Plug-and-play sanitization for TFHE

Florian Bourse[1] and Malika Izabachène[2]

[1] Independent scholar
[2] Cosmian, Paris, France

Abstract. Fully Homomorphic encryption allows to evaluate any circuits over encrypted data while preserving the privacy of the data.
Another desirable property of FHE called circuit privacy enables to preserve the privacy of the evaluation circuit, i.e. all the information on the bootstrapped ciphertext, including the computation that was performed to obtain it, is destroyed.
In this paper, we show how to directly build a circuit private FHE scheme from TFHE bootstrapping (Asiacrypt 2016). Our proof frame is inspired from the techniques used in Bourse etal (Crypto 2016), we provide a statistical analysis of the error growth during the bootstrapping procedure where we adapt discrete Gaussian lemmata over rings. We make use of a randomized decomposition for the homomorphic external product and introduce a public key encryption scheme with invariance properties on the ciphertexts distribution. As a proof of concept, we provide a C implementation of our sanitization strategy.

Keywords: Fully Homomorphic Encryption, circuit privacy, leftover hash lemma, sanitization, bootstrapping implementation.

1 Introduction

A Fully Homomorphic Encryption (FHE) scheme allows to evaluate any circuit over encrypted data without having to decrypt them. A standard requirement for homomorphic encryption schemes is semantic security against chosen plaintext attacks, which guarantees that the data remains unknown to the party who evaluates the circuit.

For most of the known FHE schemes, semantic security relies on the hardness of the Learning With Errors (LWE) problem [Reg05], or variants of LWE, meaning that the message is slightly offset by some noise that grows after a homomorphic computation. In order to be able to continue computation, a procedure called bootstrapping, first described in [Gen09], is used to refresh the noise in a ciphertext to a fixed manageable level.

The bootstrapping procedure is costly compared to basic operations, but packing techniques (e.g., [Bra12,BGV12,FV12,CKKS17,CZ17,MS18]) enable efficient amortized timings. Unfortunately, these techniques still have a very high latency (even though lots of ciphertexts can be bootstrapped at once, the time it takes for bootstrapping a single ciphertext remains prohibitively large) and are not always compatible with certain real-time applications. A series of works (e.g.,[AP14,DM15,CHK$^+$18,CGGI20]) tackles this issue by optimizing the bootstrapping of a single ciphertext, which is desirable in some scenarios.

Circuit privacy. Another property called circuit privacy is also crucial in many scenarios where the evaluation circuit contains sensitive information; typical examples being classification algorithms or financial prediction algorithms or many other scenarios where the delegated computation party may act as a service provider. Intuitively, the idea behind circuit privacy is that no one can reverse-engineer the computation into which a ciphertext went through.

A bit more formally, an FHE scheme achieves circuit privacy for a class of functions if the output distribution of the evaluation procedure of the FHE scheme only depends on the result and does not leak information on which function from the class was evaluated. In addition, knowing the secret key should not give any advantage.

The property of circuit privacy has also found applications in cryptographic protocols: a first example is the computation of a private set intersection of two datasets based on homomorphic encryption [CLR17,CMdG+21] where a receiver has input set X and sends its dataset encrypted to a sender with input Y. The latter performs some homomorphic computation so that at the end, the receiver outputs $X \cap Y$. In order to provide security against a semi-honest receiver, one technique relies on using a circuit private homomorphic encryption scheme. Another example can be found in the lattice-based threshold signature constructions [BGG+18,ASY22], built from threshold FHE, which also require to hide information in the computation of the partial decryption shares.

Different flavors of circuit privacy may be desired and some relations between them can be established. Circuit privacy against malicious adversaries, where the public keys and/or ciphertexts are not necessarily honestly generated, is much stronger than circuit privacy against passive adversaries. Fortunately, [OPP14] devised a technique to upgrade a compact FHE scheme circuit private against passive adversaries using another (possibly non-compact) FHE scheme circuit private against malicious adversaries, which can even possibly start from a non circuit private compact FHE schemes using some additional twists. Also, depending on the application, the result of the computation might be sent back and decrypted directly, but also might be sent to another server for additional computation. This refinement of the circuit private property has been defined in [GHV10] as 1-hop for the first case, or multi-hop for the second case. An FHE scheme that is i-hop circuit private allows for i steps of computation before the ciphertext has to be decrypted. Finally, an FHE scheme could achieve circuit privacy for a particular class of functions but might fail to evaluate other functions in a circuit private way.

Another closely related property is sanitization of FHE ciphertexts as defined in [DS16], which asks that the FHE scheme includes a $\mathsf{Sanitize}$ algorithm which maps ciphertexts to a canonical distribution depending only on its underlying plaintext. It is straightforward to see that sanitization implies circuit privacy: appending $\mathsf{Sanitize}$ at the end of each evaluation procedure makes it possible for the FHE scheme to reach multi-hop circuit privacy for all functions. The other direction requires some additional assumptions and technical details such as circular security assumption, but intuitively, a circuit private evaluation of the bootstrapping procedure would yield a sanitization algorithm.

We also note that the notion of circuit privacy for FHE is very closely related to circuit privacy in multiparty computation (MPC), as FHE can be used as a building block in low communication MPC protocols.

1.1 Previous works

The first circuit private technique was proposed in [Gen09] and relies on noise flooding. The noise contained in a ciphertext is the main source of leakage. The idea of noise flooding is to add a noise with very large parameter at the end of the computation to drown the existing noise and to avoid an adversary making use of the information it contains. This yields a 1-hop scheme that is circuit private for the class of all functions. This technique requires the starting modulus-to-noise ratio to be superpolynomial, which makes the security rely on a stronger LWE assumption, and requires larger parameters.

A new approach was proposed in [DS16] to avoid this downside, and achieves circuit-privacy for FHE schemes relying on LWE with a polynomial modulus-to-noise ratio, based

on a sanitization algorithm which relies on the use of the bootstrapping. Their sanitization technique is based on a paradigm they call *soak and spin*, which consists of iterations of two consecutive steps: a first step invokes a Refresh algorithm which reduces the noise of the input ciphertext to a fixed level; a second step, ReRand injects noise in the ciphertext in order to make it closer to a canonical distribution of ciphertexts. After λ consecutive iterations of Refresh and Rerand over any two input ciphertexts decrypting to the same message, the statistical distance between the two output ciphertexts is bounded by $2^{-\lambda}$. An additional requirement to their work is that the output of Rerand should decrypt to the correct message with very high probability in order to reach the sanitization property. The authors of [AP20] proposed a refined security analysis of the sanitization algorithm proposed in [DS16] based on the used of the Rényi divergence instead of the statistical distance. They show that the number of iterations can be reduced depending on the target number of evaluations that needs to be hidden. They also set the parameters in the soak and spin process such that decryption fails with exponentially small probability.

A refinement of this technique was proposed in [BdMW16] to achieve circuit privacy for NC^1 circuits without relying on the circular security assumption, and basing its security on plain LWE with polynomial modulus-to-noise ratio. The main idea of this paper is to tweak the computation paradigm to branching programs, which they show can be evaluated in a circuit private way using a two steps randomization process at each stage of the computation. This randomization process consists of both switching the homomorphic product to its randomized counterpart from [AP14], and adding a random Gaussian noise. The first step ensures that the distribution of the noise after one step of computation is close to a Gaussian noise, with a parameter independent of the previous states of the evaluation; this is done by making use of the randomized product. Adding a random Gaussian noise in the second step ensures that at each step, the noise can have any value, and not falls in a subset that could leak information about the computation being carried on. The downside of this work is that it applies to schemes that are not implemented in practice for efficiency reasons, namely the GSW cryptosystem [GSW13] and its variants [BV14], and requires the functions to be converted to branching programs.

1.2 Our contributions

We propose a practical sanitization approach for FHE schemes with polynomial modulus-to-noise ratio. Our solution is based on the bootstrapping procedure implemented using GSW homomorphic product and applies to FHEW-like cryptosystems. In particular, our sanitization algorithm is conceptually compatible with recent developement on the TFHE framework which allows the evaluation of arbitrary functions or the evaluation of arithmetic functions via large precision bootstrappings [BDF18,CIM19,GBA21,CLOT21] for example, provided the parameters are adapted.

In practice, we obtain that the overhead of the sanitization property is very low when comparing sanitizing and non-sanitizing TFHE bootstrapping for the same set of parameters. For the sake of comparison, we evaluate the cost of [DS16] rerandomization strategy for TFHE bootstrapping. With encryption of zero at hand which can be generated during a pre-processing phase, our comparison gives that our strategy is at least 3 times faster for the same set of parameters with a lower decryption failure probability. A comparison is detailed in the supplementary materials, Section C.

Our solution for sanitization relies on the same assumption as for FHE without sanitization and achieves strong properties in that our definition of sanitization is proven in the simulation based model and holds for a bounded simulator, which implies the indistinguishible based sanitization property as defined in [DS16]. Also, as in their work,

our construction implies multi-hop circuit privacy for the class of all functions, not only functions in NC^1 as in [BdMW16].

To this end, we generalize the randomization techniques from [AP14,BdMW16] to the ring setting (see Lemma 13). We also identify a slight inaccuracy in [BdMW16] that comes from the use of the Leftover Hash Lemma, invoking the 2-universality of vector dot products as a randomness extractor. While this argument works when the modulus is prime, it requires a bit more effort when this is not the case, and the bounds are usually worse. We solve this issue by adapting the techniques from [MM11] and [LW20, Theorem 5.5] to our setting (see proofs of Lemmata 11 and 19).

We also develop new techniques to analyze the distributions of polynomials – combinations of uniformly random polynomials and discrete Gaussian polynomials over lattices – by diving into the structural details of the space we are interested in, namely a power-of-two cyclotomic ring with a power-of-two modulus. We believe that the results we derive for our specific use-case, e.g., the different bounds on smoothing parameters and the collision probabilities for random variables, could be applied in other contexts, and try to present them in a generic and educational way. One of these techniques includes building a new public key encryption scheme (Construction 1) for polynomial messages, with ciphertexts of zero whose distribution is invariant when multiplied by a power of X. We believe that this property is of independent interest and give a toy example of a protocol making use of it.

As a side contribution, we also present an alternative technique in Section 6 that would yield to circuit privacy in a more direct way under some conditions on the parameters. Unfortunately, the conditions captured by Lemma 19 would require much larger parameters than the set we select for efficiency reason with our construction.

Finally, we provide an implementation of our sanitization algorithm, and show a possible set of parameters that provably achieves sanitization, together with its execution time. Our solution assumes that we have a large number of pre-computed encryptions of zero at hand. In order to give interesting experimental results, we emulate an unbounded precomputation by using the same encryption of zero multiple times. There is room of improvement with respect to finding a more efficient instantiation of our Lemma 10, which would have a positive impact on the timings of our sanitization technique.

Overview of the techniques. The underlying idea of our work is based on two observations. A first remark is that the bootstrapping procedure of TFHE can be decomposed in 3 steps, which are called BlindRotate, Extract, and KeySwitch, as illustrated on Figure 1. And if any of these 3 steps destroys unwanted information of the input, all the information about any previous evaluation on the ciphertext is washed away, and the bootstrapping procedure becomes a sanitizing algorithm.

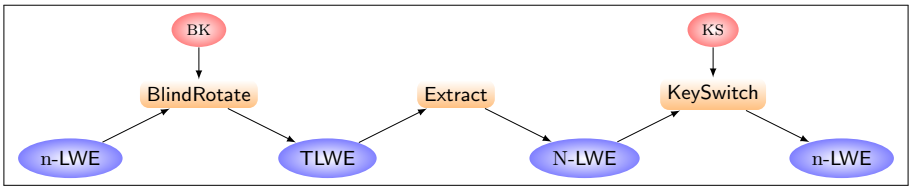

Fig. 1. TFHE bootstrapping steps: n-LWE denotes a LWE ciphertext of dimension n. After an initialization step not pictured here, the bootstrapping of an n-LWE ciphertext into a fresh n-LWE ciphertext goes through three algorithms BlindRotate, Extract, and KeySwitch and requires two sets of keys: the bootstrapping key BK and the keyswitching key KS. More details will be given later in the paper.

Our focus is the BlindRotate procedure which we modify to be a circuit private evaluation for the class of function $\mathsf{BlindRotate}_{\mathbf{a},b} : \mathbf{s} \mapsto (0, \mathsf{testv}) \cdot X^{b-\sum_i s_i a_i}$ parametrized by a LWE ciphertext $(\mathbf{a}, b)$ for any polynomial testv and a variable $\mathbf{s} \in \{0,1\}^n$.

We notice that BlindRotate computation is mostly made of homomorphic evaluations of multiplexers, which behaves nicely for circuit privacy, as shown in [BdMW16]. We show how to modify the BlindRotate procedure to make it circuit-private, by randomizing the gadget decomposition, and by adding fresh encryptions of zero as well as a Gaussian noise at each step.

In order to prove that the resulting CP-BlindRotate class of functions is indeed circuit-private, the main difficulty comes from that fact that most of the random variables are related and most of the tools at hand require independent distributions or no leakage. To avoid some of these dependencies, we follow the strategy that [BdMW16] which were applied to branching programs and analyze the distribution of the ciphertext after one step of computation at a time. This allows us to replace each state of the evaluation by a random variable independent of the computation already carried and then prove by induction that the resulting distribution at the end of the algorithm only depends on the final message and public parameters.

An additional difficulty compared to the evaluation of branching programs is that the computation does not just carry information through the different stages of the evaluation, there are also some multiplications by powers of X being done on the underlying plaintext, and also on the previous states of the accumulator. Also, we are dealing with polynomials and not scalar vectors, which means that the different components are combined with each other along products, so we cannot just directly apply the standard tools because of the dependencies between components. To tackle these issues, we carefully craft distributions that are spherical so that multiplying them by a power of X doesn't change the distribution, and we explicitly state the matrix-vector products that appear when doing polynomial multiplications in order to exploit known results on lattices.

To give a bit more details on one step of the induction, the distribution of LWE ciphertexts is given by two elements beside its associated plaintext, for which we need to analyze the distribution: a vector that should be sampled uniformly at random, and the noise component. We generalize the noise analysis [BdMW16] to the ring setting, by viewing products of polynomials in our specific case as matrix-vector products, and deriving all the bounds needed to apply the underlying lemmata. As in [BdMW16], we leverage the fact that the decomposition is now randomized, with a large enough parameter in order to prove that the noise is statistically close to a discrete Gaussian distribution, as a linear combination of discrete Gaussian distributions, and we add a small Gaussian variable that ensures that the support of this discrete Gaussian is not a sublattice, i.e., all the values can be reached by this noise and it is not restricted to a subset of values that could leak information about the evaluation being carried on.
On the other hand, the way we handle the uniformly random part of the ciphertext is completely different. In [BdMW16], this part of the proof is handled by the Leftover Hash Lemma. This requires quite large vectors, especially since the modulus (a power of 2) is not prime. For example, when half of the elements are not invertible (which is the case both with integers modulo a power of 2 and with polynomials modulo $X^N + 1$ modulo a power of 2), even when sampling two uniformly random vectors $\mathbf{a}$ and $\mathbf{b}$ of size m, their inner-product can't be statistically close to uniform unless m is at least linear in the security parameter λ. This is because with probability $\frac{1}{2^m}$, none of the elements of $\mathbf{a}$ are invertible, so the result is not invertible. In TFHE, the parameters are optimized for efficient bootstrapped operations, the polynomial vectors are of size at most 6 and of size

10 using our implementation, so this approach cannot be used for a practical implementation of circuit privacy or sanitization. For the sake of completeness, and in the eventuality that the parameters in use in the future will evolve, we also provide a proof that this technique works when the parameters are large enough, and we discuss the underlying conditions in Section 6.
In our work, we choose to add a fresh ciphertext with a correctly generated uniformly random element at each step of computation instead. This fresh ciphertext can be obtained via a public encryption key that is a collection of encryptions of zero, from which a random subset-sum is computed, multiplied by a random power of X. The distributions of those ciphertexts has to be carefully crafted to be spherical even when the sanitization key is given out. Now, we can use the Leftover Hash Lemma with a large enough number of encryptions of zero in the sanitization key without impacting the efficiency of the other algorithms, especially since those ciphertexts can be precomputed offline.

Organization. In Section 2, we present the notations used through the paper and recall the preliminaries about Gaussian distributions, and the TFHE scheme. In Section 3, we present our generalization of the randomization techniques of [BdMW16] that will be used for our result, as well as our new public key encryption scheme that verifies the required properties. In Section 4, we present our techniques for circuit privacy and show how to build a sanitization algorithm for TFHE. In Section 5, we discuss the practical parameters of our implementation. In Section 6, we present a different and more direct strategy to achieve sanitization and specify the conditions required on the parameters.

2 Preliminaries

In this section, we give notations, mathematical definitions and lemmata we will use for our proofs.

2.1 Notations and definitions

In this paper, we will note λ a security parameter. The set of integers from 1 to n will be noted $[n]$ for convenience. We use lower case bold font, e.g. $\mathbf{a}$, to denote (possibly row) vectors, and upper case bold font, e.g. $\mathbf{A}$, to denote matrices. We write the left-right concatenation of matrices using $|$, e.g. $(\mathbf{A} \mid \mathbf{B})$. We will use $\otimes$ to denote the Kronecker product of two matrices. We let q be an integer modulus such that $q = B^\ell$ with B a power of 2. We denote $\mathbb{T}$ the set of real numbers modulo one and the discretized torus $\hat{\mathbb{Z}}_q = \{0, \frac{1}{q}, \cdots, \frac{q-1}{q}\}$ is $\frac{1}{q}\mathbb{Z} \cap \mathbb{T}$. Note that $\hat{\mathbb{Z}}_q$ is isomorphic to $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. In particular, the multiplication over $\hat{\mathbb{Z}}_q$ is given by $x \cdot y = qx \cdot y$, for all $x, y \in \hat{\mathbb{Z}}_q$.

We set $\mathbb{B} = \{0, 1\}$ and $\mathbb{B}_N(X)$ the set of polynomials of degree at most N with coefficients in $\mathbb{B}$. We denote $R = \mathbb{Z}[X] \mod X^N + 1$ the set of integer polynomials of degree at most N, where N is a power-of-two. We set $R_q = \mathbb{Z}_q[X] \mod X^n + 1$ and $\hat{R}_q$ the set of polynomials with coefficients over $\hat{\mathbb{Z}}_q$ modulo $X^N + 1$. Vectors are denoted as row vectors, and $\mathbf{a}^t$ denotes the column vector which is the corresponding transpose vector of $\mathbf{a}$. The absolute value of an element a in $\mathbb{T}$, denoted $|a|$, is the absolute value of its representative that is closest to 0 (i.e., in $]-0.5, 0.5]$). We also use this representative when we define norms for vectors, matrices, and polynomials, and also use those for elements in $\hat{\mathbb{Z}}_q$. The euclidean norm of a vector $\mathbf{a}$ is denoted $\|\mathbf{a}\|_2$ and its infinity norm is denoted $\|\mathbf{a}\|_\infty$. The spectral norm of a square matrix $\mathbf{A}$, i.e. its largest singular value, is denoted $\|\mathbf{A}\|_2$.

In order to avoid cumbersome notations throughout the paper, we will use the following shorthand notation to define anticyclic matrices corresponding to polynomial multiplications modulo $X^N + 1$.

We denote $\mathtt{pow}_X = \left(1, X, \ldots, X^{N-1}\right) \in R^{1\times N}$. For any polynomial $p = \sum_{i=0}^{N-1} p_i X^i \in \hat{R}_q$, there exists a matrix $\mathbf{P} = \begin{pmatrix} p_0 & -p_{N-1} & \cdots & \cdots & -p_1 \\ p_1 & p_0 & -p_{N-1} & \cdots & -p_2 \\ \vdots & & \ddots & \vdots & \\ p_{N-1} & \cdots & \cdots & p_1 & p_0 \end{pmatrix} \in \hat{\mathbb{Z}}_q^{N\times N}$ such that $\mathtt{pow}_X \cdot \mathbf{P} = \mathtt{pow}_X \otimes p$.

We also use the following euclidean norm for polynomials in $\hat{R}_q$: $\|p\|_2 = \sqrt{\sum_{i=0}^{N-1} p_i^2}$.

Gadget vector. We let $\mathbf{g} = (1/B, \ldots, 1/B^\ell) \in \hat{R}_q^{1\times\ell}$ be the vector of powers of B and let define the gadget matrix as $\mathbf{G} = \mathbf{I}_{d+1} \otimes \mathbf{g}^t$, i.e.

$$\mathbf{G} = \begin{pmatrix} 1/B \cdots 1/B^\ell & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1/B \cdots 1/B^\ell & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \quad 0 \\ 0 & \cdots & & 0 & \cdots & 0 \quad 1/B \cdots 1/B^\ell \end{pmatrix}^t \in \hat{R}_q^{(d+1)\cdot\ell\times(d+1)} \tag{1}$$

2.2 Random variables

We write $\mathbf{y} \leftarrow \mathcal{P}$ when y is sampled from distribution $\mathcal{P}$.

Variance and covariance. For a random variable X, we denote $E[X]$ the expected value of X. The covariance of two random variables X and Y is $E\left[(X - E[X])(Y - E[Y])\right]$. The covariance matrix $\mathsf{Var}\left(\mathbf{X}\right)$ of a vector of random variables $\mathbf{X} = (X_1, \ldots, X_N)$ is the matrix whose coefficient on row i, column j is the covariance of X_i and X_j. For any two real value vectors of random variables $\mathbf{X}$ and $\mathbf{Y}$ and any real α, we have:

$$\mathsf{Var}\left(\alpha\mathbf{X} + \mathbf{Y}\right) = \alpha^2\mathsf{Var}\left(\mathbf{X}\right) + \mathsf{Var}\left(\mathbf{Y}\right).$$

Statistical distance and min-entropy. The statistical distance between two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ over a discrete domain X is defined as

$$\Delta\left(\mathcal{P}, \mathcal{Q}\right) = \tfrac{1}{2}\sum_{a\in X} |\mathcal{P}(a) - \mathcal{Q}(a)|$$

We say that two distributions $\mathcal{P}$ and $\mathcal{Q}$ are (statistically) close and we write $\mathcal{P} \approx_s \mathcal{Q}$ if their statistical distance is negligible in λ. The min-entropy of a random variable X is defined as $H_\infty(X) = -\log \max_x \Pr[X = x]$. It gives a bound on the probability of collision for two independent identically distributed random variables X and X': $\Pr(X = X') \leq 2^{-H_\infty(X)}$ The min-entropy is a useful tool when analyzing conditional probability distribution thanks to the following property: for any random variable Y on a set $\mathcal{Y}$,

$$H_\infty\left(X \mid Y\right) \geq H_\infty(X) - \log|\mathcal{Y}|.$$

2.3 Gaussian distribution over lattices

Lattices. An m-dimensional lattice is a discrete subgroup of $\mathbb{R}^m$. Given k linearly independent vectors of $\mathbb{R}^m$ $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k)$, the lattice generated by $\mathbf{B} = (\mathbf{b}_1 \mid \mathbf{b}_2 \mid \ldots \mid \mathbf{b}_k)$ is of rank k and is denoted:

$$\Lambda(\mathbf{B}) = \left\{ \sum_{i=1}^{k} \mathbf{x}_i \mathbf{b}_i \in \mathbb{R}^m \mid \mathbf{x}_i \in \mathbb{Z} \right\}.$$

For any $\mathbf{v} \in \hat{\mathbb{Z}}_q^{d+1}$, we denote the cosets of the lattice orthogonal to the gadget matrix $\mathbf{G}$ defined in equation (1) by

$$\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}) = \{\mathbf{u} \in \mathbb{Z}^{(d+1)\cdot\ell} \mid \mathbf{G}^t\mathbf{u} \in \mathbf{v} + \mathbb{Z}^{d+1}\}, \qquad \Lambda^{\perp}(\mathbf{G}) = \Lambda_{\mathbf{0}}^{\perp}(\mathbf{G})$$

For the sake of readability, we also extend the notation $\Lambda^{\perp}(\mathbf{G})$ to the set of vectors of polynomials, by identifying a polynomial $u = \sum_{i=0}^{N-1} u_i X^i$ with the vector of its coefficients $\mathbf{u} = (u_0 \mid u_1 \mid \ldots \mid u_{N-1})$:

$$\Lambda^{\perp}(\mathbf{G}) = \{\mathbf{u} \in R^{(d+1)\cdot\ell} \mid \mathbf{G}^t\mathbf{u} \in R^{d+1}\}$$

which is a full rank lattice of dimension $N(d+1)\cdot\ell$.

Gaussian Distributions. The *ellipsoidal Gaussian distribution* over $\mathbb{R}^n$ centered at $\mathbf{0}$ with covariance matrix $\Sigma = S^tS$ where $S \in \mathbb{R}^{m\times n}$ is a rank-n matrix, is defined as:

$$\rho_S(\mathbf{x}) = \exp(-\pi\mathbf{x}^t(S^tS)^{-1}\mathbf{x})$$

When $S = s\mathbf{I}_n$, the spherical Gaussian distribution ρ_S is also denoted ρ_s.

The *ellipsoidal Gaussian distribution* with parameter S over a countable set (a lattice Λ or a coset $\Lambda + \mathbf{x}$) is defined as

$$\forall \mathbf{x} \in C, \quad \mathcal{D}_{C,S}(\mathbf{x}) = \frac{\rho_S(\mathbf{x})}{\rho_S(C)}$$

Lemma 1 (Preimage sampling [GPV08,Pei10,MP12,AP14]). *There is an efficient randomized decomposition function which on input $\mathbf{v} \in \hat{\mathbb{Z}}_q^{d+1}$ outputs a sample $\mathbf{u} \in \mathbb{Z}^{(d+1)\ell}$ from a distribution negligibly close to $\mathcal{D}_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}),\gamma}$ with parameter $\gamma = \tilde{O}(1)$.*

We note that we can reach the target distribution exactly using the sampler from [BLP+13].

2.4 Additional lemmata

In the following, we will denote $\eta_\epsilon(\Lambda)$ the smoothing parameter of a lattice Λ. Intuitively, it is a Gaussian parameter value beyond which discrete Gaussian distributions over Λ behaves almost as continuous Gaussian distributions, ϵ being a bound on the statistical distance that appears in the following lemmata. The next result gives a bound on the smoothing parameter of a generic lattice.

Lemma 2 ([MR07, Lemma 3.3]). *Let Λ be any rank-m lattice and ε be any positive real. Then*

$$\eta_\varepsilon(\Lambda) \leq \lambda_m(\Lambda) \cdot \sqrt{\frac{\ln(2m(1+1/\varepsilon))}{\pi}}$$

where $\lambda_m(\Lambda)$ is the smallest R such that the ball $\mathcal{B}_R$ centered in the origin and with radius R contains m linearly independent vectors of Λ.

Lemma 3 ([Ban93]). *For any n-dimensional lattice Λ and parameter $s > 0$, the euclidean norm of a sample $\mathbf{u}$ from $\mathcal{D}_{\Lambda,s}$, $\|\mathbf{u}\|_2 \leq s\sqrt{n}$, except with probability at most 2^{-2n}.*

Lemma 4 ([Reg05, Claim 3.8]). *Let $\Lambda \subseteq \mathbb{Z}^m$ be any lattice, $\mathbf{c} \in \mathbb{R}^m$, $\varepsilon > 0$ and $r \geq \eta_\varepsilon(\Lambda)$. Then*

$$\rho_r(\Lambda + \mathbf{c}) \in \frac{r^m}{\det(\Lambda)(1 \pm \varepsilon)}$$

Lemma 5 ([AGHS13, Lemma 4]). *For any rank-m lattice Λ, $0 < \varepsilon < 1$, vector $\mathbf{c} \in \mathbb{R}^m$, and full-rank matrix $S \in \mathbb{R}^{m \times m}$, such that $\sigma_m(S) \geq \eta_\varepsilon(\Lambda)$, we have*

$$\rho_S(\Lambda + \mathbf{c}) \in \left[\frac{1-\varepsilon}{1+\varepsilon},\ 1\right] \cdot \rho_S(\Lambda).$$

where $\sigma_m(S)$ is the smallest singular value of S.

Lemma 6 (Simplified version of [Pei10, Theorem 3.1]). *Let $\varepsilon > 0, r_1, r_2 > 0$ be two Gaussian parameters, and Λ be a lattice. If $\frac{r_1 r_2}{\sqrt{r_1^2 + r_2^2}} \geq \eta_\varepsilon(\Lambda)$, then*

$$\Delta(\mathbf{y}_1 + \mathbf{y}_2, \mathbf{y}') \leq 8\varepsilon$$

where $\mathbf{y}_1 \leftarrow \mathcal{D}_{\Lambda,r_1}, \mathbf{y}_2 \leftarrow \mathcal{D}_{\Lambda,r_2}$, and $\mathbf{y}' \leftarrow \mathcal{D}_{\Lambda,\sqrt{r_1^2+r_2^2}}$

Leftover Hash Lemma. We use the following variant of the Leftover Hash Lemma, which can be found in [ALS16] as a particular case of [MM11, Lemma 2.3]

Lemma 7. *Let $q = p^k$ for p prime and $k \geq 1$. Let $m \geq n \geq 1$. Take $\mathcal{X}$ a distribution over $\mathbb{Z}^m$. Let D_0 be the uniform distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$ and D_1 be the distribution of $(\mathbf{A}, \mathbf{A} \cdot \mathbf{x}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$, where $\mathbf{A}$ is uniformly random in $\mathbb{Z}_q^{n \times m}$ and $\mathbf{x}$ is sampled from $\mathcal{X}$. Then*

$$\Delta(D_0, D_1) \leq \frac{1}{2}\sqrt{\sum_{i=1}^{k} p^{i \cdot n} \cdot \mathrm{Pr}_i},$$

where Pr_i is the collision probability of two independent samples from $\mathcal{X} \mod p^i$.

2.5 Fully Homomorphic Encryption

In this paper, we focus on secret key fully homomorphic encryption for efficiency reasons, meaning that the same key is required for encryption and decryption. This is sufficient for many applications, but our results can be easily generalized to the public key setting if required, by adding an encryption key which consists of a set of encryptions of zero. The main downside becomes storing the public key, as is usually the case in lattice-based cryptography.

Definition 1 (Fully Homomorphic Encryption). *A fully homomorphic encryption scheme is given by four polynomial time algorithms,* $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Eval})$ *described as follow:*

- $\mathsf{KeyGen}(1^\lambda)$ *on input a security parameter λ outputs an evaluation key* EVK *and a secret key* SK*;*
- $\mathsf{Encrypt}(\mathrm{SK}, \mu)$ *on input a secret key* SK *and a message μ returns a ciphertext* CT*;*

$\mathsf{Decrypt}(\mathrm{SK}, \mathrm{CT})$ *on input a secret key* SK *and a ciphertext* CT *returns a message* μ*;*
$\mathsf{Eval}(\mathrm{EVK}, f, \mathrm{CT}_1, \ldots, \mathrm{CT}_t)$ *on input an evaluation key* EVK*, a function* f *on* t *inputs, and* t *ciphertexts* $\mathrm{CT}_1, \ldots, \mathrm{CT}_t$ *returns a ciphertext* CT_f*.*

Let us denote $\mathcal{M}$ the message space and $\mathcal{C}$ the ciphertext space. For $\mu \in \mathcal{M}$, we define $\mathcal{C}_\mu = \mathsf{Decrypt}(\mathrm{SK}, \cdot)^{-1}(\mu)$, the set of all ciphertexts that decrypt to μ.

We say that an FHE scheme is *correct* if, for $(\mathrm{EVK}, \mathrm{SK})$ sampled from $\mathsf{KeyGen}(1^\lambda)$:

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

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

Definition 2 (Circuit Privacy). *A (fully) homomorphic encryption scheme is circuit-private for a class of functions* $\mathcal{F}$ *if the homomorphic evaluation of a function* $f \in \mathcal{F}$ *on encrypted messages does not leak more information than the evaluation result, even given the secret key, i.e. there exists a polynomial time simulator* Sim *such that the following property holds for any* $f \in \mathcal{F}$*:*

$$\left(\mathsf{Sim}\left(1^\lambda, f(\mu_1, \cdots, \mu_t), (\mathrm{CT}_1, \cdots, \mathrm{CT}_t), \mathrm{EVK}\right), (\mathrm{CT}_1, \cdots, \mathrm{CT}_t), \mathrm{SK}\right)$$

$$\approx_s \left(\mathsf{Eval}\left(\mathrm{EVK}, f, (\mathrm{CT}_1, \cdots, \mathrm{CT}_t)\right), (\mathrm{CT}_1, \cdots, \mathrm{CT}_t), \mathrm{SK}\right),$$

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

Remark 1. We point out that we are only dealing with honest-but-curious adversaries. This is captured by the fact that the ciphertexts and keys are sampled correctly. To prevent attacks in the malicious setting, one can use the techniques of [OPP14] to upgrade our scheme, using a maliciously circuit-private (possibly non-compact) FHE scheme. In comparison, in the following definition of sanitization, we still assume that the keys are sampled correctly, but privacy holds for any ciphertext, even maliciously generated ones.

Sanitization of ciphertexts. Intuitively, the goal of a sanitization algorithm is to remove all information conveyed in the ciphertext beside the message, that is, we want to erase its memory from any previous evaluations. More formally, we ask that a sanitization algorithm verifies two properties: it must be message-space preserving and sanitizing. These properties are defined as follow:

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

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

$$\left(\mathsf{Sim}(1^\lambda, \mu, \mathrm{EVK}), \mathrm{SK}\right) \approx_s \left(\mathsf{Sanitize}(\mathrm{EVK}, \mathrm{CT}), \mathrm{SK}\right),$$

where EVK is sampled honestly.

Remark 2. Note that given a sanitizing and message-space preserving $\mathsf{Sanitize}$ algorithm, it is easy to construct an FHE scheme that is circuit-private for all functions, by running $\mathsf{Sanitize}$ at the end of the evaluation procedure.

2.6 Background on TFHE

The TFHE encryption scheme [CGGI20] encrypts messages in a subset $\mathcal{M}$ of $\mathbb{T}$, using LWE. For simplicity, we will take $\mathcal{M} = \{0, \frac{1}{2}\}$ in the following.

A LWE encryption of a message $\mu \in \mathcal{M}$ is a vector $(\mathbf{a} \mid \mathbf{a} \cdot \mathbf{s} + \mu + e) \in \hat{\mathbb{Z}}_q^{1\times(n+1)}$ where $\mathbf{a}$ is uniform over $\hat{\mathbb{Z}}_q^{1\times n}$, e is sampled from χ_ϑ — a noise distribution with variance ϑ. The secret key $\mathbf{s}$ is sampled from a uniform distribution over $\mathbb{B}^n$. In order to decrypt $(\mathbf{a} \mid b)$, one computes an approximation of μ defined as the phase of ciphertext $(\mathbf{a} \mid b)$, $\varphi_{\mathbf{s}}(\mathbf{a} \mid b) = b - \mathbf{a} \cdot \mathbf{s}$, and rounds it to nearest element in $\mathcal{M}$.

The first step of the bootstrapping is to deterministically map ciphertexts from $\hat{\mathbb{Z}}_q^{1\times(n+1)}$ to $\mathbb{Z}_{2N}^{n+1}$. This incurs some rounding errors that could change the result of decryption on ill-formed ciphertexts. We thus take this additional rounding step into account when defining $\mathcal{C}_\mu$ in order to capture sanitization and message-space preserving even for maliciously generated ciphertexts. If parameters are set carefully, this has no impact on ciphertexts obtained in an honest way. For $\mu \in \{0, \frac{1}{2}\}$, we say that

$$\mathcal{C}_\mu = \left\{ (\mathbf{a} \mid b) \in \hat{\mathbb{Z}}_q^{1\times(n+1)} \;\middle|\; \left| \lfloor 2Nb \rceil - \sum_{i=1}^{n} s_i \lfloor 2Na_i \rceil - \mu \right| \leq \frac{1}{4} \right\}.$$

The error of a ciphertext $\mathbf{c} \in \mathcal{C}_\mu$ is defined as $\varphi_{\mathbf{s}}(\mathbf{c}) - \mu$, and its variance $\mathsf{Var}(\mathbf{c})$ as the variance of the error of $\mathbf{c}$ or equivalently the variance of $\varphi_{\mathbf{s}}(\mathbf{c})$.

The semantic security of this encryption scheme relies on the decision LWE problem [Reg05], or equivalently, its search counterpart:

- The decision LWE problem, parametrized by n and χ_ϑ, asks to distinguish the uniform distribution over $\hat{\mathbb{Z}}_q^{1\times(n+1)}$ from the distribution of fresh LWE encryptions of 0 using the same secret key $\mathbf{s}$ sampled uniformly at random in $\mathbb{B}^n$.
- The search LWE problem asks to find $\mathbf{s}$ from polynomially many fresh LWE encryptions of 0 using the same secret key $\mathbf{s}$ sampled uniformly at random in $\mathbb{B}^n$.

Our sanitization algorithm is based on the bootstrapping procedure of TFHE, which is an optimized version of FHEW bootstrapping [DM15]. The idea behind FHEW-based bootstrapping is to compute a ciphertext of $X^{\varphi_{\mathbf{s}}(\mathbf{c})}$. In order to achieve this, polynomials are encrypted using two distinct schemes: TLWE — a variant of LWE using ring elements — and TGSW.
A TLWE encryption of a message $\mu \in \hat{R}_q$ is a vector $(\mathbf{a} \mid \mathbf{a} \cdot \tilde{\mathbf{s}} + \mu + e) \in \hat{R}_q{}^{1\times(d+1)}$, where $\mathbf{a}$ is uniformly random in $\hat{R}_q{}^{d}$, e has coefficients sampled from χ_ϑ — ϑ is the noise variance. The secret key $\tilde{\mathbf{s}}$ is sampled uniformly at random over $\hat{R}_q{}^{d}$. We note $\mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu)$ the set of random variables $(\mathbf{a} \mid b)$ in $\hat{R}_q{}^{1\times(d+1)}$ whose phases $\varphi_{\tilde{\mathbf{s}}}(\mathbf{a} \mid b) = b - \mathbf{a} \cdot \tilde{\mathbf{s}}$ of center μ, and $\mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta}(\mu)$ a fresh TLWE encryption of μ under secret key $\tilde{\mathbf{s}}$ with noise variance ϑ. For $\mathbf{c} \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu)$, we define its error $\mathsf{Err}(\mathbf{c}) = \varphi_{\tilde{\mathbf{s}}}(\mathbf{c}) - \mu$, and we note $\mathsf{Var}(\mathbf{c})$ the covariance matrix of its coefficients. The semantic security of this encryption scheme relies on the ring variant of LWE [LPR10].

A TGSW encryption of a message $\mu \in R$ is a matrix $\mathbf{Z} + \mu\mathbf{G}$ in $\hat{R}_q{}^{(d+1)\cdot\ell\times(d+1)}$, where each of the $(d+1)\cdot\ell$ rows of $\mathbf{Z}$ is a TLWE encryption of 0. The secret key is the same as for TLWE and the matrix $\mathbf{G}$ is defined as in Section 2.1. We note $\mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta}(\mu)$ a fresh TGSW encryption of μ under secret key $\tilde{\mathbf{s}}$ with noise variance ϑ. More visually, we have:

$$\mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta}(\mu) = \begin{pmatrix} \mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta}(0) \\ \mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta}(0) \\ \vdots \\ \mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta}(0) \end{pmatrix} + \mu\mathbf{G} = (\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}) + \mu\mathbf{G}$$

where $\mathbf{A}$ is sampled uniformly at random in $\hat{R}_q^{(d+1)\cdot\ell\times d}$, and each coefficient appearing in $\mathbf{e}$ is sampled from χ_ϑ. The semantic security of this encryption scheme follows from the semantic security of TLWE.

Deterministic decomposition and external product: Any $\mathbf{v} \in \hat{R}_q^{1\times(d+1)}$ can be uniquely and deterministically decomposed into a small vector $\mathbf{G}^{-1}(\mathbf{v}) \in R^{1\times(d+1)\cdot\ell}$ whose coefficients are integers in $[-B/2, B/2[$ and such that $\mathbf{G}^{-1}(\mathbf{v}) \cdot \mathbf{G} = \mathbf{v}$. The full details are provided in supplementary materials Section B.1. This allows to multiply a TGSW ciphertext $\mathbf{C}$ and a TLWE ciphertext $\mathbf{c}$:

$$\mathbf{C} \boxdot \mathbf{c} = \mathbf{G}^{-1}(\mathbf{c}) \cdot \mathbf{C}$$

This external product allows homomorphic evaluations of multiplexers with convenient noise growth $\mathbf{c}_0 + \mathbf{C} \boxdot (\mathbf{c}_1 - \mathbf{c}_0)$ for a TGSW encryption $\mathbf{C}$ of the selector bit, and TLWE encryptions of the two inputs $\mathbf{c}_0$ and $\mathbf{c}_1$.

TFHE Bootstrapping: The bootstrapping procedure is a common tool to all known FHE schemes which allows to manage the noise growth during homomorphic evaluations. More formally, the bootstrapping takes as input an LWE ciphertext of some message μ and a bootstrapping key BK and outputs an LWE ciphertext of the same message μ, whose noise is below some bound controlled via the choice of parameters. The TFHE bootstrapping procedure can be decomposed in four steps, that are illustrated in Figure 1. The four steps are described below, with their type signatures explicitely defined.

- $\mathsf{Initialization}$: n-$\mathsf{LWE} \to \mathbb{Z}_{2N}^{n+1} \times \hat{R}_q$.
  Given an LWE ciphertext $(\mathbf{a} \mid b) \in \hat{\mathbb{Z}}_q^{n+1}$ as input, we define $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a_i} = \lfloor 2Na_i \rceil \in \mathbb{Z}_{2N}$ for each $i \in [n]$. Note that the cumulated error induced by rounding over $\mathbb{Z}_{2N}$ is taken into account in the definition of $\mathcal{C}_\mu$. The first step also intializes a testvector, $\mathsf{testv} \stackrel{def}{=} v(X) \in \hat{R}_q$, $v(X) = \sum_{i=0}^{N-1} v_i X^i$ and defines a noiseless RLWE encryption $(\mathbf{0}, \mathsf{testv})$. The test vector coefficients are chosen such that, after the $\mathsf{BlindRotate}$, one can retrieve an LWE ciphertext of the input message as the constant coefficient of the RLWE ciphertext.
- $\mathsf{BlindRotate}$: $\mathbb{Z}_{2N}^{n+1} \times \hat{R}_q \times \mathsf{TGSW}^n \to \mathsf{TLWE}$
  Given $\mathsf{testv} \in \hat{R}_q$, n coefficients $(\bar{a}_1, \cdots, \bar{a}_n, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$, and a bootstrapping key $\mathrm{BK}_i = \mathsf{TGSW}_{\tilde{s},\vartheta_{\mathrm{BK}}}(s_i)$ for $i \in [n]$, the $\mathsf{BlindRotate}$ algorithm returns a TLWE encryption of $\mathsf{testv} \cdot X^{\bar{\varphi}}$ where $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} s_i \bar{a}_i$.
- $\mathsf{Extract}$: $\mathsf{TLWE} \to dN$-LWE
  By interpreting a TLWE ciphertext of message $\mu \in \hat{R}_q$ under $\tilde{\mathbf{s}}$ as a vector of $d+1$ coefficients and taking into account the negacyclicity of the test vector polynomial i.e. $v_{i+N} = -v_i$, $\mathsf{Extract}$ extracts an LWE ciphertext under key $K = \mathsf{KeyExtract}(\tilde{\mathbf{s}})$ of dimension dN of the constant term of μ, $\mu(0)$.
- $\mathsf{KeySwitch}$: dN-$\mathsf{LWE} \times (n\text{-}\mathsf{LWE})^{dNt} \to n$-$\mathsf{LWE}$
  Given a dN-LWE ciphertext containing a message μ under key K, and a keyswitching key which consists of n-LWE encryptions of the bits of key K multiplied by the first t powers of $\frac{1}{\mathsf{B}_{\mathrm{KS}}}$ under secret key $\mathbf{s} \in \mathbb{B}^n$, $\mathsf{KeySwitch}$ outputs an n-LWE ciphertext of the same message μ under secret key $\mathbf{s}$.

The security of TFHE thus relies on both the decision LWE problem as well as the decision TLWE problem, but also on a circular assumption that states having LWE and TLWE encryptions of each other's secret key has no impact on security.

3 Randomization of TLWE ciphertexts

In order to achieve sanitization of ciphertexts, we make use of randomized decomposition, and public key encryptions of zero. In this section, we review their definitions and properties, we show how to construct them and analyze their impact on correctness.

3.1 Randomized decomposition

Notice that since the gadget matrix $\mathbf{G}$ only has constant coefficients, the product of $\mathbf{G}$ with a vector of polynomials $\mathbf{u} \in R^{(d+1)\cdot\ell}$ operates independently on each coefficient of $\mathbf{u}$. We can thus extend the sampling function from Lemma 1 to vectors of polynomials.

Definition 3 (Randomized Decomposition). *We define the randomized decomposition function* $\mathbf{G}_r^{-1}(\cdot) : \hat{R}_q^{1\times(d+1)} \to R^{1\times(d+1)\cdot\ell}$ *by using* N *copies of the randomized decomposition over* $\hat{\mathbb{Z}}_q^{d+1}$ *given by Lemma 1 with parameter* r*, i.e., we decompose each coefficient independently.*
We also define the randomized external product $\boxdot_r$*, which is defined as the external product* $\boxdot$ *from Section 2.6, replacing* $\mathbf{G}^{-1}(\cdot)$ *by the randomized decomposition* $\mathbf{G}_r^{-1}(\cdot)$*.*

We give the full details of the randomized decomposition in supplementary materials Section B.2. The following lemmata gives bounds on the noise propagation when using the randomized external product, which is an adaptation of [CGGI20, Corollary 3.14].

Lemma 8 (Variance of randomized External Product). *For any* $\mu \in \hat{R}_q$*,* $\mathbf{c} \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu)$ *with* $\|\mathsf{Var}(\mathbf{c})\|_2 = \vartheta_{\mathbf{c}}$*, and* $\mathbf{C} = \mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathbf{C}}}(0)$*. Then* $\mathbf{C} \boxdot_r \mathbf{c} \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(0)$ *and*

$$\|\mathsf{Var}\left(\mathbf{C} \boxdot_r \mathbf{c}\right)\|_2 \leq r^2(d+1)\ell N\vartheta_{\mathbf{C}}$$

except with probability at most $2^{-2(d+1)\cdot\ell\cdot N}$*.*

Proof. Let $\mathbf{C} = (\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e})$, with $\mathbf{A} \in \hat{R}_q^{(d+1)\cdot\ell\times d}$ and $\mathbf{e} \in \hat{R}_q^{(d+1)\cdot\ell}$. We have

$$\begin{aligned}\mathbf{C} \boxdot_r \mathbf{c} &= \mathbf{G}_r^{-1}(\mathbf{c}) \cdot \mathbf{C} \\ &= \left(\mathbf{G}_r^{-1}(\mathbf{c}) \cdot \mathbf{A} \;\middle|\; \mathbf{G}_r^{-1}(\mathbf{c}) \cdot \mathbf{A}\tilde{\mathbf{s}} + \mathbf{G}_r^{-1}(\mathbf{c}) \cdot \mathbf{e}\right)\end{aligned}$$

So $\mathsf{Err}\left(\mathbf{C} \boxdot_r \mathbf{c}\right) = \mathbf{G}_r^{-1}(\mathbf{c}) \cdot \mathbf{e}$. Using Lemma 3, the euclidean norm of $\mathbf{G}_r^{-1}(\mathbf{c})$ is bounded by $r \cdot \sqrt{(d+1)\cdot\ell\cdot N}$ except with probability at most $2^{-2(d+1)\cdot\ell\cdot N}$. Since all coefficients of $\mathbf{e}$ are independent, we conclude using properties on covariance matrices. □

Lemma 9 (Variance of randomized Multiplexer). *For any* $\beta, \mu_0, \mu_1 \in \hat{R}_q$*,* $\mathbf{c}_0 \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu_0)$ *with* $\|\mathsf{Var}(\mathbf{c}_0)\|_2 = \vartheta_{\mathbf{c}_0}$*,* $\mathbf{c}_1 \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu_1)$ *with* $\|\mathsf{Var}(\mathbf{c}_1)\|_2 = \vartheta_{\mathbf{c}_1}$*, and* $\mathbf{C} = \mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathbf{C}}}(\beta)$*. Then* $\mathbf{c}_0 + \mathbf{C} \boxdot_r (\mathbf{c}_1 - \mathbf{c}_0) \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mu_\beta)$ *and*

$$\|\mathsf{Var}\left(\mathbf{c}_0 + \mathbf{C} \boxdot_r (\mathbf{c}_1 - \mathbf{c}_0)\right)\|_2 \leq \vartheta_{\mathbf{c}_\beta} + r^2(d+1)\ell N\vartheta_{\mathbf{C}}$$

except with probability at most $2^{-2(d+1)\cdot\ell\cdot N}$*.*

Proof. Let $\mathbf{C} = \mathbf{Z} + \beta\mathbf{G}$, with $\mathbf{Z} = \mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathbf{C}}}(0)$. We have

$$\begin{aligned}\mathbf{c}_0 + \mathbf{C} \boxdot_r (\mathbf{c}_1 - \mathbf{c}_0) &= \mathbf{c}_0 + \mathbf{G}_r^{-1}(\mathbf{c}_1 - \mathbf{c}_0) \cdot (\mathbf{Z} + \beta\mathbf{G}) \\ &= \mathbf{c}_0 + \mathbf{G}_r^{-1}(\mathbf{c}_1 - \mathbf{c}_0) \cdot \mathbf{Z} + \beta\,(\mathbf{c}_1 - \mathbf{c}_0) \\ &= \mathbf{c}_1\beta + \mathbf{Z} \boxdot_r (\mathbf{c}_1 - \mathbf{c}_0)\end{aligned}$$

We conclude using Lemma 8. □

3.2 Public key encryptions of zero

To be able to generate TLWE encryptions of zero without access to the secret key, some of them are generated during the setup phase and are appended to the evaluation key as a sanitization key PK. Taking a subset sum of those allows us to reuse them everytime we need a fresh TLWE encryption of zero with a nice distribution. More formally, here are the requirements for our construction, as well as an instantiation given in Construction 1 satisfying those.

Lemma 10 (Sanitization key). *There exist two algorithms* PkGen *and* PkEnc *such that:*

PkGen$(1^\lambda, \tilde{\mathbf{s}})$*: on input a security parameter* λ*, and a* TLWE *secret key* $\tilde{\mathbf{s}} \in \hat{R_q}^d$*,* PkGen *outputs a public key* PK*.*

PkEnc(PK)*: for any* TLWE *secret key* $\tilde{\mathbf{s}}$*, on input a public key* PK *the output of* PkEnc *satisfies the two conditions:*

- $(\text{PK}, \mathsf{PkEnc}(\text{PK})) \approx_s (\text{PK}, (\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}} + e_u))$ *where* $\mathbf{u}$ *is uniform over* $\hat{R_q}^{1\times d}$*, and* PK *is honestly generated from* $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$*.*
- $\Delta\left(X^j \cdot \mathsf{PkEnc}(\text{PK}), \mathsf{PkEnc}(\text{PK})\right) = 0$*, for any* $j \in \mathbb{Z}_{2N}$*, and any* PK*.*

The first condition is required in the proof of Lemma 12. The distribution of e_u is not important for the sanization property but it has to remain small for correctness. The second condition is used in the proof of Lemma 16 and could be relaxed to be negligible, but we keep it 0 to simplify the proofs since our construction reaches the condition.

Remark 3. By adding a message to the $(d+1)$-th coordinate of $\mathsf{PkEnc}(\text{PK})$, we get a public key encryption scheme for polynomials with the nice following property: it allows packing N messages into one ciphertext; rotating the messages around the different slots by multiplying by X does not change the distribution of the ciphertext, even to somebody knowing the secret key. We give an example of protocol between $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, which coupled with the Extract procedure works as follows:

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

We believe this simple construction is of independent interest and might be use in other contexts. For example, this idea of having a proxy that filters data and rerandomizes it can be found in Access Control Encryption [DHO16].

Construction 1 *We exhibit a concrete example of a construction satisfying the properties of Lemma 10:*

- $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, *on input security parameter* λ, *and secret key* $\tilde{\mathbf{s}}$ *generates* m *fresh encryptions of* 0 *with noise variance* ϑ_{PK} $\mathrm{PK}_i = \mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta_{\mathrm{PK}}}(0)$ *and outputs* $\mathrm{PK} = (\mathrm{PK}_i)_{i=1\dots m}$.
- $\mathsf{PkEnc}(\mathrm{PK}_i)$, *on input public key* $\mathrm{PK} = (\mathrm{PK}_i)_{i=1\dots m}$ *samples a uniformly random binary vector* $\mathbf{r} = (r_i)_{i=1\dots m} \in \{0,1\}^m$, *and a uniformly random vector* $\mathbf{j} = (j_i)_{i=1\dots m} \in \mathbb{Z}_{2N}^m$ *and outputs* $\sum_{i=1}^{m} r_i X^{j_i} \mathrm{PK}_i$.

Lemma 11 proves that our instantiations of PkGen and PkEnc, given in Construction 1, satisfy the property claimed in Lemma 10.

Lemma 11. *For any* TLWE *secret key* $\tilde{\mathbf{s}}$, *Construction 1 has the following properties:*

- *Let* $\varepsilon > 0$, *if* $m > ((d+1)N \log q - 2\log\varepsilon) - 1$, *we have:*

$$\Delta\left((\mathrm{PK}, \mathsf{PkEnc}(\mathrm{PK})), (\mathrm{PK}, (\mathbf{u} \mid \mathbf{u}\cdot\tilde{\mathbf{s}} + e_u))\right) \leq \varepsilon$$

  where $\mathbf{u}$ *is uniform over* $\hat{R}_q^{1\times d}$, *and* PK *is honestly generated from* $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$;
- $\Delta\left(X^j \cdot \mathsf{PkEnc}(\mathrm{PK}), \mathsf{PkEnc}(\mathrm{PK})\right) = 0$, *for any* $j \in \mathbb{Z}_{2N}$, *and any* PK.

Proof. First, notice that for any $j \in \mathbb{Z}_{2N}$, any PK, $\mathbf{j}$ uniformly sampled in $\mathbb{Z}_{2N}^m$, $\Delta\left(X^j X^{j_i}, X^{j_i}\right) = 0$ since j_i is uniform in $\mathbb{Z}_{2N}$.
Now, for any secret key $\tilde{\mathbf{s}}$, for any $\mathbf{j} \in \mathbb{Z}_{2N}^m$, and PK honestly generated from $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, if we write $X^{j_i}\mathrm{PK}_i = (\mathbf{a}_i \mid \mathbf{a}_i\tilde{\mathbf{s}} + e_i) \in \hat{R}_q^{1\times(d+1)}$, $\mathbf{A} = (a_i)_{i=1\dots m}$, and $\mathbf{e} = (e_i)_{i=1\dots m}$, we want to analyze the joint distribution of PK and

$$\sum_{i=1}^{m} r_i X^{j_i}\mathrm{PK}_i = \left(\sum_{i=1}^{m} r_i\mathbf{a}_i \;\middle|\; \left(\sum_{i=1}^{m} r_i\mathbf{a}_i\right)\tilde{\mathbf{s}} + \sum_{i=1}^{m} r_i e_i\right)$$

which in turn is given by the distribution of $\left(\mathbf{A}, \sum_{i=1}^m r_i\mathbf{a}_i, \sum_{i=1}^m r_i e_i, \mathbf{e}\right)$. Note that since we're only dealing with sums, we can view the vectors of polynomials $\mathbf{a}_i$ of $\hat{R}_q^{1\times d}$ as vectors of $\hat{\mathbb{Z}}_q^{dN}$. Thus, we want to show that for any $\mathbf{e} \in \hat{R}_q^{m}$,

$$\Delta\left(\left(\mathbf{A}, \sum_{i=1}^{m} r_i\mathbf{a}_i, \sum_{i=1}^{m} r_i e_i, \mathbf{e}\right), \left(\mathbf{A}, \mathbf{u}, \sum_{i=1}^{m} r_i e_i, \mathbf{e}\right)\right) \leq \varepsilon$$

where each $\mathbf{a}_i$ is sampled uniformly at random in $\hat{\mathbb{Z}}_q^{dN}$, each r_i is sampled uniformly at random in $\{0,1\}$, and $\mathbf{u}$ is uniformly random in $\hat{\mathbb{Z}}_q^{dN}$.
The min-entropy of $\mathbf{r}$ is $H_\infty(\mathbf{r}) = m$, so taking account the leakage of at most $N\log(q)$ bits, $H_\infty\left(\mathbf{r} \;\middle|\; \sum_{i=1}^{m} r_i e_i\right) \geq m - N\log(q)$. For any $i = 1\dots k$, the probability of collision modulo 2^i, $\Pr_i \leq 2^{-(m-N\log(q))} = \frac{q^N}{2^m}$. Note that $\sum_{i=1}^{k} 2^{i\cdot dN} \leq 2\cdot q^{dN}$, thus, using Lemma 7,

$$\Delta\left(\left(\mathbf{A}, \sum_{i=1}^{m} r_i\mathbf{a}_i, \sum_{i=1}^{m} r_i e_i, \mathbf{e}\right), \left(\mathbf{A}, \mathbf{u}, \sum_{i=1}^{m} r_i e_i, \mathbf{e}\right)\right) \leq \sqrt{\frac{q^{(d+1)N}}{2^{m+1}}} < \varepsilon$$

□

Impact on security: We note that appending this sanitization key PK to the evaluation key gives more power to an attacker. However, semantic security still relies on the decision LWE and decision variant over rings, together with the circular security assumption. We just assume more TLWE ciphertexts at hand.

3.3 Randomization of TLWE ciphertexts

Combining those tools gives a randomization procedure for TLWE ciphertexts, which also scales the plaintext by any value α for which we know a ciphertext, and which outputs a canonical distribution, statistically independent of α. Note that α doesn't have to be known. This is captured by the following lemma, where we analyze only the distribution for TLWE samples, but adding the plaintext and verifying the correctness is straightforward. Here, you can think of $\mathbf{v}$ as being a TLWE encryption of α.

Lemma 12. *For any* $\tilde{\mathbf{s}} \in \hat{R}_q^{\,d}$, *any* $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{\,(d+1)\cdot\ell}$, *and any* $\mathbf{v} \in \hat{R}_q^{\,d+1}$. *Let* $\mathbf{E}_i$ *be the matrix such that* $\mathtt{pow}_X \cdot \mathbf{E}_i = \mathtt{pow}_X \otimes e_i$, *then we have:*

$$(\mathbf{0}, y) + \mathsf{PkEnc}(\mathrm{PK}) + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{v} \approx_s \left(\mathbf{u} \;\middle|\; \mathbf{u} \cdot \tilde{\mathbf{s}} + e' + e_u\right),$$

where $\mathbf{A}$ *is uniform over* $\hat{R}_q^{\,(d+1)\cdot\ell\times d}$, $\mathbf{u}$ *is uniform over* $\hat{R}_q^{\,1\times d}$, $y \leftarrow \frac{1}{q}\mathcal{D}_{R,r}$, PK *is honestly generated from* $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, $e_u = \mathsf{Err}\left(\mathsf{PkEnc}(\mathrm{PK})\right)$, *and* $e' \leftarrow \mathcal{D}_{\frac{1}{q}R,\Gamma}$, *with*

$$\Gamma = r\sqrt{\frac{1}{q^2}\mathbf{I}_N + \sum_{i=1}^{(d+1)\cdot\ell} \mathbf{E}_i^t\mathbf{E}_i}.$$

The proof of this lemma is twofold, since we need to analyze both the distribution of the mask, which has to be uniform, and of the noise, which has to be a discrete Gaussian. Adding a public key encryption ensures that the mask is uniform, while the randomized decomposition ensures that the noise has a correct distribution.
Before going through the technical details of the noise analysis, let us give more intuition about our proof. Our construction ensures that the resulting noise is just a linear combination of discrete Gaussians. Now, if the parameters of the discrete Gaussians are big enough, they behave like continuous Gaussians, and any linear combination of them remains a Gaussian distribution. What is left is to find the threshold we need to hit with the parameters. Formally, this is given by the following result, which can be viewed as a generalization of [BdMW16, Lemma 3.6] over rings.

Lemma 13. *Let* $\varepsilon, r, r' > 0$. *For any* $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{\,(d+1)\cdot\ell}$ *and any* $\mathbf{c} \in \hat{R}_q^{\,(d+1)\cdot\ell\cdot N}$. *Let* $\mathbf{E}_i$ *be the matrix such that* $\mathtt{pow}_X \cdot \mathbf{E}_i = \mathtt{pow}_X \otimes e_i$, *then if* $\min(r, r') \geq \sqrt{1+B^2}(1 + q\left\|\mathbf{e}\right\|_2) \cdot \sqrt{\frac{\ln(2\cdot(d+1)\cdot\ell\cdot N\cdot(1+1/\varepsilon))}{\pi}}$, *we have*

$$\Delta\left(\mathbf{e}^t\mathbf{x} + y, e'\right) < 2\varepsilon$$

where $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r}$, $y \leftarrow \frac{1}{q}\mathcal{D}_{R,r'}$, *and* $e' \leftarrow \frac{1}{q}\mathcal{D}_{R,\Gamma}$, *with*

$$\Gamma = \sqrt{\frac{r'^2}{q^2}\mathbf{I}_N + r^2 \sum_{i=1}^{(d+1)\cdot\ell} \mathbf{E}_i^t\mathbf{E}_i}.$$

Proof. In order to be able to use known results on lattices, let us consider the naive coefficient embedding of the polynomials. That is, we consider $\mathbf{c} = (c_{1,0}, c_{1,1}, \ldots, c_{1,N-1}, c_{2,0}, \ldots, c_{(d+1)\cdot\ell,N-1}) \in \frac{1}{q}\mathbb{Z}^{(d+1)\cdot\ell\cdot N}$. We define the following notations in order to better explain the intuition behind the proof:

- $\widehat{\mathbf{E}} = \left(\mathbf{E}_0 \;\middle|\; \mathbf{E}_1 \;\middle|\; \ldots \;\middle|\; \mathbf{E}_{(d+1)\cdot\ell} \;\middle|\; \frac{1}{q}\mathbf{I}_N\right) \in \frac{1}{q}\mathbb{Z}^{N\times N((d+1)\cdot\ell+1)}$;
- $\beta = \begin{pmatrix} r\mathbf{I}_{(d+1)\cdot\ell\cdot N} & \mathbf{0} \\ \mathbf{0} & r'\mathbf{I}_N \end{pmatrix} \in \mathbb{Z}^{N((d+1)\cdot\ell+1)\times N((d+1)\cdot\ell+1)}$;
- $\widehat{\mathbf{c}} = (\mathbf{c}, 0, \ldots 0) \in \frac{1}{q}\mathbb{Z}^{N((d+1)\cdot\ell+1)}$;
- $\widehat{\Lambda} = \Lambda^{\perp}(\mathbf{G})^N \times \mathbb{Z}^N \subset \mathbb{Z}^{N((d+1)\cdot\ell+1)}$.

We want to show that:

$$\Delta\left(\widehat{\mathbf{E}}\,\mathcal{D}_{\widehat{\Lambda}+\widehat{\mathbf{c}},\beta}, \mathcal{D}_{\frac{1}{q}\mathbb{Z}^N,\sqrt{\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t}}\right) \leq 2\varepsilon$$

First, notice that the support of $\widehat{\mathbf{E}}\mathcal{D}_{\widehat{\Lambda}+\widehat{\mathbf{c}},\beta}$ is $\frac{1}{q}\mathbb{Z}^N$, thanks to the $\frac{1}{q}I_n$ bloc in the matrix $\widehat{\mathbf{E}}$. Now we have to analyze the probability mass assigned to each element of $\frac{1}{q}\mathbb{Z}^N$.

Fix some $\mathbf{z} \in \frac{1}{q}\mathbb{Z}^N$. The probability mass assigned to $\mathbf{z}$ by $\widehat{\mathbf{E}}\mathcal{D}_{\widehat{\Lambda}+\widehat{\mathbf{c}},\beta}$ is proportional to $\rho_\beta(\mathcal{L}_\mathbf{z})$, where $\mathcal{L}_\mathbf{z} = \left\{\mathbf{v} \in \widehat{\Lambda} + \widehat{\mathbf{c}} \mid \widehat{\mathbf{E}}\mathbf{v} = \mathbf{z}\right\}$. We define the lattice $\mathcal{L} = \left\{\mathbf{v} \in \widehat{\Lambda} \mid \widehat{\mathbf{E}}\mathbf{v} = \mathbf{0}\right\}$; note that $\mathcal{L}_\mathbf{z}$ is a coset of $\mathcal{L}$, so $\mathcal{L}_\mathbf{z} = \mathcal{L} + \mathbf{w}_\mathbf{z}$ for any $\mathbf{w}_\mathbf{z} \in \mathcal{L}_z$. Let $\mathbf{u}_\mathbf{z} = \beta\widehat{\mathbf{E}}^t\left(\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t\right)^{-1}\mathbf{z} \in \mathbb{Z}^N$. This is well defined because $\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t$ is a positive definite matrix.
For any $\mathbf{t} \in \beta^{-1}\mathcal{L}_\mathbf{z}$ we have $\widehat{\mathbf{E}}\beta\left(\mathbf{t} - \mathbf{u}_\mathbf{z}\right) = 0$, so $\mathbf{u}_\mathbf{z}$ is orthogonal to $\beta^{-1}\mathcal{L}_\mathbf{z} - \mathbf{u}_\mathbf{z}$.
From this we have $\rho(\mathbf{t}) = \rho(\mathbf{u}_\mathbf{z}) \cdot \rho(\mathbf{t} - \mathbf{u}_\mathbf{z})$, and by summing for $\mathbf{t} \in \beta^{-1}\mathcal{L}_\mathbf{z}$:

$$\rho(\beta^{-1}\mathcal{L}_\mathbf{z}) = \rho(\mathbf{u}_\mathbf{z}) \cdot \rho(\beta^{-1}\mathcal{L}_\mathbf{z} - \mathbf{u}_\mathbf{z})$$

Observe that $\mathcal{L}_\mathbf{z} - \beta\mathbf{u}_\mathbf{z} = \mathcal{L} + \mathbf{w}_\mathbf{z} - \beta\mathbf{u}_\mathbf{z}$ and $\widehat{\mathbf{E}}(\mathbf{w}_\mathbf{z} - \beta\mathbf{u}_\mathbf{z}) = 0$, so $\beta^{-1}\mathcal{L}_\mathbf{z} - \mathbf{u}_\mathbf{z} = \beta^{-1}(\mathcal{L} - \mathbf{c}')$ for some $\mathbf{c}'$ in the vector span of the lattice $\mathcal{L}$. Since $\sigma_{N((d+1)\cdot\ell+1)}(\beta) = \min(r, r') \geq \sqrt{1+B^2}(1 + q\left\|\mathbf{e}\right\|_2) \cdot \sqrt{\frac{\ln(2\cdot(d+1)\cdot\ell\cdot N\cdot(1+1/\varepsilon))}{\pi}} \geq \eta_\varepsilon(\mathcal{L}_z)$, by Lemma 14 we obtain

$$\begin{aligned}
\rho(\beta^{-1}\mathcal{L}_\mathbf{z}) &= \rho(\mathbf{u}_\mathbf{z}) \cdot \rho_\beta(\mathcal{L} - \mathbf{c}') \\
&\in \left[\frac{1-\varepsilon}{1+\varepsilon},\ 1\right] \cdot \rho_\beta(\mathcal{L}) \cdot \rho(\mathbf{u}_\mathbf{z}) \quad \text{by Lemma 5} \\
&= \left[\frac{1-\varepsilon}{1+\varepsilon},\ 1\right] \cdot \rho_\beta(\mathcal{L}) \cdot \rho\left(\beta\widehat{\mathbf{E}}^t\left(\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t\right)^{-1}\mathbf{z}\right) \\
&= \left[\frac{1-\varepsilon}{1+\varepsilon},\ 1\right] \cdot \rho_\beta(\mathcal{L}) \cdot \rho_{\sqrt{\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t}}(\mathbf{z}) \quad \text{by definition of } \rho
\end{aligned}$$

This implies that the statistical distance between $\widehat{\mathbf{E}}\,\mathcal{D}_{\widehat{\Lambda}+\widehat{\mathbf{c}},\beta}$ and $\mathcal{D}_{\frac{1}{q}\mathbb{Z}^N,\sqrt{\widehat{\mathbf{E}}\beta^2\widehat{\mathbf{E}}^t}}$ is at most $1 - \frac{1-\varepsilon}{1+\varepsilon} \leq 2\varepsilon$. □

In order to conclude the proof of Lemma 13, we also need a bound on the smoothing parameter of the lattice $\mathcal{L}$, which is derived following a result from [BdMW16, Lemma 3.7], but adapted to our case. This bound is given in the following lemma, for which the detailed proof is included in the supplementary material Section A for the sake of completeness.

Lemma 14. *Let $\varepsilon > 0$. For any $\mathbf{e} \in \hat{R}_q^{(d+1)\cdot\ell}$, let $\mathcal{L}$ be as defined in the proof of Lemma 13. Then we have:*

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{1+B^2}(1+q\|\mathbf{e}\|_2)\cdot\sqrt{\frac{\ln(2\cdot(d+1)\cdot\ell\cdot N\cdot(1+1/\varepsilon))}{\pi}}.$$

We are now ready to prove Lemma 12:

Proof. Note that for any $\tilde{\mathbf{s}} \in \hat{R}_q^{d}$, any $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{(d+1)\cdot\ell}$, any $\mathbf{v} \in \hat{R}_q^{d+1}$, and any $\mathbf{A} \in \hat{R}_q^{(d+1)\cdot\ell\times d}$, we have:

$$\begin{aligned}\left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}}+\mathbf{e}\right) \boxdot_r \mathbf{v} &= \mathbf{G}_r^{-1}(\mathbf{v})\cdot\left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}}+\mathbf{e}\right)\\ &= \left(\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A} \mid \mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A}\tilde{\mathbf{s}}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{e}\right)\end{aligned}$$

By Lemma 7, for $\mathbf{A}$ uniformly sampled in $\hat{R}_q^{(d+1)\cdot\ell\times d}$, $\mathbf{u}$ uniformly sampled in $\hat{R}_q^{1\times d}$, $y \leftarrow \frac{1}{q}\mathcal{D}_{R,r}$, PK honestly generated from $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, and $e_u = \mathsf{Err}(\mathsf{PkEnc}(\mathrm{PK}))$, thanks to Lemma 10 we have:

$$\begin{aligned}&(\mathbf{0}, y) + \mathsf{PkEnc}(\mathrm{PK}) + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}}+\mathbf{e}\right) \boxdot_r \mathbf{v}\\ \approx_s\ &(\mathbf{u} \mid \mathbf{u}\cdot\tilde{\mathbf{s}} + y + e_u) + \left(\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A} \mid \mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A}\tilde{\mathbf{s}}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{e}\right)\\ =\ &\left(\mathbf{u}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A} \mid \left(\mathbf{u}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A}\right)\tilde{\mathbf{s}}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{e}+y+e_u\right)\\ =\ &\left(\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{e}+y+e_u\right)\end{aligned}$$

Because $\Delta\left(\mathbf{u}+\mathbf{G}_r^{-1}(\mathbf{v})\cdot\mathbf{A}, \mathbf{u}\right) = 0$, i.e., they both are uniform random variable. Now we conclude using Lemma 13. □

4 New sanitization algorithm

In this section, we first present our circuit private version of the BlindRotate algorithm, which will be used as a building block for our sanitization procedure. We then introduce the new Sanitize procedure and show it verifies the desired properties.

4.1 Circuit Private blind rotation

In order to achieve circuit privacy of the BlindRotate algorithm, the main loop of TFHE BlindRotate (line 3 of Algorithm 1) (which is a multiplexer operation) is replaced by its randomized counterpart. We also add a Gaussian noise when initializing the accumulator.

Lemma 15 (CP-BlindRotate noise propagation). *Let $\mathrm{BK} = (\mathrm{BK}_i)_i$ be a bootstrapping key where $\mathrm{BK}_i = \mathsf{TGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathrm{BK}}}(s_i)$ for $i \in [1,n]$, and $\mathrm{PK} = (\mathrm{PK}_i)_i$ be a sanitization key where $\mathrm{PK}_i = \mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta_{\mathrm{PK}}}(0)$ for $i \in [1,m]$. For any $\mathsf{testv} \in \hat{R}_q$, any $(\bar{\mathbf{a}}, \bar{b}) \in \mathbb{Z}_{2N}^{n+1}$, and $\mathbf{c} = \mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\bar{\mathbf{a}}, \bar{b}), \mathrm{BK}, \mathrm{PK}\right)$, we have that $\mathbf{c} \in \mathsf{TLWE}_{\tilde{\mathbf{s}}}(\mathsf{testv}\cdot X^{\bar{\varphi}})$ where $\bar{\varphi} = \bar{b} - \sum_{i=1}^n s_i\bar{a}_i$, and*

$$\|\mathsf{Var}(\mathbf{c})\|_2 \leq n\frac{r^2}{2\pi q^2} + n(d+1)\ell N r^2\vartheta_{\mathrm{BK}} + nm\vartheta_{\mathrm{PK}} \tag{2}$$

Proof. The bound comes from the fact that Algorithm 1 performs n homomorphic multiplexers, each adding at most $(d+1)\ell N r^2\vartheta_{\mathrm{BK}}$ to the variance of the accumulator using Lemma 9, adds n independent TLWE encryption $\mathsf{PkEnc}(\mathrm{PK})$ of variance bounded by $m\vartheta_{\mathrm{PK}}$, and n independent Gaussian noise y_i of variance $\frac{r^2}{2\pi q^2}$. □

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

Input: $(\bar{\mathbf{a}},\bar{b})\in\mathbb{Z}_{2N}^{n+1}$, a bootstrapping key $\mathrm{BK}=(\mathrm{BK}_i)_i$, where BK_i is a TGSW encryption of s_i for $i\in[1,n]$, a sanitization key PK , and two fixed messages $\mu_0=0,\mu_1=\frac{1}{2}$.

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

1: $\mathsf{ACC}=(0,\dots,0,\mathsf{testv}\cdot X^{-\bar{b}})\in\hat{R}_q^{\,d+1}$
2: **for** $i=1$ **to** n
3: $\boxed{\mathsf{ACC}+=\mathrm{BK}_i\boxdot_r((X^{\bar{a}_i}-1)\mathsf{ACC})+\mathsf{PkEnc}(\mathrm{PK})+(\mathbf{0},y_i)}\,,y_i\leftarrow\mathcal{D}_{\frac{1}{q}R,r}$
4: Return ACC

Lemma 16. *Let* $\mathrm{PK}=(\mathrm{PK}_i)_i$ *be a sanitization key where* $\mathrm{PK}_i=\mathsf{TLWE}_{\tilde{\mathbf{s}},\vartheta_{\mathrm{PK}}}(0)$ *for* $i\in[1,m]$. *For any* $\mathsf{testv}\in\hat{R}_q$, *the encryption scheme of Section 2.6 is circuit private for the class of functions* $\left(\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\bar{\mathbf{a}},\bar{b}),\cdot,\mathrm{PK}\right)\right)_{(\bar{\mathbf{a}},\bar{b})}$.

Intuitively, this means that no information is leaked by the output of CP-BlindRotate beside $\mathsf{testv}\cdot X^{\bar{\varphi}}$, even to someone who knows the secret key.

Proof. Before exhibiting our simulation algorithm, we first need to analyze the output of Algorithm 1. The difficulty in this analysis is the dependencies between all the arguments of $\mathbf{G}_r^{-1}$ and also the other terms, as well as the power of X by which the accumulator is multiplied at each step and that could leak information. In order to get around these issues, our proof uses two rounds of induction following the iterations of step 3 of Algorithm 1: a first round to sort out the powers of X, and the second one in order to deal with the random variables corresponding to the states of the accumulator one at a time. After iteration $t, t>0$, we have:

$$\begin{aligned}\mathsf{ACC}_t=&\mathsf{ACC}_{t-1}+\mathrm{BK}_t\boxdot_r\left(\left(X^{\bar{a}_t}-1\right)\mathsf{ACC}_{t-1}\right)+\mathsf{PkEnc}(\mathrm{PK})+(\mathbf{0},y_t)\\=&\mathsf{ACC}_{t-1}+(\mathrm{BK}_t-s_t\mathbf{G})\boxdot_r\left(\left(X^{\bar{a}_t}-1\right)\mathsf{ACC}_{t-1}\right)+\mathsf{PkEnc}(\mathrm{PK})+(\mathbf{0},y_t)\\&+s_t\mathbf{G}\boxdot_r\left(\left(X^{\bar{a}_t}-1\right)\mathsf{ACC}_{t-1}\right)\\=&(\mathrm{BK}_t-s_t\mathbf{G})\boxdot_r\left(\left(X^{\bar{a}_t}-1\right)\mathsf{ACC}_{t-1}\right)+\mathsf{PkEnc}(\mathrm{PK})+(\mathbf{0},y_t)\\&+\left(1+\left(X^{\bar{a}_t}-1\right)s_t\right)\mathsf{ACC}_{t-1}\end{aligned}$$

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

$$\mathsf{ACC}_t=X^{\bar{a}_t s_t}\mathsf{ACC}_{t-1}+(\mathrm{BK}_t-s_t\mathbf{G})\boxdot_r\left(\left(X^{\bar{a}_t}-1\right)\mathsf{ACC}_{t-1}\right)+\mathsf{PkEnc}(\mathrm{PK})+(\mathbf{0},y_t)$$

To make the claim more readable, we define the following shorthand notations for given $\bar{\mathbf{a}}$, $\mathbf{s}$, t and j, $X^{\leq}$ (resp. $X^{>}$) for X raised to the power $\sum_{i=1}^{t}s_i\bar{a}_i$ (resp. $\sum_{i=j+1}^{t}s_i\bar{a}_i$). Given that $\mathsf{ACC}_0=\left(\mathbf{0},\mathsf{testv}\cdot X^{-\bar{b}}\right)$, a simple induction shows that,

$$\begin{aligned}\mathsf{ACC}_t=&\left(\mathbf{0},\left(\mathsf{testv}\cdot X^{-\bar{b}}\right)\cdot X^{\leq}\right)+\sum_{j=1}^{t}X^{>}\mathsf{PkEnc}(\mathrm{PK})+\sum_{j=1}^{t}X^{>}(\mathbf{0},y_j)\\&+\sum_{j=1}^{t}X^{>}(\mathrm{BK}_j-s_j\mathbf{G})\boxdot_r\left(\left(X^{\bar{a}_j}-1\right)\mathsf{ACC}_{j-1}\right)\end{aligned}$$

Now, remark that as probability distributions,

$$\left(X^{\leq} y_t, X^{>}\mathbf{G}_r^{-1}\left(\left(X^{\bar{a}_j}-1\right)\mathsf{ACC}_{j-1}\right)\right) = \left(y_t, \mathbf{G}_r^{-1}\left(\left(X^{\bar{a}_j}-1\right)\mathsf{ACC}_{j-1}\right)\right),$$

since y_t and $\mathbf{G}_r^{-1}\left(\left(X^{\bar{a}_j}-1\right)\mathsf{ACC}_{j-1}\right)$ are drawn from spherical Gaussian distributions, and for any $v \in \frac{1}{q}R$, $i \in \mathbb{Z}_{2N}$, $\left\|X^i \cdot v\right\|_2 = \|v\|_2$, because they have the same coefficients modulo $X^{2N}-1$ in absolute value. We also have $\Delta\left(X^{>}\mathsf{PkEnc}(\mathrm{PK}), \mathsf{PkEnc}(\mathrm{PK})\right) = 0$ by Lemma 10, so

$$\begin{aligned}\mathsf{ACC}_t = &\left(\mathbf{0}, \mathsf{testv}\cdot X^{-\bar{b}}\cdot X^{\leq}\right)\\ &+\sum_{j=1}^{t}(\mathbf{0}, y_j) + \mathsf{PkEnc}(\mathrm{PK}) + \left(\mathrm{BK}_j - s_j\mathbf{G}\right)\boxdot_r\left(\left(X^{\bar{a}_j}-1\right)\mathsf{ACC}_{j-1}\right)\end{aligned}$$

Following a simple induction, using twice Lemma 12 at each step,[3]

$$\mathsf{ACC}_t = \left(\mathbf{0}, \mathsf{testv}\cdot X^{-\bar{b}}\cdot X^{\leq}\right) + \sum_{j=1}^{t}(\mathbf{0}, y_j) + \mathsf{PkEnc}(\mathrm{PK}) + \left(\mathrm{BK}_j - s_j\mathbf{G}\right)\boxdot_r \mathbf{0}$$

When $t = n$, $X^{-\bar{b}}\cdot X^{\leq} = X^{\bar{\varphi}}$ and we have:

$$\mathsf{ACC}_n = \left(\mathbf{0}, \mathsf{testv}\cdot X^{\bar{\varphi}}\right) + \sum_{t=1}^{n}(\mathbf{0}, y_t) + \mathsf{PkEnc}(\mathrm{PK}) + \left(\mathrm{BK}_t - s_t\mathbf{G}\right)\boxdot_r \mathbf{0}$$

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

$$\begin{aligned}\mathsf{Sim}\left(\mathsf{testv}\cdot X^{\bar{\varphi}}, \mathrm{BK}, \mathrm{PK}, \mathsf{testv}\right) =&\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\mathbf{0}, 0), \mathrm{BK}, \mathrm{PK}\right)\\ &+\left(\mathbf{0}, \mathsf{testv}\cdot\left(X^{\bar{\varphi}}-1\right)\right).\end{aligned}$$

□

4.2 Sanitizing algorithm

We are now ready to present our sanitization algorithm. It is basically the $\mathsf{Bootstrap}$ algorithm from TFHE, with the testvector testv chosen to evaluate the identity function, and where we replaced $\mathsf{BlindRotate}$ with $\mathsf{CP\text{-}BlindRotate}$. It is described more formally in Algorithm 2.

Lemma 17 (Sanitization noise propagation). *If the parameters are set correctly, Algorithm 2 is message-space preserving. In more details, we have that the variance of its output has a norm bounded by*

$$\frac{r^2}{2\pi q^2} + nr^2(d+1)\ell N\vartheta_{\mathrm{BK}} + n\cdot m\vartheta_{\mathrm{PK}} + d\cdot N\left(\frac{\mathsf{B}_{\mathrm{KS}}^{-2t}}{4} + t\cdot\frac{\mathsf{B}_{\mathrm{KS}}^2}{4}\vartheta_{\mathrm{KS}}\right) \tag{3}$$

Proof. This results combines Lemma 15 and Lemma 20. □

[3] Here, it would seem that the $\mathsf{PkEnc}(\mathrm{PK})$ and y_t variables could be reused, because they appear both before and after applying our lemma. However, one has to be very careful because Lemma 12 can use any argument at the right hand side of $\boxdot_r$, but it has to be independent of the random variables that appear in the statement of the lemma, which is not the case if we want to reuse our randomness.

Algorithm 2 Sanitizing algorithm for LWE encryption $(\mathbf{a}, b)$ of μ

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

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

1: $\bar{\mu} = \frac{\mu_0 + \mu_1}{2}$ and $\bar{\mu}' = \mu_0 - \bar{\mu}$
2: $\mathsf{testv} = (1 + X + \cdots + X^{N-1}) \cdot X^{-\frac{N}{2}} \cdot \bar{\mu}'$
3: $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a_i} = \lfloor 2Na_i \rceil$
4: $\mathbf{c}' = \mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}} \left((\bar{\mathbf{a}}, \bar{b}), \text{BK}, \text{PK} \right)$
5: $\mathbf{c} = \left(\mathbf{0}, \frac{1}{4} \right) + \mathsf{Extract}(\mathbf{c}')$
6: Return $\mathsf{KeySwitch}(\text{KS}, \mathbf{c})$

Lemma 18. *Algorithm 2 is a sanitizing algorithm for encryption scheme of Section 2.6.*

Proof. For the sake of readability, we omit the public parameters and keys as arguments to the simulators. The simulator Sim_s of the sanitization game, when called on a message μ, follows the same steps as the Sanitize algorithm except it calls the simulator Sim_{cp} of the circuit privacy game instead of calling CP-BlindRotate, with $\bar{\varphi} = \mu$. Indeed, the output of Sim_{cp} is indistinguishable from the one of CP-BlindRotate. However, we have to be careful, because we don't know the correct phase $\bar{\varphi}$ to invoke Sim_{cp}.
To better understand why our statement stands, we need to further analyze Extract which is recalled in details in the supplementary materials, Section B.3 for the sake of completeness.
On input a TLWE ciphertext $(\mathbf{a} \mid b)$, Extract trims the polynomial b to only its first coefficient $b(0)$, and rearranges the coefficients of $\mathbf{a}(X^{-1})$ into a vector. Recall that the output of $\mathsf{Sim}_{cp}(\bar{\varphi})$ has the following distribution:

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

So, as long as $\mathsf{testv} \cdot X^{\bar{\varphi}} = \mathsf{testv} \cdot X^{\mu}$, $\Delta \left(\mathsf{Extract} \left(\mathsf{Sim}_{cp}(\bar{\varphi}) \right), \mathsf{Extract} \left(\mathsf{Sim}_{cp}(\mu) \right) \right) = 0$. This is the case when $|\lfloor 2Nb \rceil - \sum_{i=1}^{n} s_i \lfloor 2Na_i \rceil - \mu| \leq \frac{1}{4}$, i.e. $(\mathbf{a}, b) \in \mathcal{C}_\mu$. □

5 Parameters, security and experimental results

In this section, we proceed to the selection of parameters for the CP-BlindRotate given in Algorithm 1.

Circuit Privacy: setting $q = 2^{45}$, $n = 612$, $N = 2048$, $B = 512$, $d = 1$, $\ell = 5$:

- lemma 13 gives a lower bound on r' and r, where r is the parameter of the randomized gadget decomposition sampler. In order to set up r' and r, we bound the norm of $\mathbf{e}$, where the $e_{i,j}$'s are sampled independently from a rounded continuous Gaussian of standard deviation 2^{-42}, so that $\|e\|_2 \leq \sqrt{(d+1) \cdot \ell \cdot N \cdot \vartheta_{\text{BK}}} \cdot \text{B}$ with overwhelming probability (we take $\text{B} = 10$). Taking $\varepsilon = 2^{-110}$ in Lemma 13, we obtain $\min(r, r') \geq 30825846.6$. We take r and r' both equal to 30825788.
- with $\varepsilon = 2^{-110}$ in Lemma 11, we obtain a bound on the number of TLWE samples in PK, $m \geq 184539$.

The cumulated statistical error in the algorithm CP-BlindRotate is thus $n \cdot (2 \cdot 2^{-110} + 2^{-110}) \leq 2^{-98}$. Table 1 summarizes our choice for parameters which reach this statistical bound.

ε – lem.11	ε – lem.13	r	m	$\varepsilon_{\text{total}}$
2^{-110}	2^{-110}	30825846.6	184539	2^{-98}

Table 1. Parameters and circuit privacy security bounds, taking $q = 2^{45}$, $\vartheta_{\text{BK}} = 5.17 \cdot 10^{-26}$, $\vartheta_{\text{PK}} = 5.17 \cdot 10^{-26}$, $B = 512$, $d = 1$ and $\ell = 5$.

	dim.	stdev	λ_c	λ_q
LWE	612	2^{-15}	123	85
TLWE/TGSW	2048	2^{-42}	161	117

Table 2. Security parameters for $q = 2^{45}$ obtained using the lwe-estimator. λ_c (resp. λ_q) is the security parameter obtained under the classical BKZ-sieve (resp. dual quantum) cost model.

Correctness: The final error variance after the circuit bootstrapping is given by bound (3) from Lemma 17. In order to decrypt correctly, the amplitude of the error needs to be $< \frac{1}{4}$ (rather than $\frac{1}{16}$) with overwhelming proability. With the parameters derived from the circuit privacy lemmata and setting $\mathsf{B}_{\text{KS}} = 2^3$, $t = 6$ and the keyswitching variance $\vartheta_{\mathsf{KS}} = 9.3 \cdot 10^{-10}$, we obtain an LWE ciphertext with error standard deviation ≈ 0.025, which corresponds to a probability of decryption failure below 2^{-73}.

Keys dimension, security: Table 2 shows the set of security parameters for each instance of LWE, TLWE and TGSW ciphertexts we use, where the estimation is based on the recommendation from [APS15,ACD+18,ACC+18].

- LWE - keyswitching key KS: n LWE ciphertexts with dimension $n = 612$ and noise variance $9.3 \cdot 10^{-10}$, base decomposition $\mathsf{B}_{\text{KS}} = 2^3$ and precision $t = 6$.
- TGSW - bootstrapping key BK: n TGSW ciphertexts with $B = 512$, $\ell = 5$ and $d = 1$, with noise variance $\vartheta_{\text{BK}} = 5.17 \cdot 10^{-26}$.
- TLWE - sanitization key PK: $m = 184539$ TLWE samples. For TLWE and TGSW keys, we take $N = 2048$ and $\vartheta_{\text{PK}} = \vartheta_{\text{BK}}$. The sanitization key computation step is part of the pre-processing phase.

Implementation: As a proof of concept of our technique, we provide a C implementation which can be found at `https://github.com/fhe-extension/fhe-sanitize` compatible with Windows, Linux, and MacOS. We ran multiple experiments using the parameters described previously and describe the results in Table 3. The randomness is sampled using the CSPRNG of windows or directly from /dev/urandom. We use the library FFTW [4] for our Fourier transformation, and the Gaussian sampler from [GPV08] with base sampler from [MW17] since the parameters are quite large for our discrete Gaussians.
We notice that the Gaussian sampling and generation of PkEnc(PK) can be done offline, and implemented precomputation for these. As shown by the conditions of our lemmata, the parameters required for our technique to be applied are larger than those supported by the TFHE library. Since our goal is to evaluate the overhead of our sanitization technique, we compare the non-sanitizing bootstrapping (first column of Table 3) and the sanitizing bootstrapping (remaining columns Table 3) with the same parameters on the same laptop.

The main downside of our parameter set is the size of PK. One option to bring down the size of the sanitization key would be to use the output of a pseudorandom generator for the uniform part of the ciphertexts in PK, and send the seed. However, this would only cut the size by half. In order to circumvent this issue, one possible solution would be to generate PkEnc(PK) on the fly or interactively, with m encryptions of zero being sent by

[4] from `https://www.fftw.org/`.

Bootstrap	Sanitize 5000 samples	Sanitize 500000 samples	Sanitize ∞ samples
3.15 s	21.69 s	5.20 s	4.68 s

Table 3. Timings of the concrete implementation. The experiments were run on a laptop running a 2.7 GHz Dual-Core Intel Core i5 with timings given in seconds. The first column reports the timing of our implementation of the standard TFHE bootstrapping, the second one reports the timing of our sanitization algorithm using 5000 precomputed Gaussian samples, the third one reports the timings of the same sanitization algorithm using 500000 precomputed Gaussian samples, and the last one reports the timings of the same sanitization algorithm in a setting where all the Gaussian sampling could be precomputed. In all the experiments, we assume that all required encryptions of zero are precomputed.

the user. Our experiments shows that it takes roughly 45 seconds in order to generate one $\mathsf{PkEnc}(\mathrm{PK})$ in the latter way on the same personal laptop as used for the results given in Table 3.
For our experiments, we assume that the server has access to unrestricted memory and precomputation, and we emulate this setting by precomputing only one element and reusing it.

6 Discussion : removing PkEnc

As discussed in the introduction, for our set of parameters, we need to add fresh encryptions of zero as well as in order to get ciphertexts with well distributed uniformly random parts. For the sake of completeness, we capture the conditions where thes this step can be avoided for larger parameters, and we discuss in this section the required conditions.

Lemma 19. *If the parameter ℓ and Gaussian noise parameter r are large enough, Lemma 18 still holds if* $\mathrm{PkEnc}(\mathrm{PK})$ *is removed at each loop iteration of Algorithm 2.*

Since this technique leads to an inefficient satinization algorithm, the condition for the required bounds are detailed into the following proof.

Proof. First, let us discuss the structure of R_q, where $q = 2^k$. Since $X^N + 1 = (X+1)^N \mod 2$, the ideals of R_q are generated by $X+1$ and 2, so any ideal of R_q can be written as $2^i(X+1)^j R_q$, for some $i \in [0, k-1]$ and some $j \in [0, N-1]$.
Moreover, we have that in R_q, $(X+1)^N = 2Q(X)$, with Q an invertible polynomial in R_q. This follows from the following three facts: First, $(X+1)^N = 1 + 2X^{\frac{N}{2}} + X^N \mod 4$; this can be easily shown by induction (recall that N is a power of 2). Second, for any polynomial $P \in R_q$, P is invertible if and only if $P \mod (X+1)$ is odd. If d is odd, $((X+1)C(X)+d)^{-1} = \frac{1}{d}\sum_{i=0}^{N-1}\left(-\frac{(X+1)C(X)}{d}\right)^i$. Third and last, the parity of $P \mod (X+1)$ is the same as the parity of the sum of the coefficients of P.

Combining all of these, we have that $X^{\frac{N}{2}}$ is invertible, so $(X+1)^N R_q = 2R_q$, which gives us the following alternative definition for the ideals of R_q: any ideal of R_q is of the form $I_t = (X+1)^t R_q$ for some $t \in [0, Nk-1]$.

We also have that any polynomial $P \in R_q$ can be uniquely described as

$$P = \sum_{i=0}^{Nk-1} p_i (X+1)^i, \text{ with } p_i \in \{0,1\}.$$

In the following, we will use $P \mod (X+1)^j$ to denote the truncated sum $\sum_{i=0}^{j-1} p_i(X+1)^i$, so $P \mod (X+1)$ is actually $P \mod 2 \mod (X+1)$.

We also define the gcd of two elements of R_q in the following natural way:

$$\gcd(P, P') \text{ is the smallest } t \text{ in } [0, Nk-1] \text{ such that } P, P' \in I_t$$

We are now ready to dive into the proof. We want to show that for any $\mathbf{e} \in \hat{R}_q^{(d+1)\cdot\ell}$, $\mathbf{y} \in \hat{R}_q$, $\mathbf{c} \in \hat{R}_q^{(d+1)\cdot\ell}$:

$$\left(\mathbf{A}, \mathbf{x}^t \cdot \mathbf{A}, \mathbf{e}^t\mathbf{x} + \mathbf{y}\right) \approx_s \left(\mathbf{A}, \mathbf{u}^t, \mathbf{e}^t\mathbf{x} + \mathbf{y}\right),$$

where $\mathbf{A}$ is uniform over $\hat{R}_q^{(d+1)\cdot\ell\times d}$, $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r}$ and $\mathbf{u}$ is uniform over $\hat{R}_q^{\,d}$. The remainder of the proof is handled exactly as in subsection 3.2. However, our situation does not match the usual setting to use the Leftover Hash Lemma over a ring structure, since we are dealing with rings, modulo a power of 2. Thankfully, we can adapt the proof of [MM11, Lemma 2.3]. For any distribution $\mathcal{Z}$ over a set Z, we have the following relation between the statistical distance between $\mathcal{Z}$ and the uniform distribution over Z, $\mathcal{U}_Z$, with the collision probability of $\mathcal{Z}$, $\mathrm{Col}(\mathcal{Z})$.

$$\Delta(\mathcal{Z}, \mathcal{U}_Z) \leq \frac{1}{2}\sqrt{|Z| \cdot \mathrm{Col}(\mathcal{Z}) - 1}.$$

We are interested in the distance to uniformity of $\left(\mathbf{A}, \mathbf{x}^t \cdot \mathbf{A}\right)$ conditioned on $\mathbf{e}^t\mathbf{x} + \mathbf{y}$, so following their arguments, we have this distance to uniformity bounded by

$$\frac{1}{2}\sqrt{q^{Nd}\Pr\left((\mathbf{x}^t - \mathbf{x}'^t) \cdot \mathbf{A} = \mathbf{0}\right)}$$

with $\mathbf{A}$ uniform and $\mathbf{x}, \mathbf{x}' \leftarrow \mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r}$ conditioned on $\mathbf{e}^t\mathbf{x}+\mathbf{y}$. Following their techniques, we can condition on the gcd of $\mathbf{x}$ and $\mathbf{x}'$, and obtain: for any distribution $\mathcal{X}$ on $R_q^{(d+1)\cdot\ell}$, if $\gcd(\mathbf{x}, \mathbf{x}') = t$, $\Pr\left((\mathbf{x}^t - \mathbf{x}'^t) \cdot \mathbf{A} = \mathbf{0}\right) = \frac{1}{|I_t|^d}$, so

$$\Delta\left((\mathbf{A}, \mathbf{x}^t \cdot \mathbf{A}), (\mathbf{A}, \mathbf{u}^t)\right) \leq \frac{1}{2}\sqrt{\sum_{t=1}^{Nk-1} \frac{1}{2^{d(Nk-t)}} \cdot \mathrm{Col}(\mathcal{X}_t)}$$

where $x \leftarrow \mathcal{X}$ and $\mathcal{X}_t = \mathcal{X} \mod (X+1)^t$. To conclude the proof, we just need to analyze the collision probability of the distribution $\mathbf{x} \mod (X+1)^t$ conditioned on $\mathbf{e}^t\mathbf{x} + \mathbf{y}$, for any $t \in \{1, Nk-1\}$, where $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r}$, which we bound by its maximal value, when $t = 1$. In order to bound this quantity, we use the min-entropy of $\mathbf{x} \mod (X+1)$. Since $\mathbf{e}^t\mathbf{x} + \mathbf{y}$ only has q^{Nd} possibilities, so the min-entropy of $\mathbf{x} \mod (X+1)$ conditioned on $\mathbf{e}^t\mathbf{x} + \mathbf{y}$ is greater than $H_\infty(\mathbf{x} \mod (X+1)) - Nkd$. For any $\mathbf{v} \in \left(\Lambda^\perp(\mathbf{G}) + \mathbf{c}\right) \mod (X+1)$, if the conditions of Lemma 4 are met, we have:

$$\begin{aligned}\left(\mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r} \mod (X+1)\right)(\mathbf{v}) &= \frac{\rho_r(\mathbf{v} + (X+1)\Lambda^\perp(\mathbf{G}))}{\rho_r(\Lambda^\perp(\mathbf{G}) + \mathbf{c})} \\ &\leq \frac{(1+\varepsilon)r^{(d+1)\cdot\ell}\det(\Lambda^\perp(\mathbf{G}))}{(1-\varepsilon)r^{(d+1)\cdot\ell}\det((X+1)\Lambda^\perp(\mathbf{G}))}\end{aligned}$$

Let us analyze the lattice $(X+1)\Lambda^\perp(\mathbf{G})$ more precisely. We define $\mathbf{J} \in \mathbb{Z}^{N\times N}$ the matrix such that $\mathtt{pow}_X \cdot \mathbf{J} = X+1$, and $\mathbf{B}_0 \in \mathbb{Z}^{\ell\times\ell}$ the basis of the lattice $\{\mathbf{u} \in \frac{1}{q}\mathbb{Z}^\ell \mid \mathbf{gu} \in \mathbb{Z}\}$, that is:

$$\mathbf{J} = \begin{pmatrix} 1 & 0 & \dots & 0 & -1 \\ 1 & 1 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & 1 & 0 \\ 0 & \dots & 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{B}_0 = \begin{pmatrix} B & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & B \end{pmatrix}$$

Note that while $\Lambda^\perp(\mathbf{G})$ is generated by the matrix $\mathbf{I}_d \otimes \mathbf{B}_0 \otimes \mathbf{I}_N$, $(X+1)\Lambda^\perp(\mathbf{G})$ is generated by $\mathbf{I}_d \otimes \mathbf{B}_0 \otimes \mathbf{J}$. Since $\det(\mathbf{J}) = 2$, we have that $\det\big((x+1)\Lambda^\perp(\mathbf{G})\big) = 2^{(d+1)\cdot\ell}\det\big(\Lambda^\perp(\mathbf{G})\big)$. This gives us the following bound on the min-entropy of $\mathbf{x} \mod (X+1)$:

$$H_\infty\left(\mathbf{x} \;\middle|\; \mathbf{e}^t\mathbf{x} + \mathbf{y}\right) \geq (d+1)\cdot\ell + \log\left(\frac{1-\varepsilon}{1+\varepsilon}\right) - Nkd$$

Fixing ℓ large enough will ensure that the collision probability is below any chosen value, allowing a choice of parameters that effectively yields sanitization without the need for adding fresh encryptions of zero. The last condition that has to be met is that the Gaussian parameter r has to be bigger than the smoothing parameters of both $\Lambda^\perp(\mathbf{G})$ and $(X+1)\Lambda^\perp(\mathbf{G})$. We conclude our discussion by showing an upper bound on $\eta_\varepsilon\big((X+1)\Lambda^\perp(\mathbf{G})\big)$, the bigger one. The columns of $\mathbf{I}_d \otimes \mathbf{B}_0 \otimes \mathbf{J}$ all have their norm bounded by $\sqrt{2+2B^2}$. So Lemma 2 gives us the bound $\eta_\varepsilon\big((X+1)\Lambda^\perp(\mathbf{G})\big) \leq \sqrt{(2+2B^2)\cdot\frac{\ln(2N(d+1)\cdot\ell(1+1/\varepsilon))}{\pi}}$. □

References

ACC+18. Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

ACD+18. Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 351–367. Springer, Heidelberg, September 2018.

AGHS13. Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian leftover hash lemma over infinite domains. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 97–116. Springer, Heidelberg, December 2013.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

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

AP20. Marc Abboud and Thomas Prest. Cryptographic divergences: New techniques and new applications. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 492–511. Springer, Heidelberg, September 2020.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `https://eprint.iacr.org/2015/046`.

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Ban93. Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Annals of Mathematics*, 296(4):625–635, 1993.

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

BdMW16. Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89. Springer, Heidelberg, August 2016.

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

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

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

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

BV14. Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In Moni Naor, editor, *ITCS*, pages 1–12. ACM, 2014.

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

CHK+18. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 360–384. Springer, Heidelberg, April / May 2018.

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

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

CLOT21. Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In *Asiacrypt*, LNCS 13092, pages 670–699. Spinger-Verlag, 2021.

CLR17. Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

CMdG+21. Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In *CCS 2021*, pages 1135–1150. ACM, 2021.

CZ17. Long Chen and Zhenfeng Zhang. Bootstrapping fully homomorphic encryption with ring plaintexts within polynomial noise. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *ProvSec 2017*, volume 10592 of *LNCS*, pages 285–304. Springer, Heidelberg, October 2017.

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

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

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

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

GBA21. Antonio Guimarães, Edson Borin, and Diego F. Aranha. Revisiting the functional bootstrap in TFHE. *IACR TCHES*, 2021(2):229–253, 2021. `https://tches.iacr.org/index.php/TCHES/article/view/8793`.

Gen09. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

GHV10. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, Heidelberg, August 2010.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

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

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

LW20. Feng-Hao Liu and Zhedong Wang. Rounding in the rings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 296–326. Springer, Heidelberg, August 2020.

MM11. Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 465–484. Springer, Heidelberg, August 2011.

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

MR07. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

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

MW17. Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 455–485. Springer, Heidelberg, August 2017.

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

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Heidelberg, August 2010.

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

Supplementary material

A Proof of Lemma 14

We first recall the lemma we want to prove.

Lemma 14. *Let $\varepsilon > 0$. For any $\mathbf{e} \in \hat{R}_q^{(d+1)\cdot\ell}$, let $\mathcal{L}$ be as defined in the proof of Lemma 13. Then we have:*

$$\eta_\varepsilon(\mathcal{L}) \leq \sqrt{1+B^2}(1+q\|\mathbf{e}\|_2)\cdot\sqrt{\frac{\ln(2\cdot N\cdot(d+1)\cdot\ell\cdot(1+1/\varepsilon))}{\pi}}.$$

Proof. We use Lemma 2 to bound the smoothing parameter of $\mathcal{L}$. Since $\widehat{\Lambda} = \Lambda^\perp(\mathbf{G})\times\mathbb{Z}^N$ is of dimension $N((d+1)\cdot\ell+1)$ and $\mathcal{L}$ is the sublattice of $\widehat{\Lambda}$ made of the vectors that are orthogonal to $\widehat{\mathbf{E}}$, we have that $\mathcal{L}$ is of dimension $N(d+1)\cdot\ell$. We thus exhibit $N(d+1)\cdot\ell$ independent short vectors of $\mathcal{L}$ to obtain an upper bound on $\lambda_{N(d+1)\cdot\ell}(\mathcal{L})$. We first define the matrix

$$\overline{\mathbf{B}} = \begin{pmatrix} B\mathbf{I}_N & -\mathbf{I}_N & & \\ & \ddots & \ddots & \\ & & \ddots & -\mathbf{I}_N \\ & & & B\mathbf{I}_N \end{pmatrix} \in \mathbb{Z}^{N\ell\times N\ell}$$

and remark that it is a basis for the lattice $\Lambda^\perp(\mathbf{g}) = \{\mathbf{u}\in\mathbb{Z}^{N\ell} \mid \mathbf{g}^t\mathbf{u}\in\mathbb{Z}\}$. The lattice $\widehat{\Lambda}$ is then generated by the columns of the matrix:

$$\mathbf{B} = (\mathbf{B}_1 \mid \ldots \mid \mathbf{B}_{(d+1)\cdot\ell+1}) = \left(\begin{array}{c|c} \mathbf{I}_{d+1}\otimes\overline{\mathbf{B}} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I}_N \end{array}\right) \in \mathbb{Z}^{N((d+1)\cdot\ell+1)\times N((d+1)\cdot\ell+1)}$$

For $k \leq (d+1)\cdot\ell$ let $\mathbf{U}_k = \mathbf{B}_k - q\mathbf{B}_{(d+1)\cdot\ell+1}\widehat{\mathbf{E}}\mathbf{B}_k$, since $\widehat{\mathbf{E}}\mathbf{B}_{(d+1)\cdot\ell+1} = \frac{1}{q}\mathbf{I}_N$ we directly have $\widehat{\mathbf{E}}\mathbf{U}_k = 0$ and thus $\mathbf{U}_k \in \mathcal{L}$. The vectors $\mathbf{U}_{1,1},\ldots,\mathbf{U}_{(d+1)\cdot\ell,N}$ are linearly independent since $\operatorname{span}\left(\mathbf{U}_{1,1},\ldots,\mathbf{U}_{(d+1)\cdot\ell,N},\mathbf{B}_{(d+1)\cdot\ell+1,1},\ldots,\mathbf{B}_{(d+1)\cdot\ell+1,N}\right) = \operatorname{span}\left(\mathbf{B}_{1,1},\ldots,\mathbf{B}_{(d+1)\cdot\ell+1,N}\right) = \mathbb{R}^{N((d+1)\cdot\ell+1)}$ (which comes from the fact that $\mathbf{B}$ is a basis of an $N((d+1)\cdot\ell+1)$-dimensional lattice). We now bound the norm of $\mathbf{U}_{k,i}$:

$$\begin{aligned}\|\mathbf{U}_{k,i}\|_2 &\leq \|\mathbf{B}_{k,i}\|_2 + q\left\|\left(\mathbf{B}_{(d+1)\cdot\ell+1}\widehat{\mathbf{E}}\mathbf{B}_k\right)_i\right\|_2 \\ &\leq \sqrt{1+B^2} + q\|\mathbf{e}\|_2\sqrt{1+B^2} \\ &= \sqrt{1+B^2}(1+q\|\mathbf{e}\|_2)\end{aligned}$$

Finally we obtain $\lambda_{N(d+1)\cdot\ell}(\mathcal{L}) \leq \sqrt{1+B^2}(1+q\|\mathbf{e}\|)$ and the result.

B TFHE Bootstrapping building blocks

B.1 $\mathbf{G}^{-1}(\cdot)$ computation

In Algorithm 3, we recall the $\mathbf{G}^{-1}$ algorithm that decomposes vectors in $\hat{R}_q^{1\times(d+1)}$ into vectors in $R^{1\times(d+1)\cdot\ell}$.

Algorithm 3 Gadget Decomposition

Input: A vector $\mathbf{v} = (v_1 \mid \ldots \mid v_{d+1}) \in \hat{R}_q^{1\times(d+1)}$
Output: A vector $\mathbf{x}$ such that $x \cdot \mathbf{G} = \mathbf{v}$ and $\|\mathbf{x}\|_\infty \leq B/2$

1: For each v_i choose the unique representative $\sum_{j=0}^{N-1} v_{i,j}X^j$, with $v_{i,j} \in \hat{\mathbb{Z}}_q$. Note that $v_{i,j}$ is an exact multiple of $\frac{1}{B^\ell}$.
2: Decompose each $v_{i,j}$ uniquely as $\sum_{p=1}^{\ell} v_{i,j,p}\frac{1}{B^p}$ where each $v_{i,j,p} \in [-B/2, B/2[$
3: **for** $i = 1$ **to** $d+1$
4: **for** $p = 1$ **to** ℓ
5: $x_{i,p} = \sum_{j=0}^{N-1} v_{i,j,p}X^j \in R$
6: Return $(x_{1,1} \mid \ldots \mid x_{d+1,\ell})$

Algorithm 4 Randomized Gadget Decomposition

Input: A vector $\mathbf{v} = (v_1 \mid \ldots \mid v_{d+1}) \in \hat{R}_q^{1\times(d+1)}$
Output: A vector $\mathbf{x}$ from a spherical Gaussian distribution with parameter r such that $x \cdot \mathbf{G} = \mathbf{v}$

1: For each v_i choose the unique representative $\sum_{j=0}^{N-1} v_{i,j}X^j$, with $v_{i,j} \in \hat{\mathbb{Z}}_q$. Note that $v_{i,j}$ is an exact multiple of $\frac{1}{B^\ell}$.
2: Set $\mathbf{v}_j = (v_{1,j} \mid \ldots \mid v_{d+1,j})$
3: For each j sample $\mathbf{x}_j = (x_{1,j}, \ldots, x_{(d+1)\cdot\ell,j}) \in R^{(d+1)\cdot\ell}$ from $\Lambda^\perp_{\mathbf{v}_j}(\mathbf{G})$ with parameter r
4: **for** $i = 1$ **to** $(d+1)\ell$
5: $x_i = \sum_{j=0}^{N-1} x_{i,j}X^j \in R$
6: Return $\left(x_1 \mid \ldots \mid x_{(d+1)\cdot\ell}\right)$

B.2 $\mathbf{G}_r^{-1}(\cdot)$ computation

In Algorithm 4, we recall the $\mathbf{G}_r^{-1}$ algorithm that decomposes vectors in $\hat{R}_q^{1\times(d+1)}$ into vectors in $R^{1\times(d+1)\cdot\ell}$ with spherical Gaussian distribution on a coset of $\Lambda^\perp(\mathbf{G})$.

B.3 Extract

In Algorithm 5, we recall the Extract algorithm that transforms TLWE ciphertexts into N-LWE ciphertexts.

Algorithm 5 Extracting an N-LWE ciphertext of the constant term from a TLWE ciphertext

Input: $\mathbf{c} = (\mathbf{a}, b) \in \hat{R}_q^{d+1}$.
Output: Extract $(\mathbf{c})$: an N-LWE encryption of $\varphi_{\tilde{\mathbf{s}}}(\mathbf{c})$.

1: $b' = b(0)$.
2: **for** $i = 1$ **to** d
3: Set $\left(a'_{iN+j}\right)_{j\in[0,N-1]}$ such that $\sum_{j=0}^{N-1} a'_{iN+j}X^j = a_i(\frac{1}{X})$.
4: Return $\mathbf{c}' = (\mathbf{a}', b')$.

B.4 KeySwitch

In Algorithm 6, we recall the keyswitching procedure of the bootstrapping, and the analysis of noise growth is provided in Lemma 20

Algorithm 6 keyswitching from secret key K of dimension dN to secret key $\mathbf{s}$ of dimension n

Input: $\mathbf{c} = (\mathbf{a}, b) \in \hat{\mathbb{Z}}_q^{dN+1}$, a keyswitching key $\mathrm{KS} = (\mathrm{KS}_{i,j})_{i,j}$, where $\mathrm{KS}_{i,j}$ is an n-LWE encryption of $K_i \frac{1}{\mathsf{B}_{\mathrm{KS}}^j}$ for $i \in [1, N]$ and $j \in [1, t]$.

Output: $\mathsf{KeySwitch}(\mathbf{c}, \mathrm{KS})$: an n-LWE encryption of $\varphi_K(\mathbf{c})$.

1: **for** $i = 1$ **to** dN

2: Round a_i to the nearest element $\lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t}$ in $\frac{1}{\mathsf{B}_{\mathrm{KS}}^t}\mathbb{Z}$.

3: Set $(a_{i,j})_{j\in[1,t]}$ such that $\sum_{j=1}^{t} a_{i,j} \frac{1}{\mathsf{B}_{\mathrm{KS}}^j} = \lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t}$ (Decompose $\lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t}$ in basis B_{KS}).

4: Return $\mathbf{c}' = (\mathbf{0}, b) - \sum_{i,j} a_{i,j} \mathrm{KS}_{i,j}$ where the sum ranges over $[1, dN] \times [1, t]$.

Lemma 20 (Algorithm 6 noise propagation). *Let* $\mathbf{c} \in \hat{\mathbb{Z}}_q^{dN+1}$, $\mathrm{KS} = (\mathrm{KS}_{i,j})_{i,j}$, *where* $\mathrm{KS}_{i,j}$ *is an* n-LWE *encryption of* $K_i \frac{1}{\mathsf{B}_{\mathrm{KS}}^j}$ *for* $i \in [1, dN]$ *and* $j \in [1, t]$ *with noise variance* ϑ_{KS}. *Algorithm 6 outputs a sample* $\mathbf{c}' \in n$-$\mathsf{LWE}_{\mathbf{s}}(\varphi_K(\mathbf{c}))$ *such that:*

$$\mathsf{Var}\left(\mathsf{Err}(\mathbf{c}')\right) \leq \mathsf{Var}\left(\mathsf{Err}(\mathbf{c})\right) + d\frac{tN\mathsf{B}_{\mathrm{KS}}^2}{4}\vartheta_{\mathrm{KS}} + \frac{dN}{4\mathsf{B}_{\mathrm{KS}}^{2t}} \tag{4}$$

Proof. During the first step of the algorithm, each rounding induces an error of at most $\frac{1}{2\mathsf{B}_{\mathrm{KS}}^t}$, hence the resulting variance for dN roundings.
Each part of the keyswitching key is multiplied by a coefficient between $-\frac{1}{2\mathsf{B}_{\mathrm{KS}}}$ and $\frac{1}{2\mathsf{B}_{\mathrm{KS}}}$, hence the resulting variance for combining dtN of them.

C Comparison with [DS16] satinatization strategy

In this section, we evaluate the cost of sanitizing a TFHE ciphertext using [DS16] strategy with our parameter set. As in their application to FHEW, our analysis only estimates the cost of their approach, not taking into account the probability of decryption failure. We note that this should play in their advantage in this comparison, and that a serious implementation of this technique should further refine this analysis.

Recall that [DS16] satinatization algorithm consists of iterations of two steps: (1) a refresh step which, given an LWE ciphertext as input, brings the noise down to a low level; (2) a rerandomization step which rerandomizes a ciphertext by adding a linear combination of encryptions of 0 (equivalent to our PkEnc procedure) and adding a uniform noise to decrease the statistical distance between two hypothetical ciphertexts (a poor man's noise flooding, counterpart of our small Gaussian noise).

As done in the analysis of [DS16] applied on FHEW for maximal efficiency, this rerandomization is done right after the $\mathsf{Extract}$, before switching to a lower dimension, so that the noise in the encryptions of 0 can be chosen smaller.

The number of cycles in [DS16]'s strategy varies depending on the scheme and on the parameters. This number is computed from the magnitude of the noise of the current ciphertext compared to the magnitude of the noise introduced by Rerand.

We give in Figure 2 a high-level overview of both strategies to highlight the differences between them: while our rerandomization steps are similar, we perform them at each step of the BlindRotate algorithm, whereas the soak-and-spin strategy applies it only once per iteration of the bootstrapping, but requires between 5 and 6 such iterations to reach the same level of security.

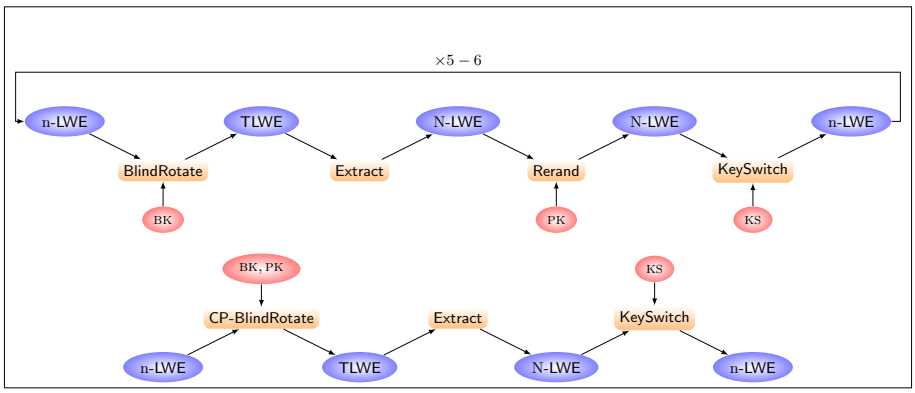

Fig. 2. High-level overview of our sanitization strategy (bottom part) and [DS16] strategy (top part).

The first BlindRotate step outputs a LWE sample of standard deviation around 2^{-21} which gives a sample of noise amplitude less that 2^{-18} with probability $1-2^{-49}$. Adding a linear combination of $n \cdot \log q$ encryptions of zero with standard deviation 2^{-15} introduces a noise negligible compared to the noise introduced by the BlindRotate step so that the standard deviation is still around 2^{-21}. The soaking noise amplitude B can be chosen up to around 0.14 and still allows correct decryption when taking into account the noise introduced by the keyswitching step. With these parameters, the statistical distance between two ciphextext in $\mathcal{C}_\mu$ reached after is around $\delta = 2^{-18}$. In order to reach the same statiscial distance as ours, we need between 5 and 6 cycles, which takes around 15 and 19 seconds with our set of parameters, according to our experiments.

To complete the comparison, we also note that our approach requires only one bootstrapping, thus reducing the probability of decryption failure.