Plug-and-play sanitization for TFHE

Florian Bourse and Malika Izabachène

Abstract. Fully Homomorphic encryption allows the evaluation of any circuits over encrypted data while preserving the privacy of the data. However, without any additional properties, no guarantee is provided for the privacy of the circuits which are evaluated.
A sanitization algorithm allows to destroy all previous information about how a ciphertext was obtained, ensuring that the circuit which was evaluated remains secret. In this paper, we present two techniques to randomize RLWE ciphertexts, and show how they can be used to achieve ciphertext sanitization for the TFHE scheme proposed by Chilotti *et al.* (Asiacrypt 2016), by modifying the bootstrapping procedure internally. The first technique is a generalization of the strategy proposed by Bourse *et al.* (Crypto 2016) to the ring setting. While this approach adapts well in theory, we show evidence that it fails to provide a practical solution.
To improve over this strategy, we relax the circuit privacy property to its computational counterpart, and make use of an efficient public randomizer composed of an RLWE-based public key encryption with additional properties on the ciphertexts distribution. This randomizer can also be used in the soak-and-spin paradigm of Ducas and Stehlé (Eurocrypt 2016). Using a backward induction over the circuit size, we also improve on the proof technique from Bourse *et al.* to avoid randomization at each step of the computation, enabling faster randomization and smaller noise growth.
As a proof of concept, we provide a C implementation of our sanitization strategy, which shows that a sanitized LWE ciphertext can be obtained almost for free compared to a bootstrapped LWE ciphertext assuming many discrete Gaussian samples at hand.

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 the 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,GPL23]) provide 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 to bootstrap 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 i-hop circuit-private FHE scheme 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 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 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 by adding a fresh encryption of zero with a super-polynomially larger noise. Indeed, the noise contained in a ciphertext is the main source of leakage. And the idea is to add a noise with a 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 super-polynomial, which makes the security rely on a stronger LWE assumption, and requires larger parameters.

A new approach, proposed in [DS16] avoids this downside, and achieves circuit-privacy for FHE schemes relying on LWE with a polynomial modulus-to-noise ratio. Their sanitization algorithm relies on the use of the bootstrapping and their technique is based on a paradigm called *soak and spin*, which consists of iterations of two consecutive steps: a first one 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 use of the Rényi divergence instead of the statistical distance. They showed 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.

In [BdMW16], Bourse *et al.* showed how to achieve circuit privacy for NC^1 circuits without relying on the circular security assumption, and basing its se-

curity on plain LWE with polynomial modulus-to-noise ratio. The main idea is to tweak the computation paradigm to branching programs, which are shown to be able to 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 after each product. 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 homomorphic product. Adding a random Gaussian noise in the second step ensures that at each step, the noise does not 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.

A concurrent work [Klu22] claimed a similar LWE sanitization result by modifying the FHEW-like bootstrapping. The proof of the claim was not correct: in the proof of Theorem 1, in the transition from Hybrid 1 to Hybrid 2, Lemma 11 is invoked with $w = 1$ on A of dimension $N \times \ell_R$ for a polynomial of degree N. However, in Lemma 11, the matrix A has dimension $w - 1 \times m$ for some parameter m. We contacted the author regarding this issue and the proof has been recently revised, so we will proceed with comparing the updated version.
On the theoretical side, one of our contribution is to extend the noise analysis from [BdMW16] to the ring setting, which additionally implies multi-hop circuit privacy over rings without bootstrapping for branching programs, whereas [Klu22] sanitization strategy cannot be used independently of the bootstrapping. [Klu22] only considers statistical sanitization security. Though statistical sanitization is stronger, we investigate other trade-offs by proposing a second sanitization strategy that relies on RLWE, a computational assumption. With this relaxation, the sanitization property holds under the same assumption as for the semantic security, allowing us to considerably reduce the size of the public key. For sake of comparison, in the new version of [Klu22], the public sanitization key is composed of $2^{11.5}$ to $2^{12.3}$ fresh LWE samples of zero in dimension $N = 2^{11}$, while in our case it is made of one RLWE encryption of zero in dimension $N = 2^{10}$. It is also noteworthy that [Klu22]'s implementation differs from the scheme in two aspects: it samples rounded continuous Gaussian instead of discrete Gaussians and it makes use of a mersenne twister generator as a random source for the gaussian samples while we make use of the operating system's CSPRNG, which is cryptographically secure.

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 bootstrappings. Also, our sanitization algorithm is compatible with recent improved

functional bootstrapping [BDF18,CIM19,GBA21,CLOT21,LY23] which allows to support larger space messages with higher precision, provided the parameters are adapted. We let the practical analysis of the combination of circuit-privacy with each of those techniques for future work.

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] randomization strategy for the TFHE bootstrapping. We discuss the comparison of both approaches in details in Section 5.1 and give detailed timings in the implementation section where we explain how our lemmata are used. In [DS16], the randomization step makes use of a large combination of LWE encryptions of zero. Compared to [DS16] strategy, our solution does not need to assume the randomizer to be pre-computed or computed on the fly since it is made of a small Gaussian noise plus only one RLWE encryption of zero and can be computed very efficiently. As the randomizer we propose consists of a much smaller public key, it allows to achieve a faster randomization and a smaller noise overhead after the procedure. It can also be used directly in combination with other work including the soak-and-spin technique from [DS16].

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 indistinguishable 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 for functions in NC^1 as in [BdMW16].

Our first solution generalizes the randomization techniques from [AP14,BdMW16] to the ring setting (see Lemma 9). One step of our generalization needs to make use of a regularity lemma over ring structures for power-of-two modulus. General Leftover Hash Lemma results over rings do exist ([LW20, Theorem 5.5] for example), but for these general results, the probability of collision over each prime ideal needs to be effectively computed. While not setting restriction on q, [LPR13, Corollary 7.5] provides a regularity lemma over rings but does not handle conditional distribution over lattice cosets. [DGKS21] also proves a regularity lemma over rings for conditional distributions when q is prime. In order to derive more efficient bounds for the power-of-two modulus case, we adapt the techniques from [MM11] and specialize the probability of collision over each prime ideals of power-two-modulus ring for conditional distributions. We also develop additional techniques to analyze the distributions of polynomials – combinations of uniformly random polynomials and discrete Gaussian polynomials over lattices for the specialized case of a power-of-two cyclotomic ring with a power-of-two modulus, which can also be adapted to other ring structures.

We propose two sanitization strategies based on a combination of a randomized decomposition and a public randomizer, which can be either taken as a small Gaussian noise or as the sum of a small Gaussian noise plus one additional RLWE encryption of zero. As a direct implication of our first strategy and [BdMW16]

result, we obtain circuit-private branching programs for homomorphic computation over rings without relying on additional public key material. The first strategy yields statistical circuit privacy under the assumption that the parameters are large. Our second strategy removes the hard constraints on the parameters and yields computational circuit privacy in a more efficient way. We provide an implementation of our sanitization algorithm showing that sanitization of FHE ciphertexts can be achieved in practice. As the TFHE bootstrapping does not provide any sanitization property, our parameters fall in a different range than the ones chosen for the TFHE and Concrete libraries [CGGI16,Zam22] which provide very efficient non-sanitized bootstrapping. In order to measure the quality of our sanitization strategy, we compare the relative cost between the non-sanitized bootstrapping versus the sanitized bootstrapping for the same set of parameters. In particular, we measured that if Gaussian samples could be pre-computed, or generated in parallel of the sanitization execution[1], the overhead of our technique is almost negligible, and since the sanitization can be done in place of the last bootstrapping after many computation steps, one can achieve circuit privacy almost for free in that case.

Overview of the techniques. The underlying idea of our high-level strategy is based on two observations. Recall that the bootstrapping procedure of TFHE can be decomposed in 3 steps, called BlindRotate, Extract, and KeySwitch, as illustrated on Figure 1. So 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.

Our focus will be on the BlindRotate procedure which we carefully modify to be a circuit private evaluation for the class of function $\mathsf{BlindRotate}_{\mathbf{a},b} : \mathbf{s} \mapsto (0, \mathsf{testv}) \cdot X^{b-\sum_i s_i a_i}$ parameterized by a LWE ciphertext $(\mathbf{a}, b)$ for any polynomial testv and a variable $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$. We notice that the BlindRotate computation is mostly made of homomorphic evaluations of multiplexers i.e. conditional operators, which behave nicely with respect to the circuit privacy property as shown in [BdMW16] [2]. We modify the BlindRotate procedure to make it circuit-private, by randomizing the gadget decomposition that we combine with a fresh randomizer at the end of the computation.

To avoid some of the dependencies between random variables, [BdMW16] proceeds by induction and analyzes the distribution of the ciphertext after each step of the computation, starting from the initial state. In order to prove their result, a small Gaussian noise is required to be added at each step of the computation. We improve on their strategy by using another proof by induction, starting from the end towards the beginning, analyzing the noise added by the remainder of the computation, and we show that adding just one public randomizer at the very end is enough to prove FHE ciphertexts sanitization and

[1] We believe that specialized hardware or libraries would significantly reduce the timing of the underlying Gaussian sampler, even without pre-computation.

[2] [BdMW16] shows circuit-private evaluation for branching programs, while not exactly being the same, it is close to multiplexers evaluation.

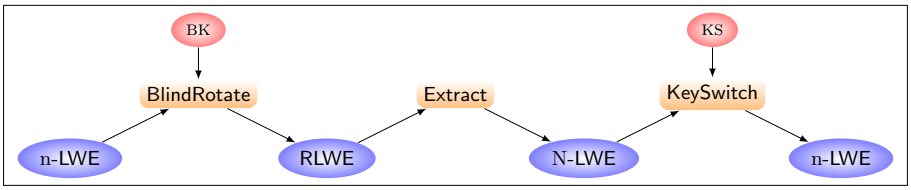

Fig. 1. TFHE bootstrapping steps: n-LWE denotes a LWE ciphertext of dimension n. After an initialization step not pictured here, the bootstrapping of an n-LWE ciphertext into a fresh n-LWE ciphertext goes through three algorithms BlindRotate, Extract, and KeySwitch and requires two sets of keys: the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

hence circuit private evaluation. We notice that this proof technique can also be applied to their setting as well and improve the parameters.

Additional challenges have to be tackled as BlindRotate is not exactly a branching program. 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, but there are also multiplications by powers of X being done on the underlying plaintext, and also on the previous states of the accumulator inside the bootstrapping. In addition, as we are dealing with polynomials and not scalar vectors, which means that the different components are combined with each other along products, the standard tools at hand for analyzing distribution cannot apply directly 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. We then explicitly state the matrix-vector products along 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: a vector that should be sampled uniformly at random, and the noise component and both parts need to be considered by our analysis.

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. As a side contribution, we generalize the noise analysis from [BdMW16] to the ring setting and to arbitrary lattices. We also extend underlying lemmata and derive underlying bounds to handle Gaussian parameters that are not necessarily spherical by viewing products of polynomials in our specific case as matrix-vector products.

We also specialize the Leftover Hash Lemma to our setting, by analyzing the structure of the ring, in order to randomize the uniform random vector part.

However, this requires quite large vectors, especially since we consider a power-of-two modulus that 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 super-logarithmic in the security parameter λ. This is because with probability $\frac{1}{2^m}$, none of the elements of $\mathbf{a}$ is invertible in such a ring, so the result is not invertible. The bound we derive on the parameters with this technique was not practical. The full conditions are given in Lemma 12 of Section 3. In TFHE, the parameters are optimized for efficient bootstrapped operations, and in that case, the polynomial vectors could be of size at most 6 and of size 7. So this first approach cannot be used for a practical implementation of circuit privacy or sanitization. A possible strategy would be to make use of an additional public key encryptions of zero as in [DS16]. However, as this technique relies on the Leftover Hash Lemma, it requires many encryptions of zero i.e. $n \log q$, where n is the LWE dimension and q is the modulus.

In Section 3.4, we propose a more efficient strategy that makes use of a suitable public key encryption scheme for which we characterize the requisites to reach FHE ciphertexts sanitization. In practice, the public randomizer just consists of a single encryption of zero, which is multiplied by a fresh Gaussian vector, and can be computed very efficiently. In this case, the sanitization property is relaxed to its computational counterpart, relying on the hardness of the decision RLWE problem with not necessarily binary secret key distribution. This construction is inspired by the techniques used in the context of Functional Encryption in [MKMS22] where one needs to protect its data against an adversary that can partially decrypt the ciphertext. We believe that this result is of independent interest. In particular, it can be plugged directly together with the technique of [DS16] to improve its efficiency at the cost of relaxing the sanitization property to hold computationally.

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.

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 $[1, n]$ for convenience. We use lower case bold font, e.g. $\mathbf{a}$, to denote (possibly row) vectors, and upper case bold font, e.g. $\mathbf{A}$, to denote matrices. We write 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 define the gadget matrix as $\mathbf{G} = \mathbf{I}_{d+1} \otimes \mathbf{g}^t$, i.e.

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

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

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

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. We say that they are computationally close and we write $\mathcal{P} \approx_c \mathcal{Q}$ if no algorithm can distinguish between the two distributions in polynomial time with non-negligible probability, i.e. for any polynomial time algorithm $\mathcal{A}$, $\mathcal{A}\left(\mathcal{P}\right) \approx_s \mathcal{A}\left(\mathcal{Q}\right)$. 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': $\quad \Pr(X = X') \leq 2^{-H_\infty(X)}$

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 $\mathcal{Y}$,

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

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

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

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 $\Sigma = S^t S$ where $S \in \mathbb{R}^{m \times n}$ is a rank-n matrix, is defined as:

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

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 C (a lattice Λ or a coset $\Lambda + \mathbf{x}$) is defined as $\forall \mathbf{x} \in C$, $\mathcal{D}_{C,S}(\mathbf{x}) = \frac{\rho_S(\mathbf{x})}{\rho_S(C)}$.

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 $\mathcal{D}_{\Lambda^\perp_{\mathbf{v}}(\mathbf{G}),\gamma}$ with parameter $\gamma = \tilde{O}(1)$.*

In addition, 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 rank-$m$ matrix $S \in \mathbb{R}^{k \times m}$, 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]). *For any rank-m lattice Λ, $0 < \varepsilon < \frac{1}{2}$, vector $\mathbf{c} \in \mathbb{R}^m$, and rank-$m$ matrix $S_1 \in \mathbb{R}^{k_1 \times m}$ and $S_2 \in \mathbb{R}^{k_2 \times m}$. If* $\min\left(\sigma_m\left(\sqrt{{S_1}^t S_1 + {S_2}^t S_2}\right), \sigma_m\left(\sqrt{\left(\left({S_1}^t S_1\right)^{-1} + \left({S_2}^t S_2\right)^{-1}\right)^{-1}}\right)\right) \geq \eta_\varepsilon(\Lambda)$,

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

where $\mathbf{y}_1 \leftarrow \mathcal{D}_{\Lambda+\mathbf{c},S_1}, \mathbf{y}_2 \leftarrow \mathcal{D}_{\Lambda,S_2}$, and $\mathbf{y}' \leftarrow \mathcal{D}_{\Lambda+\mathbf{c},\sqrt{{S_1}^t S_1 + {S_2}^t S_2}}$

Lemma 7 ([KLSS23, Lemma 5]). *For any rank-m lattice Λ, $\varepsilon > 0$ and rank-m matrix $S \in \mathbb{R}^{k \times m}$, $\sigma_m(S) \geq \eta_\varepsilon(\Lambda)$ if $\left\|\left(S^t S\right)^{-1}\right\|_2 \leq \eta_\varepsilon(\Lambda)^{-2}$.*

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 8. *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 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 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 Sanitize algorithm, it is easy to construct an FHE scheme that is circuit-private for all functions, by running Sanitize at the end of the evaluation procedure.

In this work, we also consider the relaxed variants of circuit-privacy and sanitization, where the closeness holds only computationally, instead of statistically.

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 hardness of the decision LWE problem [Reg05], or equivalently of its search counterpart:

- The decision LWE problem, parameterized 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: RLWE — a variant of LWE using ring elements — and RGSW.

A RLWE 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 $\mathsf{RLWE}_{\tilde{\mathbf{s}}}(\mu)$ 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 $\mathsf{RLWE}_{\tilde{\mathbf{s}},\vartheta}(\mu)$ a fresh RLWE encryption of μ under secret key $\tilde{\mathbf{s}}$ with noise variance ϑ. For $\mathbf{c} \in \mathsf{RLWE}_{\tilde{\mathbf{s}}}(\mu)$, 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 hardness assumption of the RLWE decision problem, the ring variant of the LWE decision problem [LPR10].

A RGSW 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 RLWE encryption of 0. The secret key is the same as for RLWE and the matrix $\mathbf{G}$ is defined as in Section 2.1. We note $\mathsf{RGSW}_{\tilde{\mathbf{s}},\vartheta}(\mu)$ a fresh RGSW encryption of μ under secret key $\tilde{\mathbf{s}}$ with noise variance ϑ. More visually, we have:

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

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.

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 A.1. This allows to multiply a RGSW ciphertext $\mathbf{C}$ and a RLWE 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 RGSW encryption $\mathbf{C}$ of the selector bit, and RLWE 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 explicitly defined.

- Initialization: n-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 initializes a test vector, $\mathsf{testv} \stackrel{def}{=} v(X) \in \hat{R}_q$, $v(X) = \sum_{i=0}^{N-1} v_i X^i$ and defines a noiseless RLWE encryption $(\mathbf{0}, \mathsf{testv})$. The test vector coefficients are chosen such that, after the BlindRotate, one can retrieve an LWE ciphertext of the input message as the constant coefficient of the RLWE ciphertext.
- BlindRotate: $\mathbb{Z}_{2N}^{n+1} \times \hat{R}_q \times \mathsf{RGSW}^n \to \mathsf{RLWE}$
  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 $\mathrm{BK}_i = \mathsf{RGSW}_{\tilde{s}, \vartheta_{\mathrm{BK}}}(s_i)$ for $i \in [n]$, the BlindRotate algorithm returns a RLWE encryption of $\mathsf{testv} \cdot X^{\bar{\varphi}}$ where $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} s_i \bar{a}_i$.
- Extract: RLWE $\to dN$-LWE
  By interpreting a RLWE 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. $v_{i+N} = -v_i$, Extract extracts an LWE ciphertext under key $K = \mathsf{KeyExtract}(\tilde{\mathbf{s}})$ of dimension dN of the constant term of μ, $\mu(0)$.
- KeySwitch: dN-LWE $\times$ $(n$-LWE$)^{dNt} \to n$-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 $\mathbf{s} \in \mathbb{B}^n$, 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 RLWE problem, but also on a circular assumption that states having LWE and RLWE encryptions of each other's secret key has no impact on security.

3 Randomization over rings

In this section we discuss the randomization of RLWE ciphertexts, an essential tool for reaching circuit privacy or sanitization properties. These results are of independent interest and can be used directly in other works, such as [DS16]. We present two solutions: the first one reaches the desired properties in a statistical sense, i.e. with security against unbounded adversaries. However, the parameters are unrealistic for an efficient implementation. The second one yields a very efficient randomization technique, but only holds under the hardness of the decision RLWE problem.

3.1 Gaussian Lemma over rings

We first start with the following lemma on discrete Gaussians, which is a generalization of the result of [BdMW16] to the ring setting, and to arbitrary lattices and Gaussian distributions that may not be spherical. It will be used both for proving results over $\Lambda^\perp(\mathbf{G})$, and over $\frac{1}{q}R$.

Lemma 9. *Let $\varepsilon > 0$, $\Lambda_1 \subset R^{d_1}$ be a rank-d_1N lattice, $S_1 \in \mathbb{R}^{k_1 \times d_1 N}$ be a rank-d_1N matrix, $S_2 \in \mathbb{R}^{k_2 \times N}$ be a rank-N matrix. For any $\mathbf{e} = (e_1, \ldots, e_{d_1}) \in \hat{R_q}^{d_1}$ and any $\mathbf{c} \in \hat{R_q}^{d_1}$, let $\mathbf{E}_i$ be the matrix such that $\mathtt{pow}_X \cdot \mathbf{E}_i = \mathtt{pow}_X \otimes e_i$ and $\mathbf{E} = (\mathbf{E}_0 \mid \mathbf{E}_1 \mid \ldots \mid \mathbf{E}_{d_1})$, then*
if $\min(\sigma_{d_1N}(S_1), \sigma_N(qS_2)) \geq (1 + q\|\mathbf{e}\|_2)\lambda_{d_1N}(\Lambda) \cdot \sqrt{\frac{\ln(2d_1N(1+1/\varepsilon))}{\pi}}$, we have

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

where $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda_1 + \mathbf{c}, S_1}$, $y \leftarrow \mathcal{D}_{\frac{1}{q}R, S_2}$, and $e' \leftarrow \mathcal{D}_{\frac{1}{q}R, \Gamma}$, with

$$\Gamma = \sqrt{S_2^t S_2 + \mathbf{E} S_1^t S_1 \mathbf{E}^t}.$$

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 $\mathbf{c} = (c_{1,0}, c_{1,1}, \ldots, c_{1,N-1}, c_{2,0}, \ldots, c_{d_1,N-1}) \in \frac{1}{q}\mathbb{Z}^{d_1N}$. We define the following notations in order to better explain the intuition behind the proof:

- $\widehat{\mathbf{E}} = \left(\mathbf{E} \;\middle|\; \frac{1}{q}\mathbf{I}_N\right) \in \frac{1}{q}\mathbb{Z}^{N \times N(d_1+1)}$;
- $\beta = \begin{pmatrix} S_1 & \mathbf{0} \\ \mathbf{0} & qS_2 \end{pmatrix} \in \mathbb{R}^{k_1+k_2 \times N(d_1+1)}$;
- $\widehat{\mathbf{c}} = (\mathbf{c}, 0, \ldots 0) \in \frac{1}{q}\mathbb{Z}^{N(d_1+1)}$;
- $\widehat{\Lambda} = \Lambda_1 \times \mathbb{Z}^N \subset \mathbb{Z}^{N(d_1+1)}$.

We want to show that:

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

Both distributions have the same support : $\frac{1}{q}\mathbb{Z}^N$. Let us analyze the probability mass assigned to each element $\mathbf{z}$ of $\frac{1}{q}\mathbb{Z}^N$. We have

$$\left(\widehat{\mathbf{E}}\mathcal{D}_{\widehat{\Lambda}+\widehat{\mathbf{c}},\beta}\right)(\mathbf{z}) = \frac{\rho_\beta(\mathcal{L}_\mathbf{z})}{\sum \rho_\beta(\mathcal{L}_{\mathbf{z}'})}, \quad \text{with } \mathcal{L}_\mathbf{z} = \left\{\mathbf{v} \in \widehat{\Lambda} + \widehat{\mathbf{c}} \mid \widehat{\mathbf{E}}\mathbf{v} = \mathbf{z}\right\},$$

where the sum ranges over all cosets $\mathcal{L}_{\mathbf{z}'}$ of $\mathcal{L}_\mathbf{0}$, when $\mathbf{z}'$ ranges over $\frac{1}{q}\mathbb{Z}^N$.
The idea is to show that $\rho_\beta(\mathcal{L}_\mathbf{z})$ is close to being proportional to $\rho_{\sqrt{\widehat{\mathbf{E}}\beta^t\beta\widehat{\mathbf{E}}^t}}(\mathbf{z})$.

$$\rho_{\sqrt{\widehat{\mathbf{E}}\beta^t\beta\widehat{\mathbf{E}}^t}}(\mathbf{z}) = \rho(\mathbf{u}_\mathbf{z}), \text{ with } \mathbf{u}_\mathbf{z} = \beta\widehat{\mathbf{E}}^t\left(\widehat{\mathbf{E}}\beta^t\beta\widehat{\mathbf{E}}^t\right)^{-1}\mathbf{z} = \beta\left(\beta^t\beta\right)^{-1}\beta^t\mathbf{u}_\mathbf{z} \in \mathbb{R}^{k_1+k_2}.$$

$$\text{For any } \mathbf{t} \in \mathcal{L}_\mathbf{z},\ \rho_\beta(\mathbf{t}) = \rho(\mathbf{u}_\mathbf{t}), \text{ with } \mathbf{u}_\mathbf{t} = \beta\left(\beta^t\beta\right)^{-1}\mathbf{t} \in \mathbb{R}^{k_1+k_2}.$$

We have $\mathbf{u}_\mathbf{z}^t\mathbf{u}_\mathbf{t} = \mathbf{u}_\mathbf{z}^t\mathbf{u}_\mathbf{z}$, so $\mathbf{u}_\mathbf{z}$ is orthogonal to $\mathbf{u}_\mathbf{t} - \mathbf{u}_\mathbf{z}$, which gives the identity

$$\begin{aligned}
\rho(\mathbf{u}_\mathbf{t}) &= \rho(\mathbf{u}_\mathbf{z}) \cdot \rho(\mathbf{u}_\mathbf{t} - \mathbf{u}_\mathbf{z}) \\
\rho_\beta(\mathbf{t}) &= \rho_{\sqrt{\widehat{\mathbf{E}}\beta^t\beta\widehat{\mathbf{E}}^t}}(\mathbf{z}) \cdot \rho_\beta\left(\mathbf{t} - \beta^t\mathbf{u}_\mathbf{z}\right) \\
\rho_\beta(\mathcal{L}_\mathbf{z}) &= \rho_{\sqrt{\widehat{\mathbf{E}}\beta^t\beta\widehat{\mathbf{E}}^t}}(\mathbf{z}) \cdot \rho_\beta\left(\mathcal{L}_\mathbf{0} + \mathbf{t} - \beta^t\mathbf{u}_\mathbf{z}\right)
\end{aligned}$$

We have that $\mathcal{L}_{\mathbf{0}}$ is the sublattice of $\widehat{\Lambda}$ that is orthogonal to $\widehat{\mathbf{E}}$, with $\widehat{\Lambda}$ of dimension $N(d_1+1)$ and $\widehat{\mathbf{E}}$ of rank N, so $\mathcal{L}_{\mathbf{0}}$ is of dimension $d_1 N$.
Let $\mathbf{b}_1, \ldots, \mathbf{b}_{d_1 N}$ be a set of $d_1 N$ independent vectors of Λ_1 of norm bounded by $\lambda_{d_1 N}(\Lambda)$.
Then, $(\mathbf{b}_1, -q\mathbf{E}\mathbf{b}_1), \ldots, (\mathbf{b}_{d_1 N}, -q\mathbf{E}\mathbf{b}_{d_1 N})$, is a set of $d_1 N$ independent vectors of $\mathcal{L}_{\mathbf{0}}$ of norm bounded by $(1+q\left\|\mathbf{e}\right\|_2)\lambda_{d_1 N}(\Lambda)$. Using lemma 2 gives

$$\eta_\varepsilon(\mathcal{L}_{\mathbf{0}}) \leq (1+q\left\|\mathbf{e}\right\|_2)\lambda_{d_1 N}(\Lambda) \cdot \sqrt{\frac{\ln\left(2d_1 N\left(1+1/\varepsilon\right)\right)}{\pi}}.$$

And by lemma 5, $\rho_\beta(\mathcal{L}_{\mathbf{0}} + \mathbf{t} - \beta^t \mathbf{u}_{\mathbf{z}}) \in \left[\frac{1-\varepsilon}{1+\varepsilon},\ 1\right] \cdot \rho_\beta(\mathcal{L}_{\mathbf{0}})$.

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

3.2 Randomized decomposition

We now present one of the essential tools for randomizing RLWE ciphertexts: the randomized decomposition that allows us to randomize the noise of a ciphertext.

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 entry 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* $\mathbf{G}_r^{-1}(\cdot)$*.*

In the following, we denote $\vartheta_\boxdot$ the variance of a Gaussian of parameter r. We have $\vartheta_\boxdot = \frac{r^2}{2\pi q^2}$. We give the full details of the randomized decomposition in supplementary materials Section A.2. The following lemmata give bounds on the noise propagation when using the randomized external product, which is an adaptation of [CGGI20, Corollary 3.14].

Lemma 10 (Variance of randomized External Product). *For any* $\mu \in \hat{R}_q$*,* $\mathbf{c} \in \mathsf{RLWE}_{\tilde{\mathbf{s}}}(\mu)$ *with* $\left\|\mathsf{Var}(\mathbf{c})\right\|_2 = \vartheta_{\mathbf{c}}$*, and* $\mathbf{C} = \mathsf{RGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathbf{C}}}(0)$*. Then* $\mathbf{C} \boxdot_r \mathbf{c} \in \mathsf{RLWE}_{\tilde{\mathbf{s}}}(0)$ *and*

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

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

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

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

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

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

We conclude using Lemma 10. □

The following corollary instantiates lemma 9 with the notations of our randomized decomposition.

Corollary 1. *For any $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{(d+1)\cdot\ell}$ and any $\mathbf{c} \in \hat{R}_q^{(d+1)\cdot\ell\cdot N}$, if $\min(\vartheta_{\boxdot}, \vartheta_y) \geq (\frac{1}{q} + \left\|\mathbf{e}\right\|_2)^2\left(1 + B^2\right) \cdot \frac{\ln(2\cdot(d+1)\cdot\ell\cdot N\cdot(1+1/\varepsilon))}{2\pi^2}$, we have:*

$$\Delta\left(\mathbf{e}^t\mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},q\sqrt{2\pi\vartheta_{\boxdot}}} + \mathcal{D}_{\frac{1}{q}R,\sqrt{2\pi\vartheta_y}}, \mathcal{D}_{\frac{1}{q}R,\Gamma}\right) \leq 2\varepsilon,$$

$$\text{with } \Gamma = \sqrt{\frac{\vartheta_y}{2\pi}\mathbf{I}_N + \frac{\vartheta_{\boxdot}}{2\pi}q^2 \sum_{i=1}^{(d+1)\cdot\ell} \mathbf{E}_i^t\mathbf{E}_i}.$$

3.3 Leftover Hash Lemma over rings

The following lemma shows that combining the randomized decomposition with an additional small Gaussian noise is enough to randomize RLWE ciphertexts as long as the parameters are large enough. It combines lemma 9 and a variant of the Leftover Hash Lemma adapted to $\hat{R}_q$ and its particular structure. We give a detailed proof in Appendix B.

Lemma 12. *Let $\epsilon_\ell, \epsilon, r > 0$. Under the following conditions:*

- $\ell > \frac{1}{d+1}\left(\log\frac{(Nk-1)(1+\varepsilon)}{4\epsilon_\ell^2(1-\varepsilon)} + d\right)$
- $r \geq \sqrt{(2+2B^2)\cdot\frac{\ln(2N(d+1)\cdot\ell(1+1/\varepsilon))}{\pi}}$,

for any $\mathbf{e} \in \hat{R_q}^{(d+1)\cdot\ell}$, $\mathbf{y} \in \hat{R_q}$, $\mathbf{c} \in \hat{R_q}^{(d+1)\cdot\ell}$:

$$\Delta\Big((\mathbf{A}, \mathbf{x}^t \cdot \mathbf{A}, \mathbf{e}^t\mathbf{x} + \mathbf{y}), (\mathbf{A}, \mathbf{u}^t, \mathbf{e}^t\mathbf{x} + \mathbf{y})\Big) < \varepsilon_\ell$$

where $\mathbf{A}$ *is uniform over* $\hat{R_q}^{(d+1)\cdot\ell\times d}$, $\mathbf{x} \leftarrow \mathcal{D}_{\Lambda^\perp(\mathbf{G})+\mathbf{c},r}$, $\mathbf{u}$ *is uniform over* $\hat{R_q}^{d}$.

3.4 Using a sanitization key

In case the bounds required by Lemma 12 cannot be met, we propose a second solution: adding a public key encryption of zero to the ciphertext. This technique allows for practical schemes, but the randomization guarantees rely on the hardness of the decision RLWE problem.
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 RLWE samples at hand.
In order for our result to hold, we require the following properties on the public encryption scheme to be used, and present a particular construction that satisfies those conditions. Intuitively, we require that the ciphertext looks uniform to an adversary, even knowing the secret key, except for the noise and the message.

Lemma 13 (Sanitization key). *There exist two algorithms* PkGen *and* PkEnc *such that:*

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

$\mathsf{PkEnc}(\mathrm{PK})$*: for any* RLWE *secret key* $\tilde{\mathbf{s}}$*, the output of* PkEnc *satisfies:*

$$(\mathrm{PK}, \mathsf{PkEnc}(\mathrm{PK})) \approx_c (\mathrm{PK}, (\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}} + e_u))$$

where $\mathbf{u}$ *is uniform over* R_q^d*,* e_u *has a variance bounded by some value* ϑ_{PK} *with overwhelming probability and* PK *is honestly generated from* $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$*.*

The distribution of e_u is not important for the sanitization property but it has to remain small for correctness.

One possible solution that reaches the statistical closeness would be to use a public key a la Regev, with PK being a set of $\omega\left((d+1)N\log(q)\right)$ encryptions of 0, and $\mathsf{PkEnc}\,(\mathrm{PK})$ being the sum of a random subset of those. However, the size of the public key and the computational complexity to generate it will be prohibitive in practice. Instead, we leverage the RLWE decisional assumption to reduce the size of PK and the time to generate $\mathsf{PkEnc}\,(\mathrm{PK})$ to almost negligible. We believe that the analysis of the distribution of ciphertexts produced by $\mathsf{PkEnc}\,(\mathrm{PK})$, its generalization to higher dimension and its application are of independent interest. For example, the public key can be used as a randomizer in combination with the soak-and-spin technique of [DS16] to obtain a better key size, efficiency and noise management trade-offs. More details are given in Section 5.1.

Construction 1 *We exhibit a concrete example of a construction satisfying the property of Lemma 13.*

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

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

  where $\mathbf{A}$ *is uniform over* $\hat{R_q}^{d' \times d}$*, and each coefficient of* $\mathbf{e} \in \hat{R_q}^{d'}$ *has coefficients sampled from* $\chi_{\vartheta_\mathbf{e}}$*, for a noise variance* $\vartheta_\mathbf{e}$*.*
- $\mathsf{PkEnc}(\text{PK})$*: on input public key* PK*, returns*

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

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

Lemma 14. *Under the hardness assumption of the* RLWE *decision problem with noise variance* ϑ*, where* $\frac{1}{\vartheta} = 2\left(\min\left(\frac{1}{\vartheta_\mathbf{r}}, \frac{1}{\vartheta_{\mathbf{e}'}}\right) + \frac{1}{\vartheta_{e''}}\left(\|q\mathbf{e}\|_2^2 + \|\tilde{\mathbf{s}}\|_2^2\right)\right)$ *as long as the following condition is met:*

$$\vartheta \geq \frac{1}{\pi}\eta_\varepsilon\left(\frac{1}{q}\mathbb{Z}^{(d'+d)N}\right)^2$$

Construction 1 satisfies the property of Lemma 13, with

$$\vartheta_{\text{PK}} = \vartheta_{e''} + q^2\vartheta_\mathbf{r}\|\mathbf{e}\|_2^2 + \|\tilde{\mathbf{s}}\|_2^2\,\vartheta_{\mathbf{e}'}.$$

To prove Lemma 14, we will use the following result, which is an adaptation of [KLSS23, Lemma 7] to our notations. We provide a detailed proof in Appendix C for the sake of completeness.

Lemma 15. *For any* $\tilde{\mathbf{s}} \in R^d, \mathbf{e} \in \hat{R_q}^{d'}$*, let* $\tilde{\mathbf{S}}_i$ *be the matrix such that* $\mathtt{pow}_X \cdot \tilde{\mathbf{S}}_i = \mathtt{pow}_X \otimes \tilde{s}_i$*, and* $\mathbf{E}_i$ *be the matrix such that* $\mathtt{pow}_X \cdot \mathbf{E}_i = \mathtt{pow}_X \otimes e_i$*. We define* $\Gamma = \left(q{\mathbf{E}_1}^t \;\middle|\; \ldots \;\middle|\; q{\mathbf{E}_{d'}}^t \;\middle|\; -\tilde{\mathbf{S}}_1^t \;\middle|\; \ldots \;\middle|\; -\tilde{\mathbf{S}}_d^t\right)$*. Note that* $\mathtt{pow}_X \cdot \Gamma = \mathtt{pow}_X \otimes (q\mathbf{e}^t \mid -\tilde{\mathbf{s}}^t)$*. We note*

$$\Sigma = \begin{pmatrix} 2\pi\vartheta_\mathbf{r}\mathbf{I}_{d'N} & \mathbf{0} \\ \mathbf{0} & 2\pi\vartheta_{\mathbf{e}'}\mathbf{I}_{dN} \end{pmatrix}, \qquad \Sigma_\mathbf{v} = \left(\Sigma^{-1} + \frac{1}{2\pi\vartheta_{\mathbf{e}''}}\Gamma^t\Gamma\right)^{-1}.$$

$$\Delta\Big(\left(\mathbf{v}, \mathbf{re} - \mathbf{e}'\tilde{\mathbf{s}} + e''\right), \left(\mathbf{v}_0 + \mathbf{v}^*, \mathbf{re} - \mathbf{e}'\tilde{\mathbf{s}} + e''\right)\Big) = 0,$$

where $\mathbf{r} \leftarrow \mathcal{D}_{R^{1\times d'}, q\sqrt{2\pi\vartheta_\mathbf{r}}}$*,* $\mathbf{e}' \leftarrow \mathcal{D}_{\frac{1}{q}R^{1\times d}, \sqrt{2\pi\vartheta_{\mathbf{e}'}}}$*,* $e'' \leftarrow \mathcal{D}_{\frac{1}{q}R, \sqrt{2\pi\vartheta_{e''}}}$*,* $\mathbf{v} = \binom{\frac{1}{q}\mathbf{r}^t}{\mathbf{e}'^t}$*,* $\mathbf{v}_0 = \frac{1}{2\pi\vartheta_{e''}}\Sigma_\mathbf{v}\binom{q\mathbf{e}}{-\tilde{\mathbf{s}}}\left(\mathbf{re} - \mathbf{e}'\tilde{\mathbf{s}} + e''\right)$ *and* $\mathbf{v}^* \leftarrow \mathcal{D}_{-\mathbf{v}_0 + \frac{1}{q}R^{d'+d}, \sqrt{\Sigma_\mathbf{v}}}$*.*

We are now ready to prove 14:

Proof. For any secret key $\tilde{\mathbf{s}} \in R^d$, and any $\mathbf{e} \in \hat{R_q}^{d'}$, let $\tilde{\mathbf{S}}_i$ (resp. $\mathbf{E}_i$) be the matrix such that $\mathtt{pow}_X \cdot \tilde{\mathbf{S}}_i = \mathtt{pow}_X \otimes \tilde{s}_i$ (resp. $\mathtt{pow}_X \cdot \mathbf{E}_i = \mathtt{pow}_X \otimes e_i$). We set

$$\Gamma = \left(q{\mathbf{E}_1}^t \middle| \ldots \middle| q{\mathbf{E}_{d'}}^t \middle| -\tilde{\mathbf{S}}_1^t \middle| \ldots \middle| -\tilde{\mathbf{S}}_d^t\right), \Sigma_{\mathbf{v}}^{-1} = \frac{1}{2\pi\vartheta_{\mathbf{e}''}}\Gamma^t\Gamma + \frac{1}{2\pi}\begin{pmatrix} \vartheta_{\mathbf{r}}\mathbf{I}_{d'N} & \mathbf{0} \\ \mathbf{0} & \vartheta_{\mathbf{e}'}\mathbf{I}_{dN} \end{pmatrix}^{-1}.$$

Lemma 15 gives us:

$$(\mathbf{r}, \mathbf{e}', \mathbf{re} - \mathbf{e}'\tilde{\mathbf{s}} + e'') = \left(\mathbf{r}_0 + \mathbf{r}^*, \mathbf{e}'_0 + \mathbf{e}'^*, e_u\right), \tag{2}$$

with $e_u = \mathbf{re} - \mathbf{e}'\tilde{\mathbf{s}} + e''$,

$$\mathbf{v}_0 = \begin{pmatrix} \frac{1}{q}{\mathbf{r}_0}^t \\ {\mathbf{e}'_0}^t \end{pmatrix} = \frac{1}{2\pi\vartheta_{e''}}\Sigma_{\mathbf{v}}\begin{pmatrix} q\mathbf{e} \\ -\tilde{\mathbf{s}} \end{pmatrix} e_u, \quad \mathbf{v}^* \leftarrow \mathcal{D}_{-\mathbf{v}_0 + \frac{1}{q}R^{d'+d}, \sqrt{\Sigma_{\mathbf{v}}}}$$

Next, we can split $\mathbf{v}^*$ into $\mathbf{v}^\dagger + \widehat{\mathbf{v}}$, with $\mathbf{v}^\dagger \leftarrow \mathcal{D}_{R^{d'+d}, \sqrt{2\pi\vartheta}}$ and $\widehat{\mathbf{v}} \leftarrow \mathcal{D}_{-\mathbf{v}_0 + \frac{1}{q}R^{d'+d}, \sqrt{\widehat{\Sigma}}}$, with $\widehat{\Sigma} = \Sigma_{\mathbf{v}} - 2\pi\vartheta\mathbf{I}_{(d'+d)N}$, by lemma 6, since :
$\widehat{\Sigma}$ is positive definite because $\sigma_{(d'+d)N}\left(\Sigma_{\mathbf{v}}\right)$ is at least

$$\left(\min\left(\frac{1}{2\pi\vartheta_{\mathbf{r}}}, \frac{1}{2\pi\vartheta_{\mathbf{e}'}}\right) + \frac{1}{2\pi\vartheta_{e''}}\left(\|q\mathbf{e}\|_2^2 + \|\tilde{\mathbf{s}}\|_2^2\right)\right)^{-1} \geq 4\pi\vartheta > 2\pi\vartheta, \text{ and}$$

$$\left\|\left(2\pi\vartheta\mathbf{I}_{(d'+d)N}\right)^{-1} + \widehat{\Sigma}^{-1}\right\|_2 = \frac{1}{2\pi\vartheta} + \sigma_{(d'+d)N}\left(\widehat{\Sigma}\right)^{-1} \leq \frac{1}{\pi\vartheta} \leq \eta_\varepsilon\left(\frac{1}{q}\mathbb{Z}^{(d'+d)N}\right)^{-2}$$

which implies that $\sigma_{(d'+d)N}\left(\left(\left(2\pi\vartheta\mathbf{I}_{(d'+d)N}\right)^{-1} + \widehat{\Sigma}^{-1}\right)^{-1}\right) \geq \eta_\varepsilon\left(\frac{1}{q}\mathbb{Z}^{(d'+d)N}\right)$.

We can then conclude by invoking the hardness of the decision RLWE problem. To sum up, we have the following hybrid argument

$$\begin{aligned}
\mathrm{PK}, \mathsf{PkEnc}(\mathrm{PK}) &= \mathrm{PK}, (\mathbf{rA} + \mathbf{e}' \mid (\mathbf{r}(\mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}) + e'') \\
&= \mathrm{PK}, (\mathbf{rA} + \mathbf{e}' \mid \mathbf{rA}\tilde{\mathbf{s}} + \mathbf{re} + e'') \\
&= \mathrm{PK}, \Big(\left(\mathbf{r}_0 + \mathbf{r}^*\right)\mathbf{A} + \mathbf{e}'_0 + \mathbf{e}'^* \mid && \text{(by 2)} \\
&\qquad \left(\left(\mathbf{r}_0 + \mathbf{r}^*\right)\mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}'_0 + \mathbf{e}'^*\right)\tilde{\mathbf{s}} + e_u\Big) \\
&\approx_s \mathrm{PK}, \Big(\left(\mathbf{r}_0 + \widehat{\mathbf{r}}\right)\mathbf{A} + \mathbf{e}'_0 + \widehat{\mathbf{e}'} + \mathbf{r}^\dagger\mathbf{A} + \mathbf{e}'^\dagger \mid \text{(by Lemma 6)} \\
&\qquad \left(\left(\mathbf{r}_0 + \widehat{\mathbf{r}}\right)\mathbf{A} + \mathbf{e}'_0 + \widehat{\mathbf{e}'} + \mathbf{r}^\dagger\mathbf{A} + \mathbf{e}'^\dagger\right)\tilde{\mathbf{s}} + e_u\Big) \\
&\approx_c \mathrm{PK}, \Big(\left(\mathbf{r}_0 + \widehat{\mathbf{r}}\right)\mathbf{A} + \mathbf{e}'_0 + \widehat{\mathbf{e}'} + \mathbf{u} \mid && \text{(RLWE)} \\
&\qquad \left(\left(\mathbf{r}_0 + \widehat{\mathbf{r}}\right)\mathbf{A} + \mathbf{e}'_0 + \widehat{\mathbf{e}'} + \mathbf{u}\right)\tilde{\mathbf{s}} + e_u\Big) \\
&= \mathrm{PK}, (\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}} + e_u),
\end{aligned}$$

The bound on ϑ_{PK} is obtained by bounding the variance of $\mathbf{re} + e'' - \mathbf{e}'\tilde{\mathbf{s}}$, using Lemma 3 to bound $\|\mathbf{r}\|_2$. □

3.5 Randomization of RLWE ciphertexts

Our procedure to scale and randomize a RLWE ciphertext consists of a randomized decomposition, and the addition of a randomizer RERAND. In order to clearly expose the two strategies, we define:

- RERAND $= (\mathbf{0}, y)$ and $\approx$ denotes $\approx_s$, if $\ell > \frac{1}{d+1}\left(\log \frac{(Nk-1)(1+\varepsilon)}{4\epsilon_\ell^2(1-\varepsilon)} + d\right)$;
- RERAND $= (\mathbf{0}, y) + \mathsf{PkEnc}(\mathrm{PK})$ and $\approx$ denotes $\approx_c$ otherwise,

with $y \leftarrow \mathcal{D}_{\frac{1}{q}R,\sqrt{2\pi\vartheta_\boxdot}}$ and PK honestly generated from $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$.

The following lemma states that this randomization technique brings any encryption of zero close to a canonical distribution.

Lemma 16. *For any* $\tilde{\mathbf{s}} \in \hat{R}_q^{\,d}$*, any* $\mathbf{e} = (e_1, \ldots, e_{(d+1)\cdot\ell}) \in \hat{R}_q^{\,(d+1)\cdot\ell}$*, and any* $\mathbf{v} \in \hat{R}_q^{\,d+1}$*, we have:*

$$\text{RERAND} + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{v} \approx \text{RERAND} + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{0},$$

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

Proof. Assuming RERAND $= (\mathbf{0}, y)$, with $y \leftarrow \mathcal{D}_{\frac{1}{q}R,\sqrt{2\pi\vartheta_\boxdot}}$, we have:

$$\begin{aligned}
\text{RERAND} + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{v} = \ & \left(\mathbf{A} \boxdot_r \mathbf{v} \mid (\mathbf{A} \boxdot_r \mathbf{v})\tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y\right) \\
\approx_s \ & \left(\mathbf{u} \mid \mathbf{u} \cdot \tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y\right) \text{ by Lemma 12} \\
\approx_s \ & \left(\mathbf{u} \mid \mathbf{u} \cdot \tilde{\mathbf{s}} + e'\right) \text{ by Corollary 1,}
\end{aligned}$$

where $\mathbf{u}$ is uniform over $\hat{R}_q^{\,1\times d}$ and $e' \leftarrow \mathcal{D}_{\frac{1}{q}R,\Gamma}$ for some Γ given by Corollary 1.

Assuming RERAND $= (\mathbf{0}, y) + \mathsf{PkEnc}(\mathrm{PK})$, with $y \leftarrow \mathcal{D}_{\frac{1}{q}R,\sqrt{2\pi\vartheta_\boxdot}}$ and PK honestly generated from $\mathsf{PkGen}(1^\lambda, \tilde{\mathbf{s}})$, by Lemma 13, we have:

$$\begin{aligned}
\text{RERAND} + \left(\mathbf{A} \mid \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}\right) \boxdot_r \mathbf{v} \approx_c \ & \left(\mathbf{u} + \mathbf{A} \boxdot_r \mathbf{v} \mid (\mathbf{u} + \mathbf{A} \boxdot_r \mathbf{v})\tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y + e_u\right) \\
= \ & \left(\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}} + \mathbf{e} \boxdot_r \mathbf{v} + y + e_u\right) \\
\approx_s \ & \left(\mathbf{u} \mid \mathbf{u}\tilde{\mathbf{s}} + e' + e_u\right) \text{ by Corollary 1,}
\end{aligned}$$

where $\mathbf{u}$ is uniform over $\hat{R}_q^{\,1\times d}$, the bound on the variance of e_u is given by Lemma 14, and $e' \leftarrow \mathcal{D}_{\frac{1}{q}R,\Gamma}$ for some Γ given by Corollary 1.

□

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 that it verifies the desired properties.

4.1 Circuit Private blind rotation

In order to achieve circuit privacy for the family of $\mathsf{BlindRotate}$ algorithms with hardwired inputs $(\bar{\mathbf{a}}, \bar{b})$, the main loop of TFHE $\mathsf{BlindRotate}$ (line 3 of Algorithm 1, which is a multiplexer operation) is replaced by its randomized counterpart. We show by induction that adding one fresh randomizer at the end of the computation is enough to hide all information about $(\bar{\mathbf{a}}, \bar{b})$ except for the result.

Algorithm 1 Private computation of a RLWE 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 RGSW encryption of s_i for $i \in [1, n]$, a fresh randomizer RERAND.

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

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

Lemma 17. *For any* $\mathsf{testv} \in \hat{R_q}$*, the encryption scheme of Section 2.6 is circuit private for the class of functions* $\big(\mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\big((\bar{\mathbf{a}}, \bar{b}), \cdot, \mathrm{RERAND}\big)\big)_{(\bar{\mathbf{a}}, \bar{b})}$*, if* RERAND *is a fresh randomizer as defined in Section 3.5.*

Proof. For $0 \leq t \leq n$, let us denote ACC_t the value of the accumulator ACC after t iterations of the loop, and ACC^* the value of ACC after adding the randomizer RERAND. For $1 \leq t \leq n$, after the t-th iteration we have:

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

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

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

For the sake of readability, we define the following shorthand notations for given $\bar{\mathbf{a}}$ and $\mathbf{s}$: $X^{\geq t}$ denotes X raised to the power $\displaystyle\sum_{i=t}^{n} s_i \bar{a}_i$. We prove by induction

that for t from n down to 0, the following holds:

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

For $t = n$, we have $\mathsf{ACC}^* = \mathsf{ACC}_n + \mathrm{RERAND}$.

Now assume that (4) holds at step $1 \leq t \leq n$, we show that the statement still holds at step $t-1$. Using (3), we have:

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

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

$$X^{\geq t+1}\left((\mathrm{BK}_t - s_t\mathbf{G}) \boxdot_r \mathbf{v}\right) = (\mathrm{BK}_t - s_t\mathbf{G}) \boxdot_r \left(X^{\geq t+1}\cdot\mathbf{v}\right)$$

We conclude using Lemma 16.
Finally, when $t = 0$, $X^{-\bar{b}}\cdot X^{\geq 0} = X^{\bar{\varphi}}$ and we have:

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

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

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

Lemma 18 (CP-BlindRotate noise propagation). *Let* $\mathrm{BK} = (\mathrm{BK}_i)_i$ *be a bootstrapping key where* $\mathrm{BK}_i = \mathsf{RGSW}_{\tilde{\mathbf{s}},\vartheta_{\mathrm{BK}}}(s_i)$ *for* $i \in [1, n]$. *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* $\mathbf{c} \in \mathsf{RLWE}_{\tilde{\mathbf{s}}}(\mathsf{testv}\cdot X^{\bar{\varphi}})$ *where* $\bar{\varphi} = \bar{b} - \sum_{i=1}^{n} s_i\bar{a}_i$, *and*

$$\|\mathsf{Var}(\mathbf{c})\|_2 \leq \vartheta_\boxdot + n2\pi q^2\vartheta_\boxdot(d+1)\ell N\vartheta_{\mathrm{BK}} + \vartheta_{\mathrm{PK}}$$

Proof. The bound comes from the fact that Algorithm 1 performs n homomorphic multiplexers, each adding at most $2\pi q^2\vartheta_\boxdot(d+1)\ell N\vartheta_{\mathrm{BK}}$ to the variance of the accumulator using Lemma 11, and one Gaussian noise element y of variance $\vartheta_\boxdot$ and public key of variance ϑ_{PK}. $\square$

4.2 Sanitizing algorithm

We are now ready to present our sanitization algorithm. It is basically the Bootstrap algorithm from TFHE, with the test vector testv chosen to evaluate the identity function, and where BlindRotate is replaced with CP-BlindRotate. It is described more formally in Algorithm 2.

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 RGSW encryption of s_i for $i \in [1, n]$, a fresh randomizer RERAND, 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: $\boxed{\mathbf{c}' = \mathsf{CP\text{-}BlindRotate}_{\mathsf{testv}}\left((\bar{\mathbf{a}}, \bar{b}), \text{BK}, \text{RERAND}\right)}$
5: $\mathbf{c} = (\mathbf{0}, \bar{\mu}) + \mathsf{Extract}(\mathbf{c}')$
6: Return $\mathsf{KeySwitch}(\text{KS}, \mathbf{c})$

Lemma 19. *Algorithm 2 is sanitizing for the encryption scheme described in Section 2.6.*

Proof. The simulator $\mathsf{Sim}_s(\mu, \text{BK}, \text{RERAND}, \text{KS})$ follows the same steps as the Sanitize algorithm except it computes $\mathbf{c'} = \mathsf{Sim}_{cp}(\mathsf{testv} \cdot X^{\mu}, \text{BK}, \text{RERAND})$, using the simulator Sim_{cp} of Lemma 17 instead of calling CP-BlindRotate.
To better understand why our statement stands, even though the input passed to Sim_{cp} does not seem to match the expected result from CP-BlindRotate, we need to further analyze Extract which is recalled in details in the supplementary materials, Section A.3 for the sake of completeness.
On input a RLWE 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:

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

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. $(\mathbf{a}, b) \in \mathcal{C}_{\mu}$. □

Lemma 20 (Sanitization noise propagation). *If the parameters are set correctly, Algorithm 2 is message-space preserving. More precisely, we have that the variance of its output has a norm bounded by*

$$\vartheta_{\boxdot} + n2\pi q^2 \vartheta_{\boxdot}(d+1)\ell N \vartheta_{\text{BK}} + \vartheta_{\text{PK}} + d \cdot N\left(\frac{\mathsf{B}_{\text{KS}}^{-2t}}{4} + t \cdot \frac{\mathsf{B}_{\text{KS}}^2}{4}\vartheta_{\text{KS}}\right)$$

Proof. This results combines Lemma 18 and Lemma 21. □

	B	ℓ	ϑ_{BK} ($\tilde{\mathbf{s}}$ and $\mathbf{e}$)	$\vartheta_{\mathbf{r}} = \vartheta_{\mathbf{e}'}$	$\vartheta_{e''}$	$\vartheta_{\boxdot}$
$q = 2^{35}$	32	7	$2^{-67.6}$	$2^{-65.6}$	$2^{-49.5}$	$2^{-39.4}$

Table 1. Parameters and secret keys and ciphertexts noise variances taking a polynomial of degree $N = 1024$.

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: The parameters are selected with respect to a careful trade-off between efficiency and security. We summarize in Table 1 the variance parameters we took with respect to the following bounds:

- RGSW ciphertexts: the bootstrapping key BK is made of n RGSW ciphertexts with $B = 32$, $\ell = 7$ and $d = 1$, with secret and noise variance $\vartheta_{\text{BK}} = 2^{-67.6}$.
- Corollary 1 gives a lower bound on the Gaussian parameter for $\mathbf{G}_r^{-1}()$. Taking the statistical advantage $\varepsilon = 2^{-80}$, we obtain $\vartheta_{\boxdot} = 2^{-39.4}$ where we bound the norm of $\mathbf{e}$ in the condition of Corollary 1 using Lemma 3.
- PkEnc(PK) - public key: with $d' = d = 1$ and $\vartheta_{\mathbf{e}} = \vartheta = \vartheta_{\text{BK}}$ to rely on the same hardness assumption as the RGSW ciphertexts, setting $\vartheta_{\mathbf{r}} = \vartheta_{\mathbf{e}'} = 4\vartheta_{\text{BK}}$ and $\vartheta_{e''} = 4\vartheta_{\text{BK}}\left(\|\tilde{\mathbf{s}}\|_2^2 + q^2\|\mathbf{e}\|_2^2\right)$ ensures minimal noise growth ϑ_{PK} while verifying the conditions of lemma 14. We again bound the norms using lemma 3.
- LWE ciphertexts: the keyswitching key KS is made of n LWE ciphertexts of dimension $n = 538$ and noise variance $2^{-27.2}$, base decomposition $\mathsf{B}_{\text{KS}} = 4$ and precision $t = 7$.

Correctness: In order to decrypt correctly, the amplitude of the error needs to be $< \frac{1}{4}$ (rather than $\frac{1}{16}$ for the gate bootstrapping) with overwhelming probability. With the parameters derived from Lemma 20, we obtain an LWE ciphertext with error standard deviation $\sigma \approx 0.024$, which corresponds to a probability of decryption failure below $\mathsf{erf}(\frac{1}{4\sqrt{2}}\sigma) \approx 2^{-82.7}$.

For LWE, RLWE and RGSW ciphertexts we use, the estimation based on the last commit of the Lattice Estimator [APS15] gives 100 bits of security.

Implementation: As a proof of concept of our technique, we provide a C implementation compatible with Windows, Linux, and MacOS and which is accessible from `https://github.com/fhe-extension/fhe-sanitize`. We ran multiple experiments using the parameters described previously and provide the results in Table 2. The randomness is sampled using the CSPRNG of windows or directly from /dev/urandom. We use the library FFTW [3] for our Fourier

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

Bootstrap -	Sanitize w/ pre-computation	Sanitize w/o pre-computation	$\mathsf{PkEnc}(\mathrm{PK})$ -
0.42 s	0.52 s	7.5 s	1 ms

Table 2. Timings of our concrete implementation. The experiments were run on a virtual machine of a 2.20GHz Intel Core i7-8750H laptop 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 pre-computed discrete Gaussian samples for $\mathbf{G}_r^{-1}(\cdot)$, the third one reports the timings including the timings of Gaussian sampling computation.

transformation. We implement the sampler for $\mathbf{G}_r^{-1}(\cdot)$ using the Gaussian sampler from [GPV08,MP12] and implement the inversion sampling for the base sampler from [MW17]. Since our goal is to evaluate the overhead of our sanitization technique, we compare the non-sanitizing bootstrapping (first column of Table 2) and the sanitizing bootstrapping (remaining columns of Table 2) with the same parameters on the same laptop, with the same level of optimizations[4].

We found that generating small Gaussian samples with our base sampler accounts for 80% of the total execution time of our implementation. We have decided to implement precomputation for these samples to assess the efficiency of our strategy with access to specialized hardware for Gaussian sampling, or to determine if these samples could be computed in parallel or before the online sanitization phase. In that case, sanitizing a FHE ciphertext with our strategy takes 0.52s, which is a small overhead compared to 0.42s for a non-sanitizing bootstrapping.

The following discussion shows that using the same set of parameters as ours, about 5 to 6 cycles of the soak-and-spin strategy from [DS16] would be needed to reach the same level of security. In order to derive concrete timings for this comparison, one could sum the timings from column 1 and 4 of Table 2 (and multiply it by the number of cycles required i.e. ≈ 5.7) but note that this should take into account that we use our optimized public key randomizer instead of the randomizer from [DS16].

5.1 Comparison with [DS16] sanitization strategy

In this section, we evaluate the cost of sanitizing a TFHE ciphertext using the strategy proposed in [DS16] with our parameter set. As in the analysis from [DS16]'s strategy applied to FHEW, proposed in [DS16, Section 4.3], our analysis estimates the cost of the [DS16] approach without considering the total probability of decryption failure. Note that such a choice would play in [DS16] strategy's advantage in this comparison.

Recall that [DS16] sanitization algorithm consists of the iterations of two steps: (1) a refresh step which, given an LWE ciphertext as input, brings the

[4] This explains our choice of implementing TFHE from scratch rather than using an existing implementation of the bootstrapping.

noise down to a low level; (2) a randomization step which randomizes a ciphertext by adding a linear combination of encryptions of 0 (which is analogous to our PkEnc procedure) and the addition of a uniform noise to decrease the statistical distance between two hypothetical ciphertexts.
As done in the [DS16] analysis applied to FHEW, for maximal efficiency, this randomization step is done right after the Extract but before switching to a lower dimension, so that the noise in the encryptions of 0 can be chosen smaller since we are using a larger dimension. The number of cycles in [DS16] strategy varies depending on the FHE scheme and 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.
Using our parameters, the first BlindRotate step outputs a LWE sample of standard deviation around $2^{-17.4}$. In order to randomize the left part of the LWE ciphertext output, either we add a linear combination of $\omega\left((d+1)N\log(q)\right)$ fresh encryptions of zero (after extraction and before keyswitching in order to have a lower noise propagation); each adds a noise of standard deviation $2^{-33.8}$. An other more efficient option would be to use our public randomizer which adds a noise of standard deviation of about 2^{-24}. We obtain that the noise introduced by the randomization step is negligible compared to the noise introduced by the BlindRotate step.
We first derive the noise induced by one partial cycle, and deduce the amount of soaking noise that can be added without compromising correctness. We obtain that the final standard deviation after one invocation of a partial cycle - which includes a non circuit-private blindrotation step, a randomization of the left part (which induces a negligible noise) and a keyswitching step - leads to a standard deviation of around 0.015. With probability similar than ours, the LWE sample has an error of amplitude less than $\frac{5}{32}$.

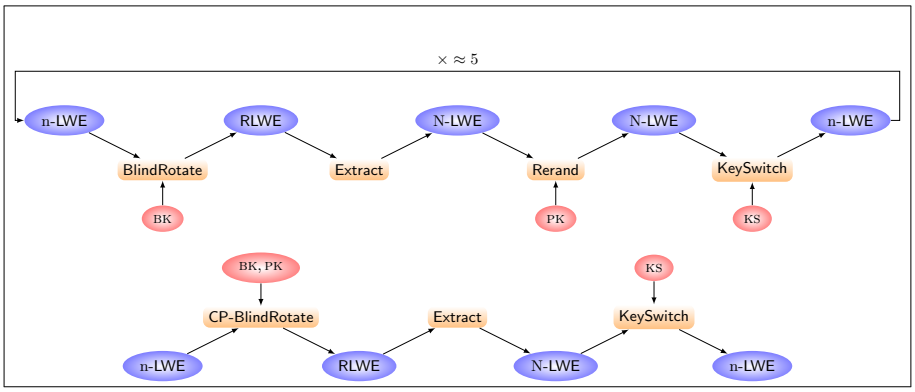

Fig. 2. High-level overview of our sanitization strategy (bottom part) and [DS16] strategy (top part).

Thus, the soaking noise amplitude B may be chosen up to around $\frac{3}{32} \approx 2^{-3.4}$ with almost the same decryption failure probability as ours. So one cycle reaches a statistical distance between LWE ciphertexts of $\frac{2^{-17.4}}{2^{-3.4}} = 2^{-14}$. In order to reach the same statistical distance as ours i.e. 2^{-80} , we would need around between 5 and 6 cycles. It can also be noted that our approach requires only one bootstrapping, reducing the probability of decryption failure obtained at the end. Figure 2 gives a high-level overview of both strategies to highlight the differences between them: assuming we perform the same randomization step, we perform them at each step of the $\mathsf{BlindRotate}$ algorithm, whereas the soak-and-spin strategy applies it only once per iteration of the bootstrapping, but requires between 5 to 6 such iterations to reach the same level of circuit privacy security.

References

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. *Journal of Mathematical Cryptology. Volume 9, Issue 3, Pages 169–203, ISSN (Online) 1862-2984*, October 2015.

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 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.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Fast fully homomorphic encryption library over the torus, 2016. `https://github.com/tfhe/tfhe`.

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.

DGKS21. Dana Dachman-Soled, Huijing Gong, Mukul Kulkarni, and Aria Shahverdi. Towards a ring analogue of the leftover hash lemma. *J. Math. Cryptol.*, 15(1):87–110, 2021.

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.

GPL23. Antonio Guimarães, Hilder V. L. Pereira, and Barry Van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 3–35. Springer, 2023.

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.

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-mlwe. In *43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 549–580. Springer, 2023.

Klu22. Kamil Kluczniak. Circuit privacy for FHEW/TFHE-style fully homomorphic encryption in practice. Cryptology ePrint Archive, Report 2022/1459, 2022. `https://eprint.iacr.org/2022/1459`.

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.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.

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.

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

MKMS22. Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. Efficient lattice-based inner-product functional encryption. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 163–193. Springer, Heidelberg, March 2022.

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.

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

Supplementary material

A TFHE Bootstrapping building blocks

A.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 $(x_{1,1} \mid \ldots \mid x_{d+1,\ell})$

A.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 $\Lambda^\perp(\mathbf{G})$.

A.3 Extract

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

A.4 KeySwitch

In Algorithm 6, we recall the keyswitching procedure of the bootstrapping, and the analysis of noise growth is provided in Lemma 21

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: $\quad x_i = \sum_{j=0}^{N-1} x_{i,j}X^j \in R$
6: Return $\left(x_1 \mid \ldots \mid x_{(d+1)\cdot\ell}\right)$

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

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

1: $b' = b(0)$.
2: **for** $i = 1$ **to** d
3: $\quad$ 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 $\mathbf{c}' = (\mathbf{a}', b')$.

Lemma 21 (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* $\mathbf{c}' \in n\text{-}\mathsf{LWE}_{\mathbf{s}}(\varphi_K(\mathbf{c}))$ *such that:*

$$\mathsf{Var}\left(\mathsf{Err}(\mathbf{c}')\right) \leq \mathsf{Var}\left(\mathsf{Err}(\mathbf{c})\right) + d\frac{tN\mathsf{B}_{\mathrm{KS}}^2}{4}\vartheta_{\mathrm{KS}} + \frac{\|K\|^2}{4\mathsf{B}_{\mathrm{KS}}^{2t}} \tag{5}$$

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 $-\frac{\mathsf{B}_{\mathrm{KS}}}{2}$ and $\frac{\mathsf{B}_{\mathrm{KS}}}{2}$, hence the resulting variance for combining dtN of them.

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 $i \in [1, dN]$ 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 $[1, dN] \times [1, t]$.

Given $(\mathbf{a}, b)$ a LWE ciphertext under secret key K in input, we have:

$$\begin{aligned}
\varphi_{\mathbf{s}}(\mathbf{c}') &= \varphi_{\mathbf{s}}(\mathbf{0}, b) - \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\varphi_{\mathbf{s}}(\mathsf{KS}_{i,j}) \\
&= b - \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\left(\mathsf{B}_{\mathrm{KS}}^{-j}K_i + \mathsf{Err}(\mathsf{KS}_{i,j})\right) \\
&= b - \sum_{i=1}^{dN} a_{i,j}\mathsf{B}_{\mathrm{KS}}^{-j}K_i + \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\mathsf{Err}(\mathsf{KS}_{i,j}) \\
&= b - \sum_{i=1}^{dN}\lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t}K_i - \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\mathsf{Err}(\mathsf{KS}_{i,j}) \\
&= b - \sum_{i=1}^{dN} a_i K_i - \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{dN}(a_i - \lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t})K_i \\
&= \varphi_K(\mathbf{a}, b) - \sum_{i=1}^{dN}\sum_{j=1}^{t} a_{i,j}\mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{dN}(a_i - \lfloor a_i \rceil_{\mathsf{B}_{\mathrm{KS}}^t})K_i
\end{aligned}$$

The LWE ciphertext output by the keyswitching procedure has variance:

$$\begin{aligned}
\mathsf{Var}\left(\mathsf{Err}(\mathbf{c}')\right) &\leq \mathsf{Var}\left(\mathsf{Err}(\mathbf{c})\right) + \frac{tdN\mathsf{B}_{\mathrm{KS}}^2}{4}\vartheta_{\mathrm{KS}} + \sum_{i=1}^{dN}\frac{K_i^2}{4\mathsf{B}_{\mathrm{KS}}^{2t}} \\
&\leq \mathsf{Var}\left(\mathsf{Err}(\mathbf{c})\right) + \frac{tdN\mathsf{B}_{\mathrm{KS}}^2}{4}\vartheta_{\mathrm{KS}} + \frac{\|K\|_2^2}{4\mathsf{B}_{\mathrm{KS}}^{2t}}
\end{aligned}$$

□

B Proof of Lemma 12

Structure of R_q. 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:

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

Proof. 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 $(\mathbf{A}, \mathbf{x}^t \cdot \mathbf{A})$ 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 $\det\left((x+1)\Lambda^\perp(\mathbf{G})\right) = 2^{(d+1)\cdot\ell}\det\left(\Lambda^\perp(\mathbf{G})\right)$. This gives us the following bound on the min-entropy of $\mathbf{x} \leftarrow \mathcal{X}_1$:

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

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 $\eta_\varepsilon\left((X+1)\Lambda^\perp(\mathbf{G})\right)$, the bigger one. The columns of $\mathbf{I}_d \otimes \mathbf{B}_0 \otimes \mathbf{J}$ all have their norm bounded by $\sqrt{2+2B^2}$.

Using Lemma 2, it is sufficient to have $r \geq \sqrt{(2+2B^2)\cdot \frac{\ln(2N(d+1)\cdot\ell(1+1/\varepsilon))}{\pi}}$. Let $\varepsilon_\ell > 0$ such that:

$$\frac{1}{2}\sqrt{\sum_{t=1}^{Nk-1} \frac{1}{2^{d(Nk-1)}} \cdot \mathrm{Col}(\mathcal{X}_t)} < \varepsilon_\ell$$

Since the maximal min-entropy of $\mathbf{x} \in \mathcal{X}_t$ conditioned on $\mathbf{e}^t\mathbf{x}+\mathbf{y}$ is obtained for $t=1$, it suffices to have:

$$(Nk-1)2^{-d(Nk-1)} \cdot 2^{-H_\infty\left(\mathbf{x} \mid \mathbf{e}^t\mathbf{x}+\mathbf{y}\right)} < (2\varepsilon_\ell)^2,$$

which gives $\ell > \frac{1}{d+1}\left(\log \frac{(Nk-1)(1+\varepsilon)}{4\epsilon_\ell^2(1-\varepsilon)} + d\right)$.

□

C Detailed adaptation of [KLSS23, Lemma 7]

We give the proof of Lemma 15.

Proof. For given $y \in \hat{R}_q$ and $\mathbf{x} \in \frac{1}{q}R^{d'+d}$, the probability mass assigned to $(\mathbf{v}_0+\mathbf{x}, y)$, with $\mathbf{v}_0 = \frac{1}{2\pi\vartheta_{e''}}\Sigma_{\mathbf{v}}\binom{q\mathbf{e}}{-\mathbf{s}}\mathbf{y}$ by the first distribution is

$$\begin{aligned}
&\mathcal{D}_{\frac{1}{q}\mathbb{Z}^{(d'+d)N},\sqrt{\Sigma}}\left(\mathbf{v}_0+\mathbf{x}\right)\cdot\mathcal{D}_{\frac{1}{q}\mathbb{Z}^N,\sqrt{2\pi\vartheta_{e''}}}\left(\mathbf{y}-\Gamma\left(\mathbf{v}_0+\mathbf{x}\right)\right)\\
&\propto \exp\left(-\pi\left(\left(\mathbf{v}_0+\mathbf{x}\right)^t\Sigma^{-1}\left(\mathbf{v}_0+\mathbf{x}\right)+\frac{1}{2\pi\vartheta_{\mathbf{e}''}}\left(\mathbf{y}-\Gamma\left(\mathbf{v}_0+\mathbf{x}\right)\right)^t\left(\mathbf{y}-\Gamma\left(\mathbf{v}_0+\mathbf{x}\right)\right)\right)\right)\\
&= \exp\left(-\pi\left(\left(\mathbf{v}_0+\mathbf{x}\right)^t\Sigma_{\mathbf{v}}^{-1}\left(\mathbf{v}_0+\mathbf{x}\right)+\frac{1}{2\pi\vartheta_{\mathbf{e}''}}\left(\mathbf{y}^t\mathbf{y}-\left(\mathbf{v_0}+\mathbf{x}\right)^t\Gamma^t\mathbf{y}-\mathbf{y}^t\Gamma\left(\mathbf{v}_0+\mathbf{x}\right)\right)\right)\right)\\
&= \exp\left(-\pi\left(\left(\mathbf{v}_0+\mathbf{x}\right)^t\Sigma_{\mathbf{v}}^{-1}\left(\mathbf{v}_0+\mathbf{x}\right)+\frac{1}{2\pi\vartheta_{\mathbf{e}''}}\mathbf{y}^t\mathbf{y}-\left(\mathbf{v_0}+\mathbf{x}\right)^t{\Sigma_{\mathbf{v}}}^{-1}\mathbf{v}_0-{\mathbf{v}_0}^t{\Sigma_{\mathbf{v}}}^{-1}\left(\mathbf{v}_0+\mathbf{x}\right)\right)\right)\\
&= \exp\left(-\pi\left(\mathbf{x}^t\Sigma_{\mathbf{v}}^{-1}\mathbf{x}-{\mathbf{v}_0}^t\Sigma_{\mathbf{v}}^{-1}\mathbf{v}_0+\frac{1}{2\pi\vartheta_{\mathbf{e}''}}\mathbf{y}^t\mathbf{y}\right)\right)
\end{aligned}$$

So, for any given y, the mass given to $\mathbf{v}_0+\mathbf{x}$ is proportional to $\rho_{\sqrt{\Sigma_{\mathbf{v}}}}$. Hence the result.