although it satisfies the separation and the triangular inequalities, this notation is not a norm, because it lacks homogeneity[6], and $\mathbb{T}^k$ is not a vector space either. But we have $\forall m \in \mathbb{Z}, \|m \cdot \boldsymbol{x}\|_p \leq |m| \|\boldsymbol{x}\|_p$. By extension, we define $\|a\|_p$ for a polynomial $a \in \mathbb{T}_N[X]$ as the p- norm of its unique representative in $\mathbb{R}[X]$ of degree $\leq N-1$ and with coefficients in $]-\frac{1}{2}, \frac{1}{2}]$.

Definition 2.2 (Infinity norm over $\mathcal{M}_{p,q}(\mathbb{T}_N[X])$). *Let $A \in \mathcal{M}_{p,q}(\mathbb{T}_N[X])$. We define the infinity norm of A as*

$$\|A\|_\infty = \max_{\substack{i\in[\![1,p]\!]\\ j\in[\![1,q]\!]}} \|a_{i,j}\|_\infty .$$

Concentrated distribution on the Torus, Expectation and Variance A distribution $\mathcal{X}$ on the torus is *concentrated* iff. its support is included in a

[5] For the first two distributions, it is tight, but the uniform distribution over $[-\sqrt{3}\sigma, \sqrt{3}\sigma]$ is even 0.78σ-subgaussian

[6] Mathematically speaking, a more accurate notion would be $\mathrm{dist}_p(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p$, which is a distance. However, the norm symbol is clearer for almost all practical purposes.

ball of radius $\frac{1}{4}$ of $\mathbb{T}$, except for negligible probability. In this case, we define the *variance* $\mathsf{Var}(\mathcal{X})$ and the expectation $\mathbb{E}(\mathcal{X})$ of $\mathcal{X}$ as respectively $\mathsf{Var}(\mathcal{X}) = \min_{\bar{x}\in\mathbb{T}} \sum p(x)|x-\bar{x}|^2$ and $\mathbb{E}(\mathcal{X})$ as the position $\bar{x} \in \mathbb{T}$ which minimizes this expression. By extension, we say that a distribution $\mathcal{X}'$ over $\mathbb{T}^n$ or $\mathbb{T}_N[X]^k$ is concentrated iff. each coefficient has an independent concentrated distribution on the torus. Then the expectation $\mathbb{E}(\mathcal{X}')$ is the vector of expectations of each coefficient, and $\mathsf{Var}(\mathcal{X}')$ denotes the maximum of each coefficient's Variance.

These expectation and variance over $\mathbb{T}$ follow the same linearity rules than their classical equivalent over the reals.

Fact 2.3. Let $\mathcal{X}_1, \mathcal{X}_2$ be two independent concentrated distributions on either $\mathbb{T}, \mathbb{T}^n$ or $\mathbb{T}_N[X]^k$, and $e_1, e_2 \in \mathbb{Z}$ such that $\mathcal{X} = e_1 \cdot \mathcal{X}_1 + e_2 \cdot \mathcal{X}_2$ remains concentrated, then $\mathbb{E}(\mathcal{X}) = e_1 \cdot \mathbb{E}(\mathcal{X}_1) + e_2 \cdot \mathbb{E}(\mathcal{X}_2)$ and $\mathsf{Var}(\mathcal{X}) \leq e_1^2 \cdot \mathsf{Var}(\mathcal{X}_1) + e_2^2 \cdot \mathsf{Var}(\mathcal{X}_2)$.

Also, subgaussian distributions with small enough parameters are necessarily concentrated:

Fact 2.4. Every distribution $\mathcal{X}$ on either $\mathbb{T}, \mathbb{T}^n$ or $\mathbb{T}_N[X]^k$ where each coefficient is σ-subgaussian where $\sigma \leq 1/\sqrt{32\log(2)(\lambda+1)}$ is a concentrated distribution: a fraction $1-2^{-\lambda}$ of its mass is in the interval $[-\frac{1}{4}, \frac{1}{4}]$.

2.1 Learning With Error problem

The Learning With Errors (LWE) problem was introduced by Regev in 2005 [21]. The Ring variant, called RingLWE, was introduced by Lyubashevsky, Peikert and Regev in 2010 [19]. Both variants are nowadays extensively used for the construction of lattice-based Homomorphic Encryption schemes. In the original definition [21], a LWE sample has its right member on the torus and is defined using continuous Gaussian distributions. Here, we will work entirely on the real torus, employing the same formalism as the Scale Invariant LWE (SILWE) scheme in [9], or LWE scale-invariant normal form in [10]. Without loss of generality, we refer to it as LWE.

Definition 2.5 ((Homogeneous) LWE). *Let $n \geq 1$ be an integer, $\alpha \in \mathbb{R}^+$ be a noise parameter and $\boldsymbol{s}$ be a uniformly distributed secret in some bounded set $\mathcal{S} \in \mathbb{Z}^n$. Denote by $\mathcal{D}^{\mathsf{LWE}}_{\boldsymbol{s},\alpha}$ the distribution over $\mathbb{T}^n \times \mathbb{T}$ obtained by sampling a couple $(\boldsymbol{a}, b)$, where the left member $\boldsymbol{a} \in \mathbb{T}^n$ is chosen uniformly random and the right member $b = \boldsymbol{a} \cdot \boldsymbol{s} + e$. The error e is a sample from a gaussian distribution with parameter α.*

- *Search problem: given access to polynomially many* LWE *samples, find $s \in \mathcal{S}$.*
- *Decision problem: distinguish between* LWE *samples and uniformly random samples from $\mathbb{T}^n \times \mathbb{T}$.*

Both the LWE search or decision problems are reducible to each other, and their average case is asymptotically as hard as worst-case lattice problems. In practice, both problems are also intractable, and their hardness increases with the the entropy of the key set $\mathcal{S}$ (*i.e.* n if keys are binary) and $\alpha \in]0, \eta_\varepsilon(\mathbb{Z})[$.

Regev's encryption scheme [21] is the following: Given a discrete message space $\mathcal{M} \in \mathbb{T}$, for instance $\{0, \frac{1}{2}\}$, a message $\mu \in \mathcal{M}$ is encrypted by summing up the *trivial* LWE sample $(\mathbf{0}, \mu)$ of μ to a Homogeneous LWE sample $(\boldsymbol{a}, b) \in \mathbb{T}^{n+1}$ with respect to a secret key $\boldsymbol{s} \in \mathbb{B}^n$ and a noise parameter $\alpha \in \mathbb{R}^+$. The semantic security of the scheme is equivalent to the LWE decisional problem. The decryption of a sample $\boldsymbol{c} = (\boldsymbol{a}, b)$ consists in computing this quantity $\varphi_s(\boldsymbol{a}, b) = b - \boldsymbol{s} \cdot \boldsymbol{a}$, which we call the *phase* of $\boldsymbol{c}$, and to round it to the nearest element in $\mathcal{M}$. Decryption is correct with overwhelming probability $1 - 2^{-p}$ provided that the parameter α is $O(R/\sqrt{p})$ where R is the packing radius of $\mathcal{M}$.

3 Generalization

In this section we extend this presentation to rings, following the generalization of [5], and also to GSW [16].

3.1 TLWE

We first define TLWE samples, together with the search and decision problems. In the following, ciphertexts are viewed as normal samples.

Definition 3.1 (TLWE samples). *Let $k \geq 1$ be an integer, N a power of 2, and $\alpha \geq 0$ be a noise parameter. A* TLWE *secret key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ is a vector of k polynomials $\in \mathfrak{R} = \mathbb{Z}[X]/X^N + 1$ with binary coefficients. For security purposes, we assume that private keys are uniformly chosen, and that they actually contain $n \approx Nk$ bits of entropy. The message space of* TLWE *samples is $\mathbb{T}_N[X]$. A fresh* TLWE *sample of a message $\mu \in \mathbb{T}_N[X]$ with noise parameter α under the key $\boldsymbol{s}$ is an element $(\boldsymbol{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$, $b \in \mathbb{T}_N[X]$ has Gaussian distribution $\mathcal{D}_{\mathbb{T}_N[X],\alpha,\boldsymbol{s}\cdot\boldsymbol{a}+\mu}$ around $\mu + \boldsymbol{s} \cdot \boldsymbol{a}$. The sample is* random *iff its left member $\boldsymbol{a}$ (also called mask) is uniformly random $\in \mathbb{T}_N[X]^k$ (or a sufficiently dense submodule* [7] *),* trivial *if $\boldsymbol{a}$ is fixed to $\mathbf{0}$,* noiseless *if $\alpha = 0$, and* homogeneous *iff its message μ is 0.*

- *Search problem: given access to polynomially many fresh random homogeneous* TLWE *samples, find their key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$.*
- *Decision problem: distinguish between fresh random homogeneous* TLWE *samples from uniformly random samples from $\mathbb{T}_N[X]^{k+1}$.*

This definition is the analogue on the torus of the Gereral-LWE problem of [5]. It allows to consider both LWE and RingLWE as a single problem. Choosing N large and $k = 1$ corresponds to the classical (bin)RingLWE (over cyclotomic

[7] A submodule G is sufficiently dense if there exists an intermediate submodule H such that $G \subseteq H \subseteq \mathbb{T}^n$, the relative smoothing parameter $\eta_{H,\varepsilon}(G)$ is $\leq \alpha$, and H is the orthogonal in $\mathbb{T}^n$ of at most $n - 1$ vectors of $\mathbb{Z}^n$. This definition allows to convert any (Ring)-LWE with non-binary secret to a TLWE instance via binary decomposition.

rings, and up to a scaling factor q). When $N = 1$ and k large, then $\mathfrak{R}$ and $\mathbb{T}_N[X]$ respectively collapses to $\mathbb{Z}$ and $\mathbb{T}$, and TLWE is simply bin-LWE (up to the same scaling factor q). Other choices of N, k give some continuum between the two extremes, with a security that varies between worst-case ideal lattices to worst-case regular lattices.

Thanks to the underlying $\mathfrak{R}$-module structure, we can sum TLWE samples, or we can make integer linear or polynomial combinations of samples with coefficients in $\mathfrak{R}$. However, each of these combinations increases the noise inside the samples. They are therefore limited to small coefficients.

We additionally define a function called the phase of a TLWE sample, that will be used many times. The phase computation is the first step of the classical decryption algorithm, and uses the secret key.

Definition 3.2 (Phase). *Let $\boldsymbol{c} = (\boldsymbol{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$ and $\boldsymbol{s} \in \mathbb{B}_N[X]^k$, we define the* phase *of the sample as $\varphi_{\boldsymbol{s}}(\boldsymbol{c}) = b - \boldsymbol{s} \bullet \boldsymbol{a}$.*
The phase is linear over $\mathbb{T}_N[X]^{k+1}$ and is $(kN+1)$-lipschitzian for the ℓ_∞ distance: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{T}_N[X]^{k+1}, \|\varphi_{\boldsymbol{s}}(\boldsymbol{x}) - \varphi_{\boldsymbol{s}}(\boldsymbol{y})\|_\infty \leq (kN+1) \|\boldsymbol{x} - \boldsymbol{y}\|_\infty$.

Note that a TLWE sample contains noise, that its semantic is only function of its phase, and that the phase has the nice property to be lipschitzian. Together, these properties have many interesting implications. In particular, we can always work with approximations, since two samples at a short distance on $\mathbb{T}_N[X]^{k+1}$ share the same properties: they encode the same message, and they can in general be swapped. This fact explains why we can work and describe our algorithms on the infinite Torus.

Given a finite message space $\mathcal{M} \subseteq \mathbb{T}_N[X]$, the (classical) decryption algorithm computes the phase $\varphi_s(\boldsymbol{c})$ of the sample, and returns the closest $\mu \in \mathcal{M}$. It is easy to see that if $\boldsymbol{c}$ is a fresh TLWE sample of $\mu \in \mathcal{M}$ with gaussian noise parameter α, the decryption of $\boldsymbol{c}$ over $\mathcal{M}$ is equal to μ as soon as α is $\Theta(\sqrt{\lambda})$ times smaller than the packing radius of $\mathcal{M}$. However decryption is harder to define for non-fresh samples. In this case, correctness of the decryption procedure involves a recurrence formula between the decryption of the sum and the sum of the decryption of the inputs conditioned by the noise parameters. In addition, message spaces of the input samples can be in different subgroups of $\mathbb{T}$. To raise the limitations of the decryption function, we will instead use a mathematical definition of message and error by reasoning directly on the following Ω-probability space.

Definition 3.3 (The Ω-probability space). *Since samples are either independent (random, noiseless, or trivial) fresh $\boldsymbol{c} \leftarrow TLWE_{\boldsymbol{s},\alpha}(\mu)$, or linear combination $\tilde{\boldsymbol{c}} = \sum_{i=1}^{p} e_i \cdot \boldsymbol{c_i}$ of other samples, the probability space Ω is the product of the probability spaces of each individual fresh samples $\boldsymbol{c}$ with the TLWE distributions defined in definitions 3.1, and of the probability spaces of all the coefficients $(e_1, \ldots, e_p) \in \mathfrak{R}^p$ or $\mathbb{Z}^p$ that are obtained with randomized algorithm.*

In other words, instead of viewing a TLWE sample as a fixed value which is the result of one particular event in Ω, we will consider all the possible values at once, and make statistics on them.

We now define functions on TLWE samples: message, error, noise variance, and noise norm. These functions are well defined mathematically, and can be used in the analysis of various algorithms. However, they cannot be directly computed or approximated in practice.

Definition 3.4. *Let $\boldsymbol{c}$ be a random variable $\in \mathbb{T}_N[X]^{k+1}$, which we'll interpret as a* TLWE *sample. All probabilities are on the Ω-space. We say that $\boldsymbol{c}$ is a* valid *TLWE sample iff there exists a key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ such that the distribution of the phase $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$ is concentrated. If $\boldsymbol{c}$ is trivial, all keys $\boldsymbol{s}$ are equivalent, else the mask of $\boldsymbol{c}$ is uniformly random, so $\boldsymbol{s}$ is unique. We then define:*

- *the* message *of $\boldsymbol{c}$, denoted as $\mathsf{msg}(\boldsymbol{c}) \in \mathbb{T}_N[X]$ is the expectation of $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$;*
- *the* error*, denoted $\mathsf{Err}(\boldsymbol{c})$, is equal to $\varphi_{\boldsymbol{s}}(\boldsymbol{c}) - \mathsf{msg}(\boldsymbol{c})$;*
- *$\mathsf{Var}(\mathsf{Err}(\boldsymbol{c}))$ denotes the variance of $\mathsf{Err}(\boldsymbol{c})$, which is by definition also equal to the variance of $\varphi_{\boldsymbol{s}}(\boldsymbol{c})$;*
- *finally, $\|\mathsf{Err}(\boldsymbol{c})\|_\infty$ denotes the maximum amplitude of $\mathsf{Err}(\boldsymbol{c})$ (possibly with overwhelming probability).*

Unlike the classical decryption algorithm, the message function can be viewed as an ideal black box decryption function, which works with infinite precision even if the message space is continuous. Provided that the noise amplitude remains smaller than $\frac{1}{4}$, the message function is perfectly linear. Using these intuitive and intrinsic functions will considerably ease the analysis of all algorithms in this paper. In particular, we have:

Fact 3.5. Given p valid and independent TLWE samples $\boldsymbol{c_1}, \ldots, \boldsymbol{c_p}$ under the same key $\boldsymbol{s}$, and p integer polynomials $e_1, \ldots, e_p \in \mathfrak{R}$, if the linear combination $\boldsymbol{c} = \sum_{i=1}^p e_i \bullet \boldsymbol{c_i}$ is a valid TLWE sample, it satisfies: $\mathsf{msg}(\boldsymbol{c}) = \sum_{i=1}^p e_i \bullet \mathsf{msg}(\boldsymbol{c}_i)$, with variance $\mathsf{Var}(\mathsf{Err}(\boldsymbol{c})) \leq \sum_{i=1}^p \|e_i\|_2^2 \cdot \mathsf{Var}(\mathsf{Err}(\boldsymbol{c_i}))$ and noise amplitude $\|\mathsf{Err}(\boldsymbol{c})\|_\infty \leq \sum_{i=1}^p \|e_i\|_1 \cdot \|\mathsf{Err}(\boldsymbol{c_i})\|_\infty$. If the last bound is $< \frac{1}{4}$, then $\boldsymbol{c}$ is necessarily a valid TLWE sample (under the same key $\boldsymbol{s}$).

In order to characterize the average case behaviour of our homomorphic operations, we shall rely on the heuristic assumption of independence below. This heuristic will only be used for practical average-case bounds. Our worst-case theorems and lemma based on the infinite norm do not use it at all.

Assumption 3.6 (Independence Heuristic). All the coefficients of the error of TLWE or TGSW samples that occur in all the linear combinations we consider are independent and concentrated. More precisely, they are σ-subgaussian where σ is the square-root of their variance.

This assumption allows us to bound the variance of the noise instead of its norm, and to provide realistic average-case bounds which often correspond to the square root of the worst-case ones. The error can easily be proved subgaussian, since each coefficients are always obtained by convolving Gaussians or zero-centered bounded uniform distributions. But the independence assumption between all the coefficients remains heuristic. Dependencies between coefficients may affect the variance of their combinations in both directions. The

independence of coefficients can be obtained by adding enough entropy in all our decomposition algorithms and by increasing some parameters accordingly, but as noticed in [11], this work-around seems more as a proof artefact, and is experimentally not needed. Since average case corollaries should reflect practical results, we leave the independence of subgaussian samples as a heuristic assumption.

3.2 TGSW

In this section we present a generalized scale invariant version of the FHE scheme GSW [16], that we call TGSW. GSW was proposed Gentry, Sahai and Waters in 2013 [16], and improved in [3] and its security is based on the LWE problem. The scheme relies on a gadget decomposition function, which we also extend to polynomials, but most importantly, the novelty is that our function is an approximate decomposition, up to some precision parameter. This allows to improve running time and memory requirements for a small amount of additional noise.

Definition 3.7 (Approximate Gadget Decomposition). *Let* $\boldsymbol{h} \in \mathcal{M}_{p,k+1}(\mathbb{T}_N[X])$ *as in (1). We say that* $Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{v})$ *is a decomposition algorithm on the gadget* $\boldsymbol{h}$ *with quality* β *and precision* ϵ *if and only if for any* TLWE *sample* $\boldsymbol{v} \in \mathbb{T}_N[X]^{k+1}$*, it efficiently and publicly outputs a small vector* $\boldsymbol{u} \in \mathfrak{R}^{(k+1)\ell}$ *such that* $\|\boldsymbol{u}\|_\infty \leq \beta$ *and* $\|\boldsymbol{u} \cdot \boldsymbol{h} - \boldsymbol{v}\|_\infty \leq \epsilon$*. Furthermore, the expectation of* $\boldsymbol{u} \cdot \boldsymbol{h} - \boldsymbol{v}$ *must to be* 0 *when* $\boldsymbol{v}$ *is uniformly distributed in* $\mathbb{T}_N[X]^{k+1}$

Definition 3.7 is generic, but in the rest of the paper, we will only use this fixed gadget:

$$\boldsymbol{h} = \left(\begin{array}{c|c|c} 1/B_g & \dots & 0 \\ \vdots & \ddots & \vdots \\ 1/B_g^\ell & \dots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \dots & 1/B_g \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1/B_g^\ell \end{array}\right) \in \mathcal{M}_{p,k+1}(\mathbb{T}_N[X]). \tag{1}$$

The matrix $\boldsymbol{h}$ consists in a diagonal of columns, each containing a super-increasing sequence of constant polynomials in $\mathbb{T}$. Algorithm 1 represents an efficient decomposition of TLWE samples on $\boldsymbol{h}$, and the following lemma proves its correctness. In theory, decomposition algorithms should be randomized to guarantee that the distribution of all error coefficients remain independent. In practice, we already rely on Heuristic 3.6. We just need that the expectation of the small errors induced by the approximations remains null, so that the message is not changed.

Lemma 3.8. *Let* $\ell \in \mathbb{N}$ *and* $B_g \in \mathbb{N}$*. Then for* $\beta = B_g/2$ *and* $\epsilon = 1/2B_g^\ell$*, Algorithm 1 is a valid* $Dec_{\boldsymbol{h},\beta,\epsilon}$*.*

Algorithm 1 Gadget Decomposition of a TLWE sample

Input: A TLWE sample $(a,b) = (a_1, \ldots, a_k, b = a_{k+1}) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$
Output: A combination $[e_{1,1}, \ldots, e_{k+1,\ell}] \in \mathfrak{R}^{(k+1)\ell}$

1: For each a_i choose the unique representative $\sum_{j=0}^{N-1} a_{i,j} X^j$, with $a_{i,j} \in \mathbb{T}$, and set $\bar{a}_{i,j}$ the closest multiple of $\frac{1}{B_g^\ell}$ to $a_{i,j}$
2: Decompose each $\bar{a}_{i,j}$ uniquely as $\sum_{p=1}^{\ell} \bar{a}_{i,j,p} \frac{1}{B_g^p}$ where each $\bar{a}_{i,j,p} \in [\![-B_g/2, B_g/2[\![$
3: **for** $i = 1$ **to** $k+1$
4: **for** $p = 1$ **to** ℓ
5: $e_{i,p} = \sum_{j=0}^{N-1} \bar{a}_{i,j,p} X^j \in \mathfrak{R}$
6: Return $(e_{i,p})_{i,p}$

Proof. Let $\boldsymbol{v} = (a,b) = (a_1, \ldots, a_k, b = a_{k+1}) \in \mathbb{T}_N[X]^{k+1}$ be a TLWE sample, given as input to Algorithm 1. Let $\boldsymbol{u} = [e_{1,1}, \ldots, e_{k+1,\ell}] \in \mathfrak{R}^{(k+1)\ell}$ be the corresponding output by construction $\|\boldsymbol{u}\|_\infty \leq B_g/2 = \beta$.
Let $\boldsymbol{\epsilon_{\mathbf{dec}}} = \boldsymbol{u} \cdot \boldsymbol{h} - \boldsymbol{v}$. For all $i \in [\![1, k+1]\!]$ and $j \in [\![1, \ell]\!]$, we have by construction $\boldsymbol{\epsilon_{\mathbf{dec}}}_{i,j} = \sum_{p=0}^{\ell} e_{i,p} \bullet \frac{1}{B_g^p} - a_{i,j} = \bar{a}_{i,j} - a_{i,j}$. Since $\bar{a}_{i,j}$ is defined as the nearest multiple of $\frac{1}{B_g^\ell}$ on the torus, we have $|\bar{a}_{i,j} - a_{i,j}| \leq 1/2B_g^\ell = \epsilon$. $\boldsymbol{\epsilon_{\mathbf{dec}}}$ has therefore a concentrated distribution when $\boldsymbol{v}$ is uniform. We now verify that it is zero-centered. Finally, if we call f the function from $\mathbb{T}$ to $\mathbb{T}$ which rounds an element x to its closest multiple of $\frac{1}{B_g^\ell}$ and the funtion g the symmetry defined by $g(x) = 2f(x) - x$ on the torus; we easily verify that the $\mathbb{E}(\boldsymbol{\epsilon_{\mathbf{dec}}}_{i,j})$ is equal to $\mathbb{E}(a_{i,j} - f(a_{i,j}))$ when $a_{i,j}$ has uniform distribution, which is equal to $\mathbb{E}(g(a_{i,j}) - f(g(a_{i,j})))$ when $g(a_{i,j})$ has uniform distribution also equal to $\mathbb{E}(f(a_{i,j}) - a_{i,j}) = -\mathbb{E}(\boldsymbol{\epsilon_{\mathbf{dec}}}_{i,j})$. Thus, the expectation of $\boldsymbol{\epsilon_{\mathbf{dec}}}$ is 0. □

We are now ready to define TGSW samples, and to extend the notions of phase of valid sample, message and error of the samples.

Definition 3.9 (TGSW samples). *Let ℓ and $k \geq 1$ be two integers, $\alpha \geq 0$ be a noise parameter and $\boldsymbol{h}$ the gadget defined in Equation (1). Let $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ be a* RingLWE *key, we say that $\boldsymbol{C} \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ is a fresh* TGSW *sample of $\mu \in \mathfrak{R}/\boldsymbol{h}^\perp$ with noise parameter α iff $\boldsymbol{C} = \boldsymbol{Z} + \mu \bullet \boldsymbol{h}$ where each row of $\boldsymbol{Z} \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ is an Homogeneous TLWE sample (of 0) with Gaussian noise parameter α. Reciprocally, we say that an element $\boldsymbol{C} \in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ is a valid TGSW sample iff there exists a unique polynomial $\mu \in \mathfrak{R}/\boldsymbol{h}^\perp$ and a unique key $\boldsymbol{s}$ such that each row of $\boldsymbol{C} - \mu \bullet \boldsymbol{h}$ is a valid TLWE sample of 0 for the key $\boldsymbol{s}$. We call the polynomial μ the message of $\boldsymbol{C}$, and we denote it by $\mathsf{msg}(C)$.*

Definition 3.10 (Phase, Error). *Let $A =\in \mathcal{M}_{(k+1)\ell,k+1}(\mathbb{T}_N[X])$ be a* TGSW *sample for a secret key $\boldsymbol{s} \in \mathbb{B}_N[X]^k$ and noise parameter $\alpha \geq 0$.*
We define the phase of A, denoted as $\varphi_{\boldsymbol{s}}(A) \in (\mathbb{T}_N[X])^{(k+1)\ell}$, as the list of the $(k+1)\ell$ TLWE *phases of each line of A. In the same way, we define the error of A, denoted $\mathsf{Err}(A)$, as the list of the $(k+1)\ell$* TLWE *errors of each line of A.*

Since TGSW samples are essentially vectors of TLWE samples, they are naturally compatible with linear operations. And both phase and message functions remain linear.

Fact 3.11. Given p valid TGSW samples $C_1, \ldots, C_p$ of messages $\mu_1, \ldots, \mu_p$ under the same key, and with independent error coefficients, and given p integer polynomials $e_1, \ldots, e_p$, the linear combination $C = \sum_{i=1}^{p} e_i \bullet C_i$ is a sample of $\mu = \sum_{i=1}^{p} e_i \cdot \mu_i$, with variance $\mathsf{Var}(C) = \left(\sum_{i=1}^{p} \|e_i\|_2^2 \cdot \mathsf{Var}(C_i)\right)^{1/2}$ and noise infinity norm $\|\mathsf{Err}(C)\|_\infty = \sum_{i=1}^{p} \|e_i\|_1 \cdot \|\mathsf{Err}(C)\|_\infty$.

Also, the phase remains $1 + kN$ lipschitzian for the infinity norm.

Fact 3.12. For all $A \in \mathcal{M}_{p,k+1}(\mathbb{T}_N[X])$, $\|\varphi_{\boldsymbol{s}}(A)\|_\infty \leq (Nk+1)\|A\|_\infty$.

We finally define the homomorphic product between TGSW and TLWE samples, whose corresponding message is simply the product of the two messages of the initial samples. Since the left member encodes an integer polynomial, and the right one a torus polynomial, this operator performs a homomorphic evaluation of their external product. Theorem 3.14 (resp. Theorem 3.15) analyzes the worst-case (resp. average-case) noise propagation of this product. Then, corollary 3.16 relates this new morphism to the classical internal product between TGSW samples.

Definition 3.13 (External product). *We define the product $\boxdot$ as*

$$\begin{aligned} \boxdot\colon \mathrm{TGSW} \times \mathrm{TLWE} &\longrightarrow \mathrm{TLWE} \\ (A, \boldsymbol{b}) &\longmapsto A \boxdot \boldsymbol{b} = Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{b}) \cdot A. \end{aligned}$$

The formula is almost identical to the classical product defined in the original GSW scheme in [16], except that only one vector needs to be decomposed. For this reason, we get almost the same noise propagation formula, with an additional term that comes from the approximations in the decomposition.

Theorem 3.14 (Worst-case External Product). *Let A be a valid* TGSW *sample of message μ_A and let $\boldsymbol{b}$ be a valid* TLWE *sample of message $\mu_{\boldsymbol{b}}$. Then $A \boxdot \boldsymbol{b}$ is a* TLWE *sample of message $\mu_A \cdot \mu_{\boldsymbol{b}}$ and $\|\mathsf{Err}(A \boxdot \boldsymbol{b})\|_\infty \leq (k+1)\ell N\beta \|\mathsf{Err}(A)\|_\infty + \|\mu_A\|_1 (1 + kN)\epsilon + \|\mu_A\|_1 \|\mathsf{Err}(\boldsymbol{b})\|_\infty$ (worst case), where β and ϵ are the parameters used in the decomposition $Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{b})$. If $\|\mathsf{Err}(A \boxdot \boldsymbol{b})\|_\infty \leq 1/4$ we are guaranteed that $A \boxdot \boldsymbol{b}$ is a valid* TLWE *sample.*

Proof. As $A = \mathrm{TGSW}(\mu_A)$, then by definition it is equal to $A = Z_A + \mu_A \cdot \boldsymbol{h}$, where Z_A is a TGSW encryption of 0 and $\boldsymbol{h}$ is the gadget matrix. In the same way, as $\boldsymbol{b} = \mathrm{TLWE}(\mu_{\boldsymbol{b}})$, then by definition it is equal to $\boldsymbol{b} = \boldsymbol{z_b} + (\boldsymbol{0}, \mu_{\boldsymbol{b}})$, where $\boldsymbol{z_b}$ is a TLWE encryption of 0. Let

$$\begin{cases} \|\mathsf{Err}(A)\|_\infty = \|\varphi_{\boldsymbol{s}}(Z_A)\|_\infty = \eta_A \\ \|\mathsf{Err}(\boldsymbol{b})\|_\infty = \|\varphi_{\boldsymbol{s}}(\boldsymbol{z_b})\|_\infty = \eta_{\boldsymbol{b}}. \end{cases}$$

Let $\boldsymbol{u} = Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{b}) \in \mathfrak{R}^{(k+1)\ell}$. By definition $A \boxdot \boldsymbol{b}$ is equal to

$$\begin{aligned} A \boxdot \boldsymbol{b} &= \boldsymbol{u} \cdot A \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot (\boldsymbol{u} \cdot \boldsymbol{h}). \end{aligned}$$

From definition 3.7, we have that $\boldsymbol{u} \cdot \boldsymbol{h} = \boldsymbol{b} + \boldsymbol{\epsilon_{dec}}$, where $\|\boldsymbol{\epsilon_{dec}}\|_\infty = \|\boldsymbol{u} \cdot \boldsymbol{h} - \boldsymbol{b}\|_\infty \leq \epsilon$. So

$$\begin{aligned} A \boxdot \boldsymbol{b} &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot (\boldsymbol{b} + \boldsymbol{\epsilon_{dec}}) \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot \boldsymbol{\epsilon_{dec}} + \mu_A \cdot \boldsymbol{z_b} + (\boldsymbol{0}, \mu_A \cdot \mu_{\boldsymbol{b}}). \end{aligned}$$

Then the phase (linear function) of $A \boxdot \boldsymbol{b}$ is

$$\varphi_{\boldsymbol{s}}(A \boxdot \boldsymbol{b}) = \boldsymbol{u} \cdot \mathsf{Err}(A) + \mu_A \cdot \varphi_{\boldsymbol{s}}(\boldsymbol{\epsilon_{dec}}) + \mu_A \cdot \mathsf{Err}(\boldsymbol{b}) + \mu_A \mu_{\boldsymbol{b}}.$$

Taking the expectation, we get that $\mathsf{msg}(A \boxdot \boldsymbol{b}) = 0 + 0 + 0 + \mu_A \mu_{\boldsymbol{b}}$, and so $\mathsf{Err}(A \boxdot \boldsymbol{b}) = \varphi_{\boldsymbol{s}}(A \boxdot \boldsymbol{b}) - \mu_A \mu_{\boldsymbol{b}}$. Then thanks to lemma 3.12, we have

$$\begin{aligned} \|\mathsf{Err}(A \boxdot \boldsymbol{b})\|_\infty &\leq \|\boldsymbol{u} \cdot \mathsf{Err}(A)\|_\infty + \|\mu_A \cdot \varphi(\boldsymbol{\epsilon_{dec}})\|_\infty + \|\mu_A \cdot \mathsf{Err}(\boldsymbol{b})\|_\infty \\ &\leq (k+1)\ell N \beta \eta_A + \|\mu_A\|_1 (1 + kN) \|\boldsymbol{\epsilon_{dec}}\|_\infty + \|\mu_A\|_1 \eta_{\boldsymbol{b}}. \end{aligned}$$

The result follows. □

We similarly obtain the more realistic average-case noise propagation, based on the independence heuristic, by bounding the Gaussian variance instead of the amplitude.

Corollary 3.15 (Average-case External Product). *Under the same conditions of theorem 3.14 and by assuming the heuristic 3.6, we have that* $\mathsf{Var}(\mathsf{Err}(A \boxdot \boldsymbol{b})) \leq (k+1)\ell N \beta^2 \mathsf{Var}(\mathsf{Err}(A)) + (1+kN) \|\mu_A\|_2^2 \epsilon^2 + \|\mu_A\|_2^2 \mathsf{Var}(\mathsf{Err}(\boldsymbol{b}))$.

Proof. Let $\vartheta_A = \mathsf{Var}(\mathsf{Err}(A)) = \mathsf{Var}(\varphi_{\boldsymbol{s}}(Z_A))$ and $\vartheta_{\boldsymbol{b}} = \mathsf{Var}(\mathsf{Err}(\boldsymbol{b})) = \mathsf{Var}(\varphi_{\boldsymbol{s}}(\boldsymbol{z_b}))$. By using the same notations as in the proof of theorem 3.14 we have that the error of $A \boxdot \boldsymbol{b}$ is $\mathsf{Err}(A \boxdot \boldsymbol{b}) = \boldsymbol{u} \cdot \mathsf{Err}(A) + \mu_A \cdot \varphi_{\boldsymbol{s}}(\boldsymbol{\epsilon_{dec}}) + \mu_A \cdot \mathsf{Err}(\boldsymbol{b})$ and thanks to assumption 3.6 and lemma 3.12, we have :

$$\begin{aligned} \mathsf{Var}(\mathsf{Err}(A \boxdot \boldsymbol{b})) &\leq \mathsf{Var}(\boldsymbol{u} \cdot \mathsf{Err}(A))) + \mathsf{Var}(\mu_A \cdot \varphi(\boldsymbol{\epsilon_{dec}})) + \mathsf{Var}(\mu_A \cdot \mathsf{Err}(\boldsymbol{b})) \\ &\leq (k+1)\ell N \beta^2 \vartheta_A + (1 + kN) \|\mu_A\|_2^2 \epsilon^2 + \|\mu_A\|_2^2 \vartheta_{\boldsymbol{b}}. \end{aligned}$$

□

The last corollary describes exactly the classical internal product between two TGSW samples, already presented in [16, 3, 13, 11] with adapted notations. As we mentionned before, it is much slower to evaluate, because it consists in $(k+1)\ell$ independent computations of the $\boxdot$ product, which we illustrate now.

Corollary 3.16 (Internal Product). *Let the product*

$$\boxtimes\colon \mathrm{TGSW} \times \mathrm{TGSW} \longrightarrow \mathrm{TGSW}$$
$$(A, B) \longmapsto A \boxtimes B = \begin{bmatrix} A \boxdot \boldsymbol{b_1} \\ \vdots \\ A \boxdot \boldsymbol{b_{(k+1)\ell}} \end{bmatrix} = \begin{bmatrix} Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{b_1}) \cdot A \\ \vdots \\ Dec_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{b_{(k+1)\ell}}) \cdot A \end{bmatrix},$$

with A and B two valid TGSW *samples of messages μ_A and μ_B respectively and $\boldsymbol{b_i}$ corresponding to the i-th line of B. Then $A \boxtimes B$ is a* TGSW *sample of message $\mu_A \cdot \mu_B$ and $\|\mathsf{Err}(A \boxdot B)\|_\infty \leq (k+1)\ell N\beta \|\mathsf{Err}(A)\|_\infty + \|\mu_A\|_1 (1+kN)\epsilon + \|\mu_A\|_1 \|\mathsf{Err}(B)\|_\infty$ (worst case). If $\|\mathsf{Err}(A \boxdot B)\|_\infty \leq 1/4$ we are guaranteed that $A \boxdot B$ is a valid* TGSW *sample.*
Furthermore, by assuming the heuristic 3.6, we have that $\mathsf{Var}(\mathsf{Err}(A \boxdot B)) \leq (k+1)\ell N\beta^2 \mathsf{Var}(\mathsf{Err}(A)) + (1+kN)(\mu_A\epsilon)^2 + \mu_A^2 \mathsf{Var}(\mathsf{Err}(\boldsymbol{b}))$ (average case).

Proof. Let A and B be two TGSW samples, and μ_A and μ_B their message. By definition, the i-th row of B encodes $\mu_B \cdot \boldsymbol{h_i}$, so the i-th row of $A \boxtimes B$ encodes $(\mu_A\mu_B) \cdot \boldsymbol{h_i}$. This proves that $A \boxtimes B$ encodes $\mu_A\mu_B$. Since the internal product $A \boxtimes B$ consists in $(k+1)\ell$ independent runs of the external products $A \boxdot \boldsymbol{b_i}$, the noise propagation formula directly follows from Thm. 3.14 and Cor. 3.15. □

The last corollaries describe exactly the internal products already presented in [16], [3], [13] and [11] with adapted notations. In the next section, we show that all internal products in the bootstrapping procedure can be replaced with the external one. Consequently, we expect a speed-up of a factor at least $(k+1)\ell$.

4 Application: Single gate Bootstrapping in less than 0.1 seconds

In this section, we show how to use Theorem 3.14 to speed-up the bootstrapping presented in [11]. With additional optimizations, we drastically reduce the bootstrapping key size, and also reduce a bit the noise overhead. To bootstrap a LWE sample $(a, b) \in \mathbb{T}^{n+1}$, which is rescaled as $(\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}}) \mod 2N$, using relevant encryptions of its secret key $\boldsymbol{s} \in \mathbb{B}^n$, the overall idea is the following. We start from a fixed polynomial testv $\in \mathbb{T}_N[X]$, which is our phase detector: its i-th coefficient is set to the value that the bootstrapping should return if $\varphi_{\boldsymbol{s}}(a, b) = i/2N$. testv is first encoded in a trivial LWE sample. Then, we iteratively rotate its coefficients, using external multiplications with TGSW encryptions of the hidden monomials $X^{-s_i\bar{a}_i}$. By doing so, the original testv gets rotated by the (hidden) phase of $(\boldsymbol{a}, b)$, and in the end, we simply extract the constant term as a LWE sample.

4.1 TLWE to LWE extraction

Like in previous work, extracting a LWE sample from a TLWE sample simply means rewriting polynomials into their list of coefficients, and discarding the

$N-1$ last coefficients of b. This yields a LWE encryption of the constant term of the initial polynomial message.

Definition 4.1 (TLWE Extraction). *Let $(\boldsymbol{a''}, b'')$ be a $\mathrm{TLWE}_{\boldsymbol{s''}}(\mu)$ sample with key $\boldsymbol{s''} \in \mathfrak{R}^k$, We call $\mathsf{KeyExtract}(\boldsymbol{s''})$ the integer vector $\boldsymbol{s'} = (\mathsf{coefs}(s''_1(X), \ldots, \mathsf{coefs}(s''_k(X)) \in \mathbb{Z}^{kN}$ and $\mathsf{SampleExtract}(\boldsymbol{a''}, b'')$ the LWE sample $(\boldsymbol{a'}, b') \in \mathbb{T}^{kN+1}$ where $\boldsymbol{a'} = (\mathsf{coefs}(a''_1(1/X), \ldots, \mathsf{coefs}(a''_k(1/X))$ and $b' = b''_0$ the constant term of b''. Then $\varphi_{\boldsymbol{s'}}(a', b')$ (resp. $\mathsf{msg}(a', b')$) is equal to the constant term of $\varphi_{\boldsymbol{s''}}(a'', b'')$ (resp. to the constant term of $\mu = \mathsf{msg}(a'', b'')$). And $\|\mathsf{Err}(a', b')\|_\infty \leq \|\mathsf{Err}(a'', b'')\|_\infty$ and $\mathsf{Var}(\mathsf{Err}(a', b')) \leq \mathsf{Var}(\mathsf{Err}(a'', b''))$.*

4.2 LWE to LWE Key-Switching Procedure

Given a $\mathsf{LWE}_{\boldsymbol{s'}}$ sample of a message $\mu \in \mathbb{T}$, the key switching procedure initially proposed in [7, 5] outputs a $\mathsf{LWE}_{\boldsymbol{s}}$ sample of the same μ without increasing the noise too much. Contrary to previous exact keyswitch procedures, here we tolerate approximations.

Definition 4.2. *Let $\boldsymbol{s'} \in \{0,1\}^{n'}$, $\boldsymbol{s} \in \{0,1\}^n$, a noise parameter $\gamma \in \mathbb{R}$ and a precision parameter $t \in \mathbb{N}$, we call key switching secret $\mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s}, \gamma, t}$ a sequence of fresh LWE samples $\mathsf{KS}_{i,j} \in \mathsf{LWE}_{\boldsymbol{s}, \gamma}(s'_i \cdot 2^{-j})$ for $i \in [1, n']$ and $j \in [1, t]$.*

Lemma 4.3 (Key switching). *Given $(\boldsymbol{a'}, b') \in \mathsf{LWE}_{\boldsymbol{s'}}(\mu)$ where $\boldsymbol{s'} \in \{0,1\}^{n'}$ with noise $\eta' = \|\mathsf{Err}(\boldsymbol{a'}, b')\|_\infty$ and a keyswitching key $\mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s}, \gamma, t}$, where $\boldsymbol{s} \in \{0,1\}^n$, the key switching procedure outputs a LWE sample $(\boldsymbol{a}, b) \in \mathsf{LWE}_{\boldsymbol{s}_n}(\mu)$ where $\|\mathsf{Err}(\boldsymbol{a}, b)\|_\infty \leq \eta' + n' t \gamma + n' 2^{-(t+1)}$.*

Proof. We have

$$\begin{aligned}
\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) &= \varphi_{\boldsymbol{s}}(\boldsymbol{0}, b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \varphi_{\boldsymbol{s}}(\mathsf{KS}_{i,j}) \\
&= b' - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \left(2^{-j} s'_i + \mathsf{Err}(\mathsf{KS}_{i,j})\right) \\
&= b' - \sum_{i=1}^{n'} \bar{a}'_i s'_i - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) \\
&= b' - \sum_{i=1}^{n'} a'_i s'_i - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{n'} (a'_i - \bar{a}'_i) s'_i \\
&= \varphi_{\boldsymbol{s'}}(\boldsymbol{a'}, b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{n'} (a'_i - \bar{a}'_i) s'_i.
\end{aligned}$$

The expectation of the left side of the equality is equal to $\mathsf{msg}(\boldsymbol{a}, b)$. For the right side, each $a_{i,j}$ is uniformly distributed in $\{0,1\}$ and $(a'_i - \bar{a}'_i)$ is a 0-centered variable so the expectation of the sum is 0. Thus, $\mathsf{msg}(\boldsymbol{a}, b) = \mathsf{msg}(\boldsymbol{a'}, b')$. We obtain $\|\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) - \mathsf{msg}(\boldsymbol{a}, b)\|_\infty \leq \eta' + n' \cdot t \cdot \gamma + n' 2^{-(t+1)}$. □

Algorithm 2 KeySwitch procedure

Input: A LWE sample $(\boldsymbol{a'} = (a'_1, \ldots, a'_{n'}), b') \in \mathsf{LWE}_{\boldsymbol{s'}}(\mu)$, a switching key $\mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s}}$ where $\boldsymbol{s'} \in \{0,1\}^{n'}$, $\boldsymbol{s} \in \{0,1\}^n$ and $t \in \mathbb{N}$ a precision parameter

Output: A LWE sample $\mathsf{LWE}_{\boldsymbol{s}}(\mu)$

1: Let $\bar{a}'_i$ be the closest multiple of $\frac{1}{2^t}$ to a'_i, thus $|\bar{a}'_i - a'_i| < 2^{-(t+1)}$

2: Binary decompose each $\bar{a}_i = \sum_{j=1}^{\ell} a_{i,j} \cdot 2^{-j}$ where $a_{i,j} \in \{0,1\}$

3: Return $(\boldsymbol{0}, b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \cdot \mathsf{KS}_{i,j}$

Corollary 4.4. *Let t be an integer parameter. Under Assumption 3.6 Given $(\boldsymbol{a'}, b') \in \mathsf{LWE}_{\boldsymbol{s'}}(\mu)$ with noise variance $\eta' = \mathsf{Var}(\mathsf{Err}(\boldsymbol{a'}, b'))$ and a key switching key $\mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s}, \gamma, \ell}$, the key switching procedure outputs an LWE sample $(\boldsymbol{a}', b') \in \mathsf{LWE}_{\boldsymbol{s}}(\mu)$ where $\mathsf{Var}(\mathsf{Err}(\boldsymbol{a}, b)) \leq \eta' + n' \cdot t \cdot \gamma^2 + n' 2^{-2(t+1)}$.*

4.3 Bootstrapping Procedure

Given a LWE sample $\mathsf{LWE}_{\boldsymbol{s}}(\mu) = (\boldsymbol{a}, b)$, the bootstrapping procedure constructs an encryption of μ under the same key $\boldsymbol{s}$ but with a fixed amount of noise. As in [11], we will use TLWE as an intermediate encryption scheme to perform a homomorphic evaluation of the phase but here we will use its external product from theorem 3.14 with a TGSW encryption of the key $\boldsymbol{s}$.

Definition 4.5. *Let $\boldsymbol{s} \in \mathbb{B}^n$, $\boldsymbol{s''} \in \mathbb{B}_N[X]^k$ and α be a noise parameter. We define the bootstrapping key $BK_{\boldsymbol{s} \to \boldsymbol{s''}, \alpha}$ as the sequence of n TGSW samples where $BK_i \in \mathrm{TGSW}_{\boldsymbol{s''}, \alpha}(s_i)$.*

Algorithm 3 Bootstrapping procedure

Input: A LWE sample $(\boldsymbol{a}, b) \in \mathsf{LWE}_{\boldsymbol{s}, \eta}(\mu)$, a bootstrapping key $\mathrm{BK}_{\boldsymbol{s} \to \boldsymbol{s''}, \alpha}$, a keyswitch key $\mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s}, \gamma}$ where $\boldsymbol{s'} = \mathsf{KeyExtract}(\boldsymbol{s''})$, two fixed messages $\mu_0, \mu_1 \in \mathbb{T}$

Output: A LWE sample $\mathsf{LWE}_{\boldsymbol{s}, \nu}\left(\mu_0 \text{ if } \varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) \in \left]-\frac{1}{4}, \frac{1}{4}\right[; \mu_1 \text{ else}\right)$

1: Let $\bar{\mu} = \frac{\mu_1 + \mu_0}{2}$ and $\bar{\mu}' = \mu_0 - \bar{\mu}$

2: Let $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a}_i = \lfloor 2Na_i \rceil$ **for each** $i \in [1, n]$

3: Let testv $:= (1 + X + \ldots + X^{N-1}) \times X^{-\frac{2N}{4}} \cdot \bar{\mu}' \in \mathbb{T}_N[X]$

4: $\mathsf{ACC} \leftarrow \left(X^{\bar{b}} \cdot (\boldsymbol{0}, \text{testv})\right) \in \mathbb{T}_N[X]^{k+1}$

5: **for** $i = 1$ **to** n

6: $\quad \mathsf{ACC} \leftarrow \left[\boldsymbol{h} + (X^{-\bar{a}_i} - 1) \cdot \mathrm{BK}_i\right] \boxdot \mathsf{ACC}$

7: Let $\boldsymbol{u} := (\boldsymbol{0}, \bar{\mu}) + \mathsf{SampleExtract}(\mathsf{ACC})$

8: Return $\mathsf{KeySwitch}_{\mathsf{KS}}(\boldsymbol{u})$

We first provide a comparison between the bootstrapping of Algorithm 3 and [11, Algorithm 1,2] proposal.

– Like [11], we rescale the computation of the phase of the input LWE sample so that it is modulo $2N$ (line 2) and we map all the corresponding operations in the multiplicative cyclic group $\{1, X, \ldots, X^{2N-1}\}$. Since our LWE samples are described over the real torus, the rescaling is done explicitly in line 2. This rescaling may induce a cumulated rounding error of amplitude at most $\delta \approx \sqrt{n}/4N$ in the average case and $\delta \leq (n+1)/4N$ in the worst case. In the best case, this amplitude can decrease to zero ($\delta = 0$) if in the actual representation of LWE samples, all the coefficients are restricted to multiple of $\frac{1}{2N}$, which would be the analogue of [11]'s setting.
– As in [11], messages are encoded as roots of unity in $\mathcal{R}$. Our accumulator is a TLWE sample instead of a TGSW sample in [11]. Also accumulator operations use the external product from Theorem 3.14 instead of the slower classical internal product. The test vector $(1+X+\ldots+X^{N-1})$ is embedded in the accumulator from the very start, when the accumulator is still noiseless while in [11], it is added at the very end. This removes a factor $\sqrt{N}$ to the final noise overhead.
– All the TGSW ciphertexts of $X^{-\bar{a}_i s_i}$ required to update the accumulator internal value are computed dynamically as a very small polynomial combination of BK_i in the for loop (line 5). This completely removes the need to decompose each $\bar{a}_i$ on an additional base B_r, and to precompute all possibilities in the bootstrapping key. In other words, this makes our bootstrapping key 46 times smaller than in [11], for the exact same noise overhead. Besides, due to this squashing technique, two accumulator operations were performed per iteration instead of one in our case. This gives us an additional 2X speed up.

Theorem 4.6 (Bootstrapping Theorem). *Let $\boldsymbol{h} \in \mathcal{M}_{\ell(k+1),k+1}(\mathbb{T}_N[X])$ be the gadget defined in Equation 1 and let $Dec_{\boldsymbol{h},\epsilon,\beta}$ be the associated vector gadget decomposition function.*

Let $\boldsymbol{s} \in \mathbb{B}^n$, $\boldsymbol{s''} \in \mathbb{B}_N[X]^k$ and α, γ be noise amplitudes. Let $BK = BK_{\boldsymbol{s} \to \boldsymbol{s''},\alpha}$ be a bootstrapping key, let $\boldsymbol{s'} = \mathsf{KeyExtract}(\boldsymbol{s''}) \in \mathbb{B}^{kN}$ and $\mathsf{KS} = \mathsf{KS}_{\boldsymbol{s'} \to \boldsymbol{s},\gamma,t}$ be a keyswitching secret.

Given $(\boldsymbol{a}, b) \in \mathsf{LWE}_{\boldsymbol{s}}(\mu)$ for $\mu \in \mathbb{T}$, two fixed messages μ_0, μ_1, Algorithm 3 outputs a sample in $\mathsf{LWE}_{\boldsymbol{s}}(\mu')$ s.t. $\mu' = \mu_0$ if $|\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b)| < -1/4 - \delta$ and $\mu' = \mu_1$ if $|\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b)| > 1/4 + \delta$ where δ is the cumulated rounding error equal to $\frac{n+1}{4N}$ in the worst case and $\delta = 0$ if the all coefficients of $(\boldsymbol{a}, b)$ are multiple of $\frac{1}{2N}$. Let $\boldsymbol{v}$ be the output of Algorithm 3. Then $\|\mathsf{Err}(\boldsymbol{v})\|_\infty \leq 2n(k+1)\ell\beta N\alpha + kNt\gamma + n(1+kN)\epsilon + kN2^{-(t+1)}$.

Proof. Line 1: the division by two over torus gives two possible values for $(\bar{\mu}, \bar{\mu}')$. In both cases, $\bar{\mu} + \bar{\mu}' = \mu_0$ and $\bar{\mu} - \bar{\mu}' = \mu_1$.

Line 2: let $\bar{\varphi} \stackrel{def}{=} \bar{b} - \sum_{i=1}^n \bar{a}_i s_i \mod 2N$. We have

$$\left|\varphi - \frac{\bar{\varphi}}{2N}\right| = b - \frac{\lfloor 2Nb \rceil}{2N} + \sum_{i=1}^n \left(a_i - \frac{\lfloor 2Na_i \rceil}{2N}\right) s_i \leq \frac{1}{4N} + \sum_{i=1}^n \frac{1}{4N} \leq \frac{n+1}{4N}. \quad (2)$$

And if the coefficients $(\boldsymbol{a}, b) \in \frac{1}{2N}\mathbb{Z}/\mathbb{Z}$, then $\varphi = \frac{\bar{\varphi}}{2N}$. In all cases, $|\varphi - \frac{\bar{\varphi}}{2N}| < \delta$.

At line 3, the test vector testv $:= (1+X+\ldots+X^{N-1}) \cdot X^{-\frac{2N}{4}} \bullet \bar{\mu}'$ is defined such that for all $p \in [0, 2N]$, the constant term of $X^p \bullet \text{testv}$ is either $\bar{\mu}'$ if $p \in]-\frac{N}{2}, \frac{N}{2}[$ and $-\bar{\mu}'$ else.

In the loop for (from line 5 to 6), we will prove the following invariant: At the beginning of iteration $i+1 \in [1, n+1]$ (i.e. at the end of iteration i), $\mathsf{msg}(ACC_i) = X^{b-\sum_{j=1}^{i} \bar{a}_j s_j} \bullet \text{testv}$ and $\|\mathsf{Err}(ACC_i)\|_\infty \leq \sum_{j=1}^{i} \Big(2(k+1)\ell N\beta \|\mathsf{Err}(\mathrm{BK}_j)\|_\infty + (1+kN)\epsilon\Big)$.

At the beginning of iteration $i = 1$, the accumulator contains a trivial ciphertext $\mathsf{msg}(ACC_1) = \Big(X^{\bar{b}} \bullet \text{testv}\Big)$, so $\|\mathsf{Err}(ACC_1)\|_\infty = 0$.

During iteration i, $A_i = \boldsymbol{h} + (X^{-\bar{a}_i} - 1) \bullet \mathrm{BK}_i$ is a TGSW sample of message $X^{-\bar{a}_i s_i}$ (this can be seen by replacing s_i with its two possible values 0 and 1) and of noise $\|\mathsf{Err}(A_i)\|_\infty \leq 2\|\mathsf{Err}(\mathrm{BK}_i)\|_\infty$. This inequality holds from lemma 3.11. Then, we have:

$$\begin{aligned}
\mathsf{msg}(ACC_i) &= \mathsf{msg}\Big(A_i \boxdot ACC_{i-1}\Big) \\
&= \mathsf{msg}\Big(A_i\Big) \bullet \mathsf{msg}(ACC_{i-1}) && \text{(from Theorem 3.14)} \\
&= X^{-\bar{a}_i s_i} \cdot (X^{b-\sum_{j=1}^{i-1} \bar{a}_j s_j} \bullet \text{testv})
\end{aligned}$$

and from the norm inequality of Theorem 3.14,

$$\begin{aligned}
\|\mathsf{Err}(ACC_i)\|_\infty &\leq (k+1)\ell N\beta \|\mathsf{Err}(A_i)\|_\infty + \|\mathsf{msg}(A_i)\|_1 (1+kN)\epsilon + \\
&\quad + \|\mathsf{msg}(A_i)\|_1 \|\mathsf{Err}(ACC_{i-1})\|_\infty \\
&\leq (k+1)\ell N\beta 2 \|\mathsf{Err}(\mathrm{BK}_i)\|_\infty + (1+kN)\epsilon + \|\mathsf{Err}(ACC_{i-1})\|_\infty .
\end{aligned}$$

This proves the invariant by induction on i.

After $\mathsf{SampleExtract}$ (line 7), the message of u is equal to the constant term of the message of ACC_n, i.e. $X^{\bar{\varphi}} \bullet \text{testv}$ where $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} \bar{a}_i s_i$. If $\bar{\varphi} \in [\![-N/2, N/2[\![$, the constant term is equal to $\bar{\mu}'$ and $-\bar{\mu}'$ otherwise.

In other words, $|\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b)| < 1/4 - \delta$, then $\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) < 1/4 - \delta$ and $\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) \geq -1/4 + \delta$ and thus using Equation (2), we obtain that $\bar{\varphi} \in]-\frac{N}{2}, \frac{N}{2}[\![$ and thus, the message of u is equal to $\bar{\mu}'$. And if $|\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b)| > 1/4 + \delta$ then $\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) > 1/4 + \delta$ or $\varphi_{\boldsymbol{s}}(\boldsymbol{a}, b) < -1/4 - \delta$ and using Equation (2), we obtain the message of u is equal to $-\bar{\mu}'$.

Since $\mathsf{SampleExtract}$ does not add extra noise, $\|\mathsf{Err}(\boldsymbol{u})\|_\infty \leq \|\mathsf{Err}(ACC_n)\|$. Since the KeySwitch procedure preserves the message, the message of $v = \mathsf{KeySwitch}_{\mathsf{KS}}(\boldsymbol{u})$ is equal to the message of u. And $\|\mathsf{Err}(\boldsymbol{v})\|_\infty \leq \|\mathsf{Err}(\boldsymbol{u})\|_\infty + kNt\gamma + kN2^{-(t+1)}$. □

Corollary 4.7. *Let $\vartheta_{BK} = \mathsf{Var}(\mathsf{Err}(BK_i)) = 2/\pi \cdot \alpha^2$ and $V_{\mathsf{KS}} = \mathsf{Var}(\mathsf{Err}(\mathsf{KS}_i)) = 2/\pi \cdot \gamma^2$. Under the same conditions of Theorem 4.6, and assuming Assumption 3.6, then the Variance of the output v of Algorithm 3 satisfies $\mathsf{Var}(\mathsf{Err}(v)) \leq 2Nn(k+1)\ell\beta^2\vartheta_{BK} + kNtV_{\mathsf{KS}} + n(1+kN)\epsilon^2 + kN2^{-2(t+1)}$.*

Proof. The proof is the same as for the proof of the bound on $\|\mathsf{Err}(\boldsymbol{v})\|_\infty$ replacing all $\|\|_\infty$ inequalities by $\mathsf{Var}()$ inequalities. □

4.4 Application to circuits

In [11], the homomorphic evaluation of a NAND gate between LWE samples is achieved with 2 additions (one with a noiseless trivial sample) and a bootstrapping. Let $\mathrm{BK} = \mathrm{BK}_{\boldsymbol{s}\to\boldsymbol{s}'',\alpha}$ be a bootstrapping key and $\mathsf{KS} = \mathsf{KS}_{\boldsymbol{s}'\to\boldsymbol{s},\gamma,t}$ be a keyswitching secret defined as in thm. 4.6 such that $2n(k+1)\ell\beta N\alpha + kNt\gamma + n(1+kN)\epsilon + kN2^{-(t+1)} < \frac{1}{16}$, We denote as $\mathsf{Bootstrap}\,(\boldsymbol{c})$ the output of the bootstrapping procedure described in Algorithm 3 applied to $\boldsymbol{c}$ with $\mu_0 = 0$ and $\mu_1 = \frac{1}{4}$. Let consider two LWE samples c_1 and c_2, with message space $\{0, 1/4\}$ and $\|\mathsf{Err}(\boldsymbol{c_1})\|_\infty, \|\mathsf{Err}(\boldsymbol{c_2})\|_\infty \leq \frac{1}{16}$. The result is obtained by computing $\tilde{\boldsymbol{c}} = (\boldsymbol{0}, \frac{5}{8})\text{-}\boldsymbol{c_1}\text{-}\boldsymbol{c_2}$, plus a bootstrapping. Indeed the possible values for the messages of $\tilde{\boldsymbol{c}}$ are $\frac{5}{8}, \frac{3}{8}$ if either c_1 or c_2 encode 0, and $\frac{1}{8}$ if both encode $\frac{1}{4}$. Since the noise amplitude $\|\mathsf{Err}(\tilde{\boldsymbol{c}})\|_\infty$ is $< \frac{1}{8}$, then $|\varphi_{\boldsymbol{s}}(\tilde{\boldsymbol{c}})| > \frac{1}{4}$ iff. $\mathrm{NAND}(\mathsf{msg}(\boldsymbol{c_1}), \mathsf{msg}(\boldsymbol{c_2})) = 1$. This explains why it suffices to bootstrap $\tilde{\boldsymbol{c}}$ with parameters $(\mu_1, \mu_0) = (\frac{1}{4}, 0)$ to get the answer. By using a similar approach, it is possible to directly evaluate with a single bootstrapping all the basic gates:

- $\mathrm{HomNOT}(\boldsymbol{c}) = (\boldsymbol{0}, \frac{1}{4})\text{-}\boldsymbol{c}$ (no bootstrapping is needed);
- $\mathrm{HomAND}(\boldsymbol{c_1}, \boldsymbol{c_2}) = \mathsf{Bootstrap}\left((\boldsymbol{0}, -\frac{1}{8}) + \boldsymbol{c_1} + \boldsymbol{c_2}\right)$;
- $\mathrm{HomNAND}(\boldsymbol{c_1}, \boldsymbol{c_2}) = \mathsf{Bootstrap}\left((\boldsymbol{0}, \frac{5}{8})\text{-}\boldsymbol{c_1}\text{-}\boldsymbol{c_2}\right)$;
- $\mathrm{HomOR}(\boldsymbol{c_1}, \boldsymbol{c_2}) = \mathsf{Bootstrap}\left((\boldsymbol{0}, \frac{1}{8}) + \boldsymbol{c_1} + \boldsymbol{c_2}\right)$;
- $\mathrm{HomXOR}(\boldsymbol{c_1}, \boldsymbol{c_2}) = \mathsf{Bootstrap}\,(2 \cdot (\boldsymbol{c_1}\text{-}\boldsymbol{c_2}))$.

The $\mathrm{HomXOR}(\boldsymbol{c_1}, \boldsymbol{c_2})$ gate can be achieved also by performing $\mathsf{Bootstrap}\,(2 \cdot (\boldsymbol{c_1} + \boldsymbol{c_2}))$.

4.5 Parameters Implementation and Timings

In this section, we review our implementation parameters and provide a comparison with previous works.

Samples. From a theoretical point of view, our scale invariant scheme is defined over the real torus $\mathbb{T}$, where all the operations are modulo 1. In practice, since we can work with approximations, we chose to rescale the elements over $\mathbb{T}$ by a factor 2^{32}, and to map them to 32-bit integers. Thus, we take advantage of the native and automatic mod 2^{32} operations, including for the external multiplication with integers. Except for some FFT operations, this seems more stable and efficient than working with floating point numbers and reducing modulo 1 regularly. Polynomials mod $X^N + 1$ are either represented as the classical list of the N coefficients, either using the Lagrange half-complex representation, which consists in the complex $(2 \cdot 64\text{bits})$ evaluations of the polynomial over the roots of unity $\exp(i(2j+1)\pi/N)$ for $j \in [\![0, \frac{N}{2}[\![$. Indeed, the $\frac{N}{2}$ other evaluations are the conjugates of the first ones, and do not need to be stored. The conversion

between both representations is done via Fast Fourier Transform (FFT) (using the library *FFTW*[12], also used by [11]). Note that the direct FFT transform is $\sqrt{2N}$ lipschitzian, so the lagrange half-complex representation tolerates approximations, and 53bits of precision is indeed more than enough, provided that the real representative remains small. However, the modulo 1 that can reduce the coefficients of Torus polynomials cannot be applied from the Lagrange representation: we need to perform regular transformations to and from the classical representation. Luckily, it does not represent an overhead, since these conversions are needed anyway, at each iteration of the bootstrapping in order to decompose the accumulator in base $\boldsymbol{h}$.

Parameters. We take the same or even stronger security parameters as [11], but we adapt them to our notations. We used $n = 500$, $N = 1024$, $k = 1$.

- LWE samples: $32 \cdot (n + 1)$ bits ≈ 2 KBytes.
  The mask of all LWE samples (initial and KeySwitch) are clamped to multiples of $\frac{1}{2048}$. Therefore, the phase computation in the bootstrapping is exact ($\delta = 0$).
- TLWE samples: $(k + 1) \cdot N \cdot 32$ bits ≈ 8 KBytes.
- TGSW samples: $(k + 1) \cdot \ell$ TLWE samples ≈ 48 KBytes.
  To define $\boldsymbol{h}$ and $\mathrm{Dec}_{\boldsymbol{h},\beta,\epsilon}$, we used $\ell = 3$, $B_g = 1024$, so $\beta = 512$ and $\epsilon = 2^{-31}$.
- Bootstrapping Key: n TGSW samples ≈ 23.4 MBytes.
  We used $\alpha = 9.0 \cdot 10^{-9}$. Since we have a lower noise overhead, our parameter is higher than the parameter $\approx 3.25 \cdot 10^{-10}$ of [11], (i.e. ours is more secure), but in counterpart, our TLWE key is binary. See Section 6 for more details on the security analysis.
- Key Switching Key: $k \cdot N \cdot t$ LWE samples ≈ 29.2 MBytes.
  we used $\gamma = 3.05 \cdot 10^{-5}$, $t = 15$ (The decomposition in the key switching has an precision 2^{-16}).

- Correctness: The final error variance after bootstrapping is $9.24.10^{-6}$, by Corollary 4.7. It corresponds to a standard deviation of $\sigma = 0.00961$. In [11], the final standard deviation is larger 0.01076. In other words, the noise amplitude after our bootstrapping is $< \frac{1}{16}$ with very high probability $\mathsf{erf}(1/16\sqrt{2}\sigma) \geq 1 - 2^{-33.56}$ (this is comparable to probability $\geq 1 - 2^{-32}$ in [11]).

Note that the size of the key switching key can be reduced by a factor $n + 1 = 501$ if all the masks are the output of a pseudo random function; we may for instance just give the seed. The same technique can be applied to the bootstrapping key, on which the size is only reduced by a factor $k + 1 = 2$.

Implementation tools and Source Code. The source code of our implementation is available on github `https://github.com/tfhe/tfhe`. We implemented the FHE scheme in C/C++, and run the bootstrapping algorithm on a 64-bit single core (i7-4930MX) at 3.00GHz. This seems to correspond to the machine used in [11].

We implemented a version with classical representation for polynomials, and a version in Lagrange half-complex representation. The following table compares the number of multiplications or FFT that are required to complete one external product and the full bootstrapping.

	#(Classical products)	#(FFT + Lagrange repr.)
External product	12	8
Bootstrapping	6000	4006
Bootstrapping in [11]	(72000)	48000

In practice, we obtained a running time of 52ms per bootstrapping using the Lagrange half-complex representation. It is coherent with the 12x speed-up predicted by the table. Profiling the execution shows that the FFTs and complex multiplications are still taking more than 90% of the total time. Other operations like keyswitch have a negligible running time compared to the main loop of the bootstrapping.

5 Leveled Homomorphic encryption

In the previous section, we showed how to accelerate the bootstrapping computation in FHE. In this section, we focus on the improvement of Leveled Homomorphic encryption schemes. We present an efficient way to evaluate any deterministic automata homomorphically.

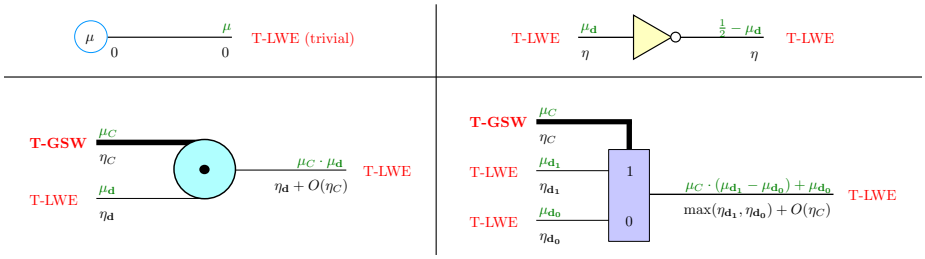
5.1 Boolean circuits interpretation

In order to express our external product in a circuit, we consider two kinds of wires: *control wires* which encode either a small integer or a small integer polynomial. They will be represented by a TGSW sample; and *data wires* which encode either a sample in $\mathbb{T}$ or in $\mathbb{T}_N[X]$. They will be represented by a TLWE sample. The gates we present contain three kinds of slots: *control input*, *data input* and *data output*. In this following section, the rule to build valid circuits is that all control wires are freshly generated by the user, and the data input ports of our gates can be either freshly generated or connected to a data output or to another gate.

We now give an interpretation of our leveled scheme, to simulate boolean circuits only. In this case, the message space of the input TLWE samples will be restricted to $\{0, \frac{1}{2}\}$, and the message space of control gates to $\{0, 1\}$.

- The constant source $\texttt{Cst}(\mu)$ for $\mu \in \{0, \frac{1}{2}\}$ is defined with a single data output equal to $(\mathbf{0}, \mu)$.
- The negation gate $\texttt{Not}(\boldsymbol{d})$ takes a single data input $\boldsymbol{d}$ and outputs $(\mathbf{0}, \frac{1}{2}) - \boldsymbol{d}$.
- The controlled And gate $\texttt{CAnd}(C, \boldsymbol{d})$ takes one control input C and one data input $\boldsymbol{d}$, and outputs $C \boxdot \boldsymbol{d}$.
- The controlled Mux gate $\texttt{CMux}(C, \boldsymbol{d_1}, \boldsymbol{d_0})$ takes one control input C and two data inputs $\boldsymbol{d_1}, \boldsymbol{d_0}$ and returns $C \boxdot (\boldsymbol{d_1} - \boldsymbol{d_0}) + \boldsymbol{d_0}$.

Unlike classical circuits, these gates have to be composed with each other depending on the type of inputs/outputs. In our applications, the TGSW encryptions are always fresh ciphertexts.



Theorem 5.1 (Correctness). *Let* $\mu \in \{0, \frac{1}{2}\}$, $\boldsymbol{d}, \boldsymbol{d_1}, \boldsymbol{d_0} \in \mathrm{TLWE}_{\boldsymbol{s}}(\{0, \frac{1}{2}\})$ *and* $C \in \mathrm{TGSW}_{\boldsymbol{s}}(\{0,1\})$.

- $\mathit{msg}(\mathtt{Cst}(\mu)) = \mu$
- $\mathit{msg}(\mathtt{Not}(\boldsymbol{d})) = \frac{1}{2} - \mu = \mathit{not}\,\mu$
- $\mathit{msg}(\mathtt{CAnd}(C, \boldsymbol{d})) = \mathit{msg}(C) \cdot \mathit{msg}(\boldsymbol{d})$
- $\mathit{msg}(\mathtt{CMux}(C, \boldsymbol{d_1}, \boldsymbol{d_0})) = \mathit{msg}(C)\,?\mathit{msg}(\boldsymbol{d_1}){:}\mathit{msg}(\boldsymbol{d_0})$

Theorem 5.2 (Worst-case noise). *In the conditions of thm 5.1, we have*

- $\|\mathit{Err}(\mathtt{Cst}(\mu))\|_\infty = 0$
- $\|\mathit{Err}(\mathtt{Not}(\boldsymbol{d}))\|_\infty = \|\mathit{Err}(\boldsymbol{d})\|_\infty$
- $\|\mathit{Err}(\mathtt{CAnd}(C, \boldsymbol{d}))\|_\infty \leq \|\mathit{Err}(\boldsymbol{d})\|_\infty + \eta(C)$
- $\|\mathit{Err}(\mathtt{CMux}(C, \boldsymbol{d_1}, \boldsymbol{d_0}))\|_\infty \leq \max(\|\mathit{Err}(\boldsymbol{d_0})\|_\infty, \|\mathit{Err}(\boldsymbol{d_1})\|_\infty) + \eta(C)$, *where* $\eta(C) = (k+1)\ell N\beta \|\mathit{Err}(C)\|_\infty + (kN+1)\epsilon$.

Proof. The noise is indeed null for constant gates, and negated for the Not gate, which preserves the norm. The noise bound for the CAnd gate is exactly the one from Theorem 3.14, however, we need to explain why there is a max in the CMux formula instead of the sum we would obtain by blindly applying thm 3.14. Let $\boldsymbol{d} = \boldsymbol{d_1} - \boldsymbol{d_0}$, recall that in the proof of Theorem 3.14, the expression of $C \boxdot \boldsymbol{d}$ is $\mathrm{Dec}_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{d}) \cdot \boldsymbol{z_C} + \mu_C \boldsymbol{\epsilon}_{\mathrm{dec}} + \mu_C \boldsymbol{z_d} + (\boldsymbol{0}, \mu_C \cdot \mu_d)$, where $C = \boldsymbol{z_C} + \mu_C \cdot \boldsymbol{h}$ and $\boldsymbol{d} = \boldsymbol{z_d} + \mu_d$, $\boldsymbol{z_C}$ and $\boldsymbol{z_d}$ are respectively TGSW and TLWE samples of 0, and $\|\boldsymbol{\epsilon}_{\mathrm{dec}}\|_\infty \leq \epsilon$. Thus, $\mathtt{CMux}(C, \boldsymbol{d_1}, \boldsymbol{d_0})$ is the sum of four terms:

- $\mathrm{Dec}_{\boldsymbol{h},\beta,\epsilon}(\boldsymbol{d}) \cdot \boldsymbol{z_C}$ of norm $\leq (k+1)\ell N\beta\eta_C$;
- $\mu_C \boldsymbol{\epsilon}_{\mathrm{dec}}$ of norm $\leq (kN+1)\epsilon$;
- $z_{d_0} + \mu_C(z_{d_1} - z_{d_0})$, which is either z_{d_1} or z_{d_0}, depending on the value of μ_C;
- $\mu_{d_0} + \mu_C \cdot (\mu_{d_1} - \mu_{d_0})$, which is the output message $\mu_C ? \mu_{d_1} : \mu_{d_0}$, and is not part of the noise.

Thus, summing the three terms concludes the proof. □

Corollary 5.3 (Average noise of boolean gates). *In the conditions of thm 5.1, and in the conditions of Assumption 3.6, we have:*

- $\mathsf{Var}(\mathsf{Err}(\mathtt{Cst}(\mu))) = 0$;
- $\mathsf{Var}(\mathsf{Err}(\mathtt{Not}(\boldsymbol{d}))) = \mathsf{Var}(\mathsf{Err}(\boldsymbol{d}))$;
- $\mathsf{Var}(\mathsf{Err}(\mathtt{CAnd}(C, \boldsymbol{d}))) \leq \mathsf{Var}(\mathsf{Err}(\boldsymbol{d})) + \vartheta(C)$;
- $\mathsf{Var}(\mathsf{Err}(\mathtt{CMux}(C, \boldsymbol{d_1}, \boldsymbol{d_0}))) \leq \max(\mathsf{Var}(\mathsf{Err}(\boldsymbol{d_0})), \mathsf{Var}(\mathsf{Err}(\boldsymbol{d_1}))) + \vartheta(C)$, *where* $\vartheta(C) = (k+1)\ell N\beta^2 \mathsf{Var}(\mathsf{Err}(C)) + (kN+1)\epsilon^2$.

Proof. Same as thm 5.2, replacing all norm inequalities by Variance inequalities. □

We now obtain theorems which are analogue to [13], with a bit less noise on the mux gate, but with the additional restriction that CAnd and CMux have a control wire, which must necessarily be a fresh TGSW ciphertext.

The next step is to understand the meaning of this additional restriction in terms of expressiveness of the resulting homomorphic circuits.

It is clear that we cannot build a random boolean circuit, and just apply the noise recurrence formula from theorem 5.2 or cor. 5.3 to get the output noise level. Indeed, it is not allowed to connect a data wire to an control input.

In the following section, we will show that we can still obtain the two most important circuits of [13], namely the deterministic automata circuits, which can evaluate any permutation of regular languages with noise propagation sublinear in the word length and the lookup table, which evaluates arbitrary functions with sublinear noise propagation.

5.2 Deterministic automata

It is folklore that every deterministic program which reads its input bit-by-bit in a pre-determined order, uses less than B bits of memory, and produces a boolean answer, is equivalent to a deterministic automata of at most 2^B states (independently of the time complexity). This is in particular the case for every boolean function of p variables, that can be trivially executed with $p-1$ bits of internal memory by reading and storing its input bit-by-bit before returning the final answer. It is of particular interest for most arithmetic functions, like addition, multiplication, or CRT operations, whose naive evaluation only requires $O(\log(p))$ bits of internal memory.

Let $\mathcal{A} = (Q, i, T_0, T_1, F)$ be a deterministic automata (over the alphabet $\{0,1\}$, where Q is the set of states, $i \in Q$ denotes the initial state, T_0, T_1 are the two transitions (deterministic) functions from Q to Q and $F \subset Q$ is the set of final states. Such automata is used to evaluate (rational) boolean functions on words where the image of $(w_1, \ldots, w_p) \in \mathbb{B}^p$ is equal to 1 iff. $T_{w_p}(T_{w_{p-1}}(\ldots(T_{w_1}(i)))) \in F$, and 0 otherwise.

Following the construction of [13], we show that we are able to evaluate any deterministic automata homomorphically using only constant and CMux gates efficiently. The noise propagation remains linear in the length of the word w, but compared to [13, Thm. 7.11], we reduce the number of evaluated CMux gates by a factor $|w|$ for a specific class of acyclic automata that are linked to fixed-time algorithms.

Theorem 5.4 (Evaluating Deterministic Automata). *Let* $\mathcal{A} = (Q, i, T_0, T_1, F)$ *be a deterministic automata. Given* p *valid* TGSW *samples* $C_1, \ldots, C_p$ *encrypting the bits of a word* $\boldsymbol{w} \in \mathbb{B}^p$*, with noise amplitude* $\eta = \max_i \|\mathsf{Err}(C_i)\|_\infty$ *and* $\vartheta = \max_i \mathsf{Var}(\mathsf{Err}(C_i))$*, by evaluating at most* $\leq p\#Q$ *Cmux gates, one can produce a TLWE sample* $\boldsymbol{d}$ *which encrypts* $\frac{1}{2}$ *iff* $\mathcal{A}$ *accepts* $\boldsymbol{w}$*, and* 0 *otherwise such that* $\|\mathsf{Err}(\boldsymbol{d})\|_\infty \leq p \cdot ((k+1)\ell N\beta\eta + (kN+1)\epsilon)$*. Assuming Heuristic 3.6,* $\mathsf{Var}(\mathsf{Err}(\boldsymbol{d})) \leq p \cdot ((k+1)\ell N\beta^2\vartheta + (kN+1)\epsilon^2)$*. Furthermore, the number of evaluated* `CMux` *can be decreased to* $\leq \#Q$*. if* $\mathcal{A}$ *satisfies either one of the conditions:*
(i) for all $q \in Q$ *(except KO states), all the words that connect* i *to* q *have the same length;*
(ii) $\mathcal{A}$ *only accepts words of the same length.*

Proof. We initialize $\#Q$ noiseless ciphertexts $\boldsymbol{d}_{q,p}$ for $q \in Q$ with $\boldsymbol{d}_{q,p} = (\boldsymbol{0}, \frac{1}{2}) = \mathtt{Cst}(\frac{1}{2})$ if $q \in F$ and $\boldsymbol{d}_{q,p} = (\boldsymbol{0}, 0) = \mathtt{Cst}(0)$ otherwise. Then for each letter of $\boldsymbol{w}$, we map the transitions as follow for all $q \in Q$ an $j \in [\![0, p-1]\!]$: $\boldsymbol{d}_{q,j-1} = \mathtt{CMux}(\boldsymbol{C_j}, \boldsymbol{d_{T_1(q),j}}, \boldsymbol{d_{T_0(q),j}})$. And we finally output $\boldsymbol{d_{i,0}}$.

Indeed, with this construction, we have

$$\mathsf{msg}(\boldsymbol{d_{i,0}}) = \mathsf{msg}(\boldsymbol{d_{T_{w_1}(i),1}}) = \ldots = \mathsf{msg}(\boldsymbol{d_{T_{w_p}(T_{w_{p-1}}\ldots(T_{w_1}(i))\ldots),p}}),$$

which encrypts $\frac{1}{2}$ iff $T_{w_p}(T_{w_{p-1}} \ldots (T_{w_1}(i)) \ldots) \in F$, *i.e.* iff $w_1 \ldots w_p$ is accepted by $\mathcal{A}$. This proves correctness.

For the complexity, each $\boldsymbol{d}_{q,j}$ for all $q \in Q$ an $j \in [\![0, p-1]\!]$ is computed with a single `CMux`. By applying the noise propagation inequalities of Theorem 5.2 and Corollary 5.3, it follows by an immediate induction on j from p down to 0, that for all $j \in [\![0, p]\!]$, $\|\mathsf{Err}(\boldsymbol{d_{q,j}})\|_\infty \leq (p-j) \cdot ((k+1)\ell N\beta\eta + (kN+1)\epsilon)$ and $\mathsf{Var}(\mathsf{Err}(\boldsymbol{d_{q,j}})) \leq (p-j) \cdot ((k+1)\ell N\beta^2\vartheta + (kN+1)\epsilon^2)$.

Note that it is sufficient to evaluate only the $\boldsymbol{d_{q,j}}$ when q is accessible by at least one word of length j. Thus, if the $\mathcal{A}$ satisfies the additional condition (i), then for each $q \in Q$, we only need to evaluate $\boldsymbol{d_{q,j}}$ for at most one position j. Thus, we evaluate less than $\#Q$ CMux gates in total.

Finally, if $\mathcal{A}$ satisfies (ii), then we first compute the minimal deterministic automata of the same language (and removing the KO state if it is present), then with an immediate proof by contradiction, this minimal automata satisfies (i), and has less than $\#Q$ states. □

For sake of completeness, since every boolean function with p variables can be evaluated by an Automata (that accepting only words of length p), we obtain the evaluation of arbitrary boolean function as an immediate corollary, which is the leveled variant of [13, Cor 7.9].

Lemma 5.5 (Arbitrary Functions). *Let* f *be any boolean function with* p *inputs, and* $\boldsymbol{c_1}, \ldots, \boldsymbol{c_p}$ *be* p $\mathrm{TGSW}_{\boldsymbol{s}}(\{0,1\})$ *ciphertexts of* $x_1, \ldots, x_p \in \{0,1\}$*, with noise* $\|\mathsf{Err}(\boldsymbol{c_i})\|_\infty \leq \eta$ *for all* $i \in [1, p]$*. Then the CMux-based Reduced Binary Decision Diagram of* f *computes a* $\mathrm{TLWE}_{\boldsymbol{s}}$ *ciphertext* $\boldsymbol{d}$ *of* $\frac{1}{2}f(x_1, \ldots, x_p)$ *with noise* $\|\mathsf{Err}(\boldsymbol{d})\|_\infty \leq p((k+1)\ell N\beta\eta + (kN+1)\epsilon)$ *by evaluating* $\mathcal{N}(f) \leq 2^p$

CMux gates where $\mathcal{N}(f)$ is the number of distinct partial functions $(x_l, \ldots, x_p) \to f(x_1, \ldots, x_p)$ for all $l \in [\![1, p+1]\!], (x_1, \ldots, x_{l-1}) \in \mathbb{B}^{l-1}$.

Proof (sketch). A trivial automata which evaluates f consists in its full binary decision tree, with the initial state $i = q_{0,0}$ as the root, each state $q_{l,j}$ depth $l \in [\![0, p-1]\!]$ and $j \in [\![0, 2^l - 1]\!]$ is connected with $T_0(q_{l,j}) = q_{l+1,2j}$ and $T_1(q_{l,j}) = q_{l+1,2j+1}$, and at depth p, $q_{p,j} \in F$ iff $f(x_1, \ldots, x_p) = 1$ where $j = \sum_{l=1}^{p} x_l 2^{p-l}$. The minimal version of this automaton has at most $\mathcal{N}(f)$ states, the rest follows from Theorem 5.4. □

Application: compilation for leveled homomorphic circuits We now give an example of how we can map a problem to an automata in order to perform a leveled homomorphic evaluation. We will illustrate this concept on the computation of the p-th bit of an integer product $a \times b$ where a and b are given in base 2. We do not claim that the automata approach is the fastest way to solve the problem, arithmetic circuits based on bitDecomp/recomposition are likely to be faster. But the goal is to clarify the generality and simplicity of the process. All we need is a fixed-time algorithm that solves the problem using the least possible memory. Among all algorithms that compute a product, the most naive ones are in general the best: here, we choose the elementary-school multiplication algorithm that computes the product bit-by-bit, starting from the LSB, and counting the current carry with the fingers. The pseudocode of this algorithm is recalled in Algorithm 4. The pseudo-code is almost given as a deterministic automata, since each step reads a single input bit, and uses it to update its internal state (x, y), that can be stored in only $M = \log_2(4p)$ bits of memory. More precisely, the states Q of the corresponding automata $\mathcal{A}$ would be all $(j, (x, y))$ where $j \in [\![0, j_{\max}]\!]$ is the step number (*i.e.* number of reads from the beginning) and $(x, y) \in \mathbb{B} \times [\![0, 2p[\![$ are the $4p$ possible values of the internal memory. The initial state is $(0, 0, 0)$, the total number of reads $j_{\max}$ is $\leq p^2$, and the final states are all $(j_{\max}, x, y)$ where y is odd. This automata satisfies condition (i), since a state (j, x, y) can only be reached after reading j inputs, so by theorem Thm.5.4, the output can be homomorphically computed by evaluating less than $\#Q \leq 4p^3$ CMux gates, with some $O(p)$ noise overhead. The number of Mux can decrease by a factor 8 by minimizing the automata. Using the same parameters as the bootstrapping key, for $p = 32$, evaluating one Mux gate takes about 0.0002s, so the whole program (16384 Cmux) would be homomorphically evaluated in 3.2 seconds.

We mapped a problem from its high-level description to an algorithm using very few bits of memory. Since low memory programs are in general more naive, it should be easier to find them than obtaining a circuit with low multiplicative depth that would be required for other schemes such as BGV, FHE over integers. Once a suitable program is found, as in the previous example, compiling it to a net-list of CMux gates is straightforward by our Theorem 5.4.

Algorithm 4 elementary fixed time algorithm that computes the p-th bit of the product of a and b

Input: a and b as little endian bits
Output: p-th bit of ab

Internal memory: $x \in \{0,1\}, y \in [\![0, 2p[\![$
initialize $x = 0, y = 0$
for $k = 0$ to $p-1$ **do**
    for $i = 0$ to $k-1$ **do**
        read a_i; $x = a_i$
        read b_{k-i}; $y = y + xb_{k-i}$
    end for
    read a_k; $x = a_k$
    read b_0; $y = \lfloor (y + xb_0)/2 \rfloor$
end for
for $i = 0$ to p **do**
    read a_i; $x = a_i$
    read b_{p-i}; $y = y + xb_{p-i}$
end for
accept if $y == 1 \mod 2$

6 Practical security parameters

For an asymptotical security analysis, since the phase is lipschitzian, TLWE samples can be equivalently mapped to their closest binLWE (or bin-RingLWE), which in turn can be reduced to standard LWE/ringLWE with full secret using the modulus-dimension reduction [6] or group-switching techniques [13]. It can then be reduced to worst case BDD instances. It is also easy to write a direct and tighter search-to-decision reductions for TLWE, or a direct worst-case to average-case reductions from TLWE to Gap-SVP or BDD.

In this section, we will rather focus on the practical hardness of LWE, and express after all the security parameter λ directly as a function of the entropy of the secret n and the error rate α.

Our analysis is based on the work described in [2]. This paper studies many attacks against LWE, ranging from a direct BDD approach with standard lattice reduction, sieving, or with a variant of BKW [4], resolution via man in the middle attacks. Unfortunately, they found out that there is no single-best attack. According to their results table [2, Section 8, Tables 7,8] for the range of dimensions and noise used for FHE, it seems that the SIS-distinguisher attack is often the best candidate (related to the Lindner-Peikert [17] model, and also used in the parameter estimation of [11]). However, since q is not a parameter in our definition of TLWE, we need to adapt their results. This section relies on the following heuristics concerning the experimental behaviour of lattice reduction algorithms. They have been extensively verified and used in practice.

1. The fastest lattice reduction algorithms in practice are blockwise lattice algorithms (like BKZ-2.0[8], D-BKZ [20], or the slide reduction with large blocksize [14, 20]).
2. Practical blockwise lattice reduction algorithms have an intrinsic quality $\delta > 1$ (which depends on the blocksize), and given a m-dimensional real basis B of volume V, they compute short vectors of norm $\delta^m V^{1/m}$.
3. The running time of BKZ-2.0 (expressed in bit operations) as a function of the quality parameter is: $\log_2(t_{\mathrm{BKZ}})(\delta) = \frac{0.009}{\log_2(\delta)^2} - 27$ (According to the extrapolation by Albrecht et al [1] of Liu-Nguyen datasets [18]).
4. The coordinates of vectors produced by lattice reduction algorithms are balanced. Namely, if the algorithm produces vectors of norm $\|v\|_2$, each coefficient has a marginal Gaussian distribution of standard deviation $\|v\|_2/\sqrt{n}$. Provided that the geometry of the lattice is not too skewed in particular directions, this fact can sometimes be proved, especially if the reduction algorithm samples vectors with Gaussian distribution over the input lattice. This simple fact is at the heart of many attacks based on Coppersmith techniques with lattices.
5. For mid-range dimensions and polynomially small noise, the SIS-distinguisher plus lattice reduction algorithms combined with the search-to-decision is the best attack against LWE; (but this point is less clear, according to the analysis of [1], at least, this attack model tends to overestimate the power of the attacker, so it should produce more conservative parameters).
6. Except for small polynomial speedups in the dimension, we don't know better algorithms to find short vectors in random anti-circulant lattices than generic algorithms. This folklore assumption seems still up-to date at the time of writing.

If one finds a small integer combination that cancels the mask of homogeneous LWE samples, one may use it to distinguish them from uniformly chosen random samples. If this distinguisher has small advantage ε, we repeat it about $1/\varepsilon^2$ times. Then, thanks to the search to decision reduction (which is particularly tight with our TLWE formulation), each successful answer of the distinguisher reveals one secret key bit. To handle the continuous torus, and since q is not a parameter of TLWE either, we show how to extend the analysis of [2] to our scheme.

Let $(\boldsymbol{a_1}, b_1), \ldots, (\boldsymbol{a_m}, b_m)$ be either m LWE samples of parameter α or m uniformly random samples of $\mathbb{T}^{n+1}$, we need to find a small combination $v_1, \ldots, v_m$ of samples such that $\sum v_i \boldsymbol{a_i}$ is small. This condition differs from most previous models, were working on a discrete group, and required an exact solution. By allowing approximations, we may find solutions for much smaller m than the usual bound $n \log q$, even $m < n$ can be valid. Now, consider the $(m+n)$-dimensional lattice, generated by the rows of the following basis $B \in \mathcal{M}_{n+m,n+m}(\mathbb{R})$:

$$B = \left[\begin{array}{ccc|ccc} 1 & & 0 & & & \\ & \ddots & & & 0 & \\ 0 & & 1 & & & \\ \hline a_{1,1} & \cdots & a_{1,n} & 1 & & 0 \\ \vdots & \ddots & \vdots & & \ddots & \\ a_{m,1} & \cdots & a_{m,n} & 0 & & 1 \end{array}\right].$$

Our target is to find a short vector $\boldsymbol{w} = [x_1, \ldots, x_n, v_1, \ldots, v_m]$ in the lattice of B, whose first n coordinates $(x_1, \ldots, x_n) = \sum_{i=1}^{m} v_i \boldsymbol{a_i} \mod 1$ are shorter than the second part $(v_1, \ldots, v_m)$. To take this skewness into account, we choose a real parameter $q > 1$ (that will be optimized later), and apply the unitary transformation f_q to the lattice, which multiplies the first n coordinates by q and the last m coordinates by $1/q^{n/m}$. Although this matrix looks like a classical LWE matrix instance, the variable q is a real parameter, and it doesn't need to be an integer. It then suffices to find a regular short vector with balanced coordinates in the transformed lattice, defined by this basis:

$$f_q(B) = \left[\begin{array}{ccc|ccc} q & & 0 & & & \\ & \ddots & & & 0 & \\ 0 & & q & & & \\ \hline qa_{1,1} & \cdots & qa_{1,n} & \frac{1}{q^{n/m}} & & 0 \\ \vdots & \ddots & \vdots & & \ddots & \\ qa_{m,1} & \cdots & qa_{m,n} & 0 & & \frac{1}{q^{n/m}} \end{array}\right], \text{ with } q \in \mathbb{R} > 1.$$

The direct approach is to apply the fastest algorithm (BKZ-2.0 or slide reduction) directly to $f_q(B)$, which outputs a vector $f_q(\boldsymbol{w})$ of standard deviation $\delta^{n+m}/\sqrt{n+m}$ where $\delta \in]1, 1.1]$ is the quality of the reduction.

Once we have a vector $\boldsymbol{w}$, all we need is to analyse the term $\sum_{i=1}^{m} v_i b_i = \sum_{i=1}^{m} v_i(\boldsymbol{a_i}\boldsymbol{s} + e_i) = \boldsymbol{s} \cdot \sum_{i=1}^{m}(v_i \boldsymbol{a_i}) + \sum_{i=1}^{m} v_i \boldsymbol{e_i} = \boldsymbol{s} \cdot \boldsymbol{x} + \boldsymbol{v} \cdot \boldsymbol{e}$.

It has Gaussian distribution of square parameter $\sigma^2 = \frac{\delta^{2(m+n)}\pi}{2q^2} \cdot \frac{nS^2}{m+n} + \frac{q^{2n/m}\delta^{2(m+n)}\alpha^2 m}{m+n} = \delta^{2(m+n)}\left(\frac{\pi S^2}{2q^2} \cdot \frac{n}{m+n} + q^{2n/m}\alpha^2 \frac{m}{m+n}\right)$. Here $S = \frac{\|\boldsymbol{s}\|}{\sqrt{n}} \approx \frac{1}{\sqrt{2}}$. By definition of the smoothing parameter, it may be distinguished from the uniform distribution with advantage ε as long as $\sigma^2 \geq \eta_\varepsilon^2(\mathbb{Z})$. To summarize, the security parameter of LWE is (bounded by) the solution of the following system of equations

$$\lambda(n, \alpha) = \log_2(t_{\text{attack}}) = \min_{0<\varepsilon<1} \log_2\left(\frac{n}{\varepsilon^2} t_{\text{BKZ}}(n, \alpha, \varepsilon)\right) \tag{3}$$

$$\log_2(t_{\text{BKZ}})(n, \alpha, \varepsilon) = \frac{0.009}{\log_2(\delta)^2} - 27 \tag{4}$$

$$\ln(\delta)(n, \alpha, \varepsilon) = \max_{\substack{m>1 \\ q>1}} \frac{1}{2(m+n)}\left(\ln(\eta_\varepsilon^2(\mathbb{Z})) - \ln\left(\frac{\pi S^2}{2q^2}\frac{n}{m+n} + q^{\frac{2n}{m}}\alpha^2 \frac{m}{m+n}\right)\right) \tag{5}$$

$$\eta_\varepsilon(\mathbb{Z}) \approx \sqrt{\frac{1}{\pi}\ln(\frac{1}{\varepsilon})}. \tag{6}$$

Here, Eq. (3) means that we need to run the distinguisher $\frac{1}{\varepsilon^2}$ times per unknown key bit (by Chernoff's bound), and we need to optimize the advantage ε accordingly. Eq.(4) is the heuristic prediction of the running time of lattice reduction. In Eq.(5) q and m need to be chosen in order to maximize the targeted approximation factor of the lattice reduction step.

Differentiating Equation (5) in q, we find that its maximal value is

$$q_{\text{best}} = \left(\frac{\pi S^2}{2\alpha^2}\right)^{\frac{m}{2(m+n)}}.$$

Replacing this value and setting $t = \frac{n}{m+n}$, Equation (5) becomes:

$$\ln(\delta)(n,\alpha,\varepsilon) = \max_{t>0} \frac{1}{2n}\left(t^2\ell_2 + t(1-t)\ell_1\right) \text{ where } \begin{cases}\ell_1 = \ln\left(\frac{\eta_\varepsilon^2(\mathbb{Z})}{\alpha^2}\right)\\ \ell_2 = \ln\left(\frac{2\eta_\varepsilon^2(\mathbb{Z})}{\pi S^2}\right)\end{cases}.$$

Finally, by differentiating this new expression in t, the maximum of δ is reached for $t_{\text{best}} = \frac{\ell_1}{2(\ell_1-\ell_2)}$, because $\ell_1 > \ell_2$, which gives the best choices of m and q and δ. Finally, we optimize ε numerically in Eq.(3).

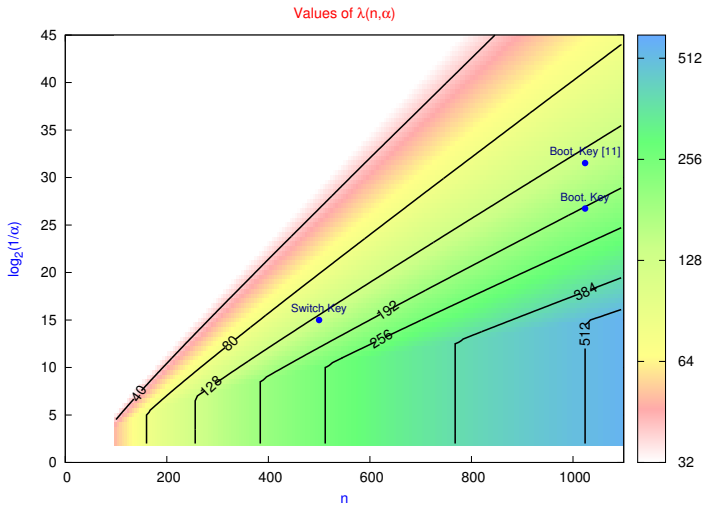
All previous results are summarized in Figure 6, which displays the security parameter λ as a function of $n, \log_2(\alpha)$.

In particular, in the following table we precise the values for the keyswitching key and the bootstrapping key (for our implementation and for the one in [11]).

	n	α	$\boldsymbol{\lambda}$	$\varepsilon_{\text{best}}$	m_{best}	q_{best}	δ_{best}
Switch key	500	2^{-15}	136	2^{-12}	444	125.7	1.0058
Boot. key	1024	$9.0 \cdot 10^{-9}$	194	2^{-10}	968	7664.	1.0048
Boot.key, [11]	1024	$3.25 \cdot 10^{-10}$	141	2^{-7}	993	44096	1.0055

The table shows that the strength of the lattice reduction is compatible with the values announced in [11]. Our model predicts that the lattice reduction phase is harder ($\delta = 1.0055$ in our analysis and $\delta = 1.0064$ in [11]), but the value of ε is bigger in our case. Overall, the security of their parameters-set is evaluated by our model to 136-bits of security, which is larger than the $\geq$ 100-bits of security announced in [11]. The main reason is that we take into account the number of times we need to run the SIS-distinguisher to obtain a non negligible advantage. Since our scheme has a smaller noise propagation overhead, we were able to raise the input noise levels in order to strengthen the system, so with the parameters we chose in our implementation, our model predicts 194-bits of security for the bootstrapping key and 136-bits for the keyswitching key (which remains the bottleneck).

Fig. 1. Security parameter λ as a function of n and α for LWE samples



This curve shows the security parameter levels λ (black levels) as a function of $n = kN$ (along the x-axis) and $\log_2(1/\alpha)$ (along the y-axis) for TLWE *(also holds for bin-LWE), considering both the attack of this section and the collision attack in time $2^{n/2}$.*

7 Conclusion

In this paper, we presented a generalization of the LWE and GSW homomorphic encryption schemes. We improved the execution timing of the bootstrapping procedure and we reduced the size of the keys by keeping at least the same security as in previous fast implementations. This result has been obtained by simplifying the multiplication morphism, which is the main operation used in the scheme we described. As a proof of concept we implemented the scheme itself and we gave concrete parameters and timings. Furthermore, we extend the applicability of the external product to leveled homomorphic encryption. We finally gave a detailed security analysis. Now the main drawback to make our scheme adapted for real life applications is the expansion factor of the ciphertexts of around 400000 with fairly limited batching capabilities.

Acknowledgements This work has been supported in part by the CRYPTOCOMP project.

References

1. M. R. Albrecht, C. Cid, J. Faugère, R. Fitzpatrick, and L. Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74/2:325–354, 2015.

2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.
3. J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *Crypto*, pages 297–314, 2014.
4. A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. of ACM*, 50(4):506–519, 2003.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
6. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D.Stehlé. Classical hardness of learning with errors. In *Proc. of 45th STOC*, pages 575–584. ACM, 2013.
7. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
8. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *Proc. of Asiacrypt*, pages 1–20, 2011.
9. J. H. Cheon and D. Stehlé. Fully homomophic encryption over the integers revisited. In *Advances in Cryptology–EUROCRYPT 2015*, pages 513–536. Springer, 2015.
10. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. A homomorphic lwe based e-voting scheme. In *Post-Quantum Cryptography*, pages 245–265. Springer, 2016.
11. L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Eurocrypt*, pages 617–640, 2015.
12. M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".
13. N. Gama, M. Izabachène, P. Q. Nguyen, and X. Xie. Structural lattice reduction: Generalized worst-case to average-case reductions. *IACR Cryptology ePrint Archive*, 2014:48, 2014.
14. N. Gama and P. Q. Nguyen. Predicting Lattice Reduction. In *Eurocrypt*, 2008.
15. C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, pages 169–178, 2009.
16. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Crypto*, pages 75–92, 2013.
17. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Proc. of CT-RSA*, volume 6558 of *LNCS*. Springer-Verlag, 2011.
18. M. Liu and P. Q. Nguyen. Solving bdd by enumeration: An update. In *Proc. of CT-RSA*, volume 7779 of *LNCS*, pages 293–309. Springer-Verlag, 2013.
19. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *In Proc. of EUROCRYPT, volume 6110 of LNCS*, pages 1–23. Springer, 2010.
20. D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. In *Proc. of Eurocrypt 2016*, volume 9665 of *LNCS*, pages 820–849. Springer-Verlag, 2016.
21. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.