Published in D. Lie and V. Cortier, Eds., *32nd ACM Conference on Computer and Communications Security (ACM CCS 2025)*, pp. 2951-2965, ACM Press, 2025.

Fast Homomorphic Evaluation of LWR-based PRFs

Amit Deo
Zama
Paris, France

Marc Joye
Zama
Paris, France

Benoît Libert
Zama
Paris, France

Benjamin R. Curtis
Zama
Paris, France

Mayeul de Bellabre
Zama
Paris, France

Abstract

Certain applications of fully homomorphic encryption (such as transciphering, universal thresholdizers, and PIR) require randomness while operating over encrypted data. This randomness has to be obliviously generated in the encrypted domain and remain encrypted throughout the computation. Moreover, it should be guaranteed that independent-looking random coins can be obliviously generated for different computations.

In this work, we consider the homomorphic evaluation of pseudorandom functions (PRFs) with a focus on practical lattice-based candidates. In the homomorphic PRF evaluation setting, given a fully homomorphic encryption of the PRF secret key $\boldsymbol{s}$, it should be possible to homomorphically compute encryptions of PRF evaluations $\{\mathsf{PRF}_{\boldsymbol{s}}(x_i)\}_{i=1}^{M}$ for public inputs $\{x_i\}_{i=1}^{M}$. We consider this problem for PRF families based on the hardness of the Learning-With-Rounding (LWR) problem introduced by Banerjee, Peikert, and Rosen (EUROCRYPT 2012). We build on a random oracle variant of a PRF construction suggested by Banerjee *et al.* and demonstrate that it can be evaluated using only two sequential programmable bootstraps in the TFHE homomorphic encryption scheme. We also describe several modifications of this PRF—which we prove as secure as the original function—that support homomorphic evaluations using only one programmable bootstrap per slot.

Numerical experiments were conducted using practically relevant FHE parameter sets from the TFHE-rs library. Our benchmarks show that a throughput of about 1000 encrypted pseudorandom bits per second (resp. 900 encrypted pseudorandom bits per second) can be achieved on an AWS hpc7a.96xlarge machine (resp. on a standard laptop with an Apple M2 chip), on a single thread. The PRF evaluation keys in our experiments have sizes roughly 40% and 60% of a bootstrapping key.

CCS Concepts

• **Security and privacy** → **Cryptography**; **Software and application security**;

Keywords

Fully homomorphic encryption (FHE), Pseudorandom functions (PRFs), Learning-with-rounding (LWR), Oblivious randomness generation

ACM Reference Format:
Amit Deo, Marc Joye, Benoît Libert, Benjamin R. Curtis, and Mayeul de Bellabre. 2025. Fast Homomorphic Evaluation of LWR-based PRFs. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765177

1 Introduction

Fully homomorphic encryption (FHE) provides a method of outsourcing computation on sensitive data [65, 41]. In particular, a client may encrypt data (e.g., patient medical records) using an FHE scheme and outsource some computation/evaluation (e.g., some diagnosis) on that data. Importantly, the server never learns the initial data, or the result of its computation.

A recurring problem with existing FHE schemes [22, 20, 36, 44, 24, 25] is their ciphertext expansion: a ciphertext is normally at least an order of magnitude larger than its corresponding plaintext. This poses a significant problem when storing a large number of FHE ciphertexts in a remote database. A solution inspired by the performance of symmetric-key ciphers is called *transciphering*. The idea is to first store symmetric-key ciphertexts in the remote database long term. Then, whenever computation on some database element(s) is requested, the server homomorphically evaluates the secret-key deciphering algorithm to obtain FHE ciphertext(s) encrypting the same database plaintext(s). To preserve privacy, the server is given an encrypted version of the client's symmetric key. In summary, transciphering essentially allows a server to use an encrypted version of the client's symmetric key to transform a symmetric-key ciphertext into an FHE one. As pseudorandom functions (PRFs) are a key building block of symmetric-key cryptography, the problem of transciphering boils down to homomorphically evaluating a PRF as efficiently as possible.

The problem of homomorphically evaluating pseudorandom functions also arises in the context of universal thresholdizers (UT) [18, 2, 35], which generically provide threshold realizations of various primitives (including public-key encryption and digital signatures). In particular, UTs can be used to turn any digital signature into a non-interactive threshold signature. The idea is to homomorphically execute the signing algorithm of the underlying signature before running a threshold decryption protocol on the resulting FHE ciphertext so as to obtain the final signature. However, if the signature scheme is probabilistic, its signing algorithm must be

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS '25, Taipei, Taiwan

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765177

de-randomized (by the standard trick of deriving its randomness from the message using a PRF) to ensure that all parties will decrypt the same FHE ciphertext.

Yet another application of homomorphically evaluating PRFs is producing encrypted randomness for blockchain smart contracts. An example of where one might require this is for simulating dice rolls or more generally playing games with randomness on the blockchain. In slightly more detail, a privacy-preserving blockchain may consist of a series of FHE ciphertexts encrypted under a key shared amongst an assigned group of validators. The task of these validators is to decrypt *particular* ciphertexts in a distributed manner. An example of an implementation of such a blockchain is fhEVM [32]. During a game/smart contract execution, clients can produce encryptions of random values by homomorphically evaluating a PRF. In doing so, the plaintexts remain hidden from the client but can still be considered random by relying on the security of the underlying PRF. These ciphertexts can then be used as dice rolls or even fed into an encrypted shuffling algorithm in a card game. Shuffling based on FHE-encrypted PRF outputs is also useful [37] in the context of private information retrieval [27]. In more theoretical applications, FHE-friendly PRFs/PRGs can also serve as building blocks for circuit-size-independent non-interactive zero-knowledge proofs [41, 42]. They can also be used [57] to decrease the communication complexity of garbled circuit protocols.

When it comes to homomorphically computing secret-key pseudorandom objects, one may end up evaluating a complex circuit, which may be time-consuming and lead to impractical parameters without resorting to bootstrapping. Indeed, all existing FHE schemes involve ciphertexts containing a noise that grows during homomorphic evaluations. At some point, the noise grows too large to enable correct decryption, so FHE schemes specify a bootstrapping algorithm which resets the noise to some predefined size. FHE schemes such as FHEW/TFHE [34, 25] take bootstrapping one step further. In particular, in their basic version, they enable the application of a *negacyclic* univariate function to the plaintext during the bootstrapping operation. The process of homomorphically evaluating a function during a bootstrapping operation is referred to as *programmable* bootstrapping (PBS). Essentially, for any *negacyclic*[1] univariate function f, applying the programmable bootstrapping algorithm takes an encryption of m and outputs an encryption of $f(m)$ with a predefined noise level. Another important point is that bootstrapping in FHEW/TFHE is very efficient in terms of latency (i.e., takes milliseconds) compared to BFV/BGV [20, 36, 22] where bootstrapping takes multiple seconds.

1.1 Our Contributions and Techniques

We consider the question of how efficiently we can homomorphically evaluate pseudorandom functions based on lattice assumptions. More specifically, we address the problem of evaluating PRFs based on the difficulty of the Learning-With-Rounding (LWR) problem [12], which can be seen as a variant of the Learning-With-Errors (LWE) problem where the noise is deterministically generated. For a public matrix $\mathbf{A} \in \mathbb{Z}_Q^{n\times m}$ with moduli p and Q such that $p < Q$ and a secret vector $\boldsymbol{s} \in \mathbb{Z}^n$, the LWR problem is to distinguish $\lceil (p/Q)\cdot(\mathbf{A}^\top \cdot \boldsymbol{s} \bmod Q)\rfloor$ from a uniformly random vector in $\mathbb{Z}_p^m$. Here, the notation $\lceil\cdot\rfloor$ denotes rounding to the nearest integer (rounding upwards in the case of a tie). The conjectured hardness of LWR naturally leads to a pseudorandom generator [12, Section 1.1] which expands a seed $\boldsymbol{s}$ into a longer pseudorandom string $\lceil (p/Q)\cdot(\mathbf{A}^\top \cdot \boldsymbol{s} \bmod Q)\rfloor$ using a public matrix $\mathbf{A}$. By the GGM construction [46], it also implies a pseudorandom function family. Furthermore, it yields a more direct PRF construction (explicitly described in [19] but already implicit in [12]) in the random oracle model. Given input x and a secret key $\boldsymbol{s} \in \mathbb{Z}^n$, the PRF evaluation is defined to be $\lceil (p/Q)\cdot(\mathbf{A}(x)^\top \cdot \boldsymbol{s} \bmod Q)\rfloor$, where the matrix $\mathbf{A}(x) = H(x) \in \mathbb{Z}_Q^{n\times m}$ is derived from a random oracle H.

In this paper, we show that the more efficient random-oracle-based construction can be evaluated efficiently using a small number of sequential PBSes if we restrict the ratio Q/p to be small. We note that the known reductions from LWE to LWR either assume that Q/p is super-polynomial [12] or that the number m of samples given to the distinguisher is a priori bounded [7, 15]. In fact, a certain class of reductions for $Q/p = \mathrm{poly}(\lambda)$ and an unbounded number of samples was shown to be impossible [63]. However, even for an unbounded number of samples, LWR still appears to be exponentially hard (as commented in [12, 16]) in the parameter regime $Q/p = \Omega(\sqrt{n})$ when $p \mid Q$.

We note that our approach would also apply to evaluate the Banerjee and Peikert PRF [11]. They generalise the key homomorphic PRF from [12, 19] to prove pseudorandomness in the standard model via different encodings of the input. However, in this case we cannot claim security under LWE, because their proof only applies for a super-polynomial ratio Q/p and we require this ratio to be small. Instead, we keep the random oracle version, which is more efficient and can be proved pseudorandom under LWR in the random oracle model.

In the following, we show that, for a polynomial ratio Q/p, we can evaluate the above LWR-based PRF using a small number of sequential bootstraps. The idea is to compute an input-dependent vector $\boldsymbol{a} = H(x) \in \mathbb{Z}_Q^n$ and view $\boldsymbol{c} = (-\boldsymbol{a}, 0) \in \mathbb{Z}_Q^{n+1}$ as an LWE ciphertext (with plaintext modulus p) that has a very large noise, but still decrypts to $\left\lceil \frac{p}{Q}\cdot(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \mod Q)\right\rfloor$ under the LWE secret key $\boldsymbol{s} \in \mathbb{Z}^n$. So, if we have an FHE bootstrapping key encrypting the PRF secret key $\boldsymbol{s}$, we can obtain a low-noise encryption of the same value $\left\lceil \frac{p}{Q}\cdot(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \mod Q)\right\rfloor$ by bootstrapping $\boldsymbol{c} = (-\boldsymbol{a}, 0) \in \mathbb{Z}_Q^{n+1}$.

With the bootstrapping techniques of FHEW [34] and TFHE [25], one difficulty is that we must find the appropriate *negacyclic* functions to evaluate in order to perform this operation. A direct application of the techniques from Liu *et al.* [58] to evaluate the original PRF requires three sequential PBSes for each slot of $\log p$ pseudorandom bits. To improve upon this baseline, we use a "Full-domain functional bootstrapping" method from Ma *et al.* [60] so as to reduce the PBS depth (i.e., the number of sequential bootstraps per pseudorandom plaintext slot) to 2.

Finally, we suggest a modified version of the random oracle-based PRF of [12, 19] which supports homomorphic evaluations in depth one and dispenses with the need to sequentially evaluate different negacyclic functions. By "depth-one", we mean that, if the output space of the PRF is $\mathbb{Z}_p^\ell$, each slot of $\log p$ output bits only costs one PBS to evaluate and ℓ slots can be processed in

[1] In particular, for a domain $\mathbb{Z}_{2Q}$ and image $\mathbb{Z}_q$ a negacyclic function f satisfies $f(x+Q) = -f(x) \bmod q$.

parallel in order to obtain a long output in $\mathbb{Z}_p^\ell$. Our construction essentially applies a PBS using a single negacyclic version of the usual rounding function. As a result, the "ciphertext" $(-\boldsymbol{a}, 0)$ from above gets mapped to an encryption of

$$(-1)^{\mathrm{msb}(\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod 2Q)} \cdot \left\lfloor \frac{p}{Q} \cdot (\langle \boldsymbol{a}, \boldsymbol{s} \rangle \mod Q) \right\rceil . \tag{1}$$

When p and Q are powers of two, we can show that the above function is a PRF based on the pseudorandomness of LWR. The reduction works by deriving the base-2 digits and the sign of the modified PRF from LWR samples whose moduli are scaled up by appropriate powers of two.

We can prove that the modified PRF is as secure as the original one, and we give reductions for both the floor function and the nearest integer rounding function. Note that the former achieves slightly better parameters. Although the base 2 is appropriate for practical implementations of TFHE, one could replace it by other bases if required.

A point worth mentioning is that due to the structure of TFHE, we typically end up using LWR with power-of-two moduli $Q \in \{2^9, 2^{10}, 2^{11}\}$ and $p \in \{2, \ldots, 2^5\}$ in our depth-1 construction. In particular, p is often very small which allows LWR to remain hard for relatively small LWR dimension n_{LWR} compared to the TFHE LWE dimension n_{LWE}. This can be leveraged by truncating the PBS operation; i.e., by performing n_{LWR} blind rotation steps during the PBS rather than the usual n_{LWE} steps. Since blind rotation dominates PBS latency, we may achieve a factor $n_{\mathrm{LWR}}/n_{\mathrm{LWE}}$ improvement in terms of latency.

1.2 Related Work

The idea of using bootstrapping to obliviously generate FHE encryptions of random bits was previously used in the past (see, e.g., [1]). In this paper, we consider a derandomized version of the process where we prove that obliviously generated ciphertexts indeed encrypt pseudorandom messages uniquely determined by an encrypted seed and public input. For this purpose, we also aim at relying on the pseudorandomness of a well-studied PRF family.

Homomorphic evaluation of PRFs/ciphers using FHE has received a lot of attention in the research literature. With transciphering in mind, a natural task was to optimize the evaluation of AES [43]. Until recently, the efficiency of this approach was questionable. However, recent works managed to evaluate a single block of AES around 30 seconds [66] and 9 seconds [67] using 16-threaded implementations. Another line of research is the development of FHE-friendly cipher/PRF constructions such as LowMC [6], PASTA [33], FASTA [28], FLIP [62], FiLIP [61], Elisabeth [29], Rubato [48], Chaghri [8], and Transistor [13]. Unfortunately, the security level of such schemes is not well understood and attacks are still being discovered [47, 56, 45]. In an attempt to avoid this problem, the standardized cipher Trivium [50] and the subsequent cipher Kreyvium [23] have been investigated as good options for efficient transciphering [10]. The homomorphic evaluation of Trivium and other symmetric primitives (including SIMON, AES, and Keccack) was also considered via a framework [17] allowing to evaluate more complex Boolean functions. However, even Trivium and Kreyvium have recently been subjected to improved attacks [49]. For the sake of not putting all one's eggs in the same basket, it is desirable to have alternative solutions based on more stable algorithmic assumptions, in particular if they enable higher throughputs than the homomorphic evaluation of stream ciphers. This motivates us to consider LWR-based PRFs and exploit the fact that their structure blends quite well with the bootstrapping paradigm of FHEW/TFHE.

In a direction somewhat analogous to ours, a recent work shows how to homomorphically evaluate an adaptation of the LWR PRF using the BGV/BFV scheme [37]. In particular, this work tweaks the LWR-based PRF in a way that replaces the exact rounding function with an alternative based on the Legendre symbol. The advantage of this is that the resulting PRF is homomorphically computable using a reasonably small number of leveled multiplications (concretely ≈ 20). It should be noted that these leveled multiplications lead to an output ciphertext with a noise larger than that of a freshly bootstrapped ciphertext which may need to be considered in certain applications. Moreover, their modified rounding function makes the resulting PRF significantly deviate from the well-known construction and introduces a novel variant of the LWR assumption. As of today, this assumption does not appear to be implied by the original assumption and has not undergone much cryptanalytic effort. In contrast, we rely on an LWR assumption that has been standing for over a decade. A related work that homomorphically evaluates the random-oracle variant of the *standard* LWR-based PRF is [31]. The FHE scheme used is BGV and the implementation uses a "$\Lambda \circ \lambda$" [30] backend. The main strength of this line of work lies in the simplicity from a programmer perspective. In particular, an un-batched homomorphic PRF evaluation can be implemented in just a few dozen lines. The implementation produces a reported 64 encrypted bits at a latency of around 10 seconds.[2] However, it is worth noting that the experiments are run on a less powerful machine than ours and the authors mention that further optimization should be possible.

Our approach has a common feature with the HERMES transciphering technique [9] in that the latter also allows switching from a lattice-based symmetric cipher (with the difference that they use an LWE-based one while we rely on LWR) to CKKS/BGV/BFV (instead of TFHE in our case) without relying on any ad-hoc assumption. Yet, their 1.58 expansion factor (i.e., the ratio between the size of the ciphertext stored on a server and the plaintext size) is larger than ours. Moreover, they achieve a latency of around 26 seconds before any output is computed with around 60,000 bits per second of amortized throughput. To achieve these numbers, 5.31MB of additional key material is used (which is just 1% of the corresponding bootstrapping key size).

Brakerski *et al.* [21] took a different (non-transciphering-based) approach allowing to reduce the expansion rate of FHE schemes by constructing a ciphertext compression mechanism leading to rate-$(1 - o(1))$ FHE. Their technique applies to packed LWE ciphertexts sharing a common header (typically, a vector over $\mathbb{Z}_q^n$) where ℓ message-carrying slots encrypt ℓ distinct binary plaintexts under *distinct* LWE secrets. It shrinks each of these ℓ slots down to a single bit, thus replacing a vector in $\mathbb{Z}_q^\ell$ by a binary string $\{0, 1\}^\ell$. For $\ell = \widetilde{\Omega}(\lambda^2) = \mathrm{poly}(\lambda)$ plaintexts, the achieved rate (i.e. plaintext size divided by ciphertext size) is $1 - \mathcal{O}(1/\lambda)$ when expanding the

[2] Results were recorded on a 2015 iMac with a 4 GHz Core i7 and 16GB RAM.

ciphertext header from a seed. In other words, the size of a compressed output ciphertext on ℓ plaintext inputs approaches the size of around ℓ plaintexts as ℓ grows to infinity. However, subsequent FHE evaluations cannot be done on compressed ciphertexts and require bootstrapping to "undo" the compression. Applying this method to plain TFHE thus requires $\ell = \mathrm{poly}(\lambda)$ bootstrapping keys (one for each of the original ℓ distinct keys). This overhead may be avoided by introducing a ring structure to TFHE as in [52] with a larger than usual ring dimension ℓ to achieve reasonable expansion factor. However, the sub-optimal granularity (i.e., the fact that a block of $\ell = \mathrm{poly}(\lambda)$ ciphertexts is required before a compression operation can begin) remains. Note that one may also apply the compression to BGV/BFV.

Outline of the paper. The rest of this paper is organized as follows. We begin with background and definitions in Section 2. Next, in Section 3, we discuss the homomorphic evaluation of the standard LWR function in depth 2. Then, in Section 4 we briefly discuss ring-LWR-based PRFs in the context of BFV. In Section 5, we present the depth-1 evaluation of our modified LWR-based PRF and prove it as secure as the original one. Further variants are also discussed. This is followed by an implementation of a depth-1 construction using the TFHE-rs library in Section 6.

In the supplementary material, we recall the standard definition of pseudorandom functions and the description of GGSW ciphertexts. We also present a concrete instantiation of the transciphering application.

2 Background and Definitions

Notation. In the following, when D is a distribution, $x \sim D$ means that x is a random variable distributed according to D. The notation $x \hookleftarrow D$ denotes the explicit action of sampling an element x according to the distribution D. For a finite set $\mathcal{S}$, $U(\mathcal{S})$ stands for the uniform distribution over $\mathcal{S}$. For any integer $q \geq 2$, $\mathbb{Z}_q$ denotes the ring of integers with addition and multiplication modulo q. For any real number y, we use $\lceil y \rfloor$ to denote rounding y to the nearest integer (rounding upwards in the case of a tie), $\lfloor y \rfloor$ to denote the floor function and $\lceil y \rceil$ to denote the ceiling function. If y is replaced by a vector $\boldsymbol{y}$, we apply the rounding, floor and ceiling functions entry-wise. Logarithms will always have a base of 2. For $x \in \mathbb{Z}_q$, $\mathrm{msb}(x)$ denotes the most significant bit of the length-$\lceil \log q \rceil$ binary representation of x interpreted as an integer in $\{0, \ldots, q-1\}$.

2.1 Cryptographic Assumptions

LWE/LWR Assumptions. We first recall the *Learning-With-Errors (LWE)* assumption defined by Regev [64].

Definition 2.1 (LWE assumption). Let integers $m \geq n \geq 1$, $q \geq 2$ and let χ_s, χ_e be distributions over $\mathbb{Z}$. The $\mathrm{LWE}_{n,m,q,\chi_s,\chi_e}$ problem consists in distinguishing between the distributions

$$\{(\mathbf{A}^\top, \mathbf{A}^\top \boldsymbol{s} + \boldsymbol{e}) \mid \mathbf{A} \sim U(\mathbb{Z}_q^{n\times m}),\ \boldsymbol{s} \hookleftarrow \chi_s^n,\ \boldsymbol{e} \sim \chi_e^m\}$$

and $U(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m)$.

When the distribution of $\boldsymbol{s}$ is the uniform distribution $U(\mathbb{Z}_q^n)$, the assumption is sometimes denoted by $\mathrm{LWE}_{n,m,q,\chi_e}$.

We now recall the *Learning-With-Rounding* (LWR) problem [12].

Definition 2.2 (LWR assumption). Let integers $m \geq n \geq 1$, $q > p \geq 2$ and let χ_s be a distribution over $\mathbb{Z}$. The *Learning-With-Rounding* ($\mathrm{LWR}_{n,m,q,p,\chi_s}$) problem consists in distinguishing between the distributions

$$\{(\mathbf{A}^\top, \lceil (p/q)\cdot(\mathbf{A}^\top \boldsymbol{s} \bmod q)\rfloor \bmod p) \mid \mathbf{A} \sim U(\mathbb{Z}_q^{n\times m}),\ \boldsymbol{s} \hookleftarrow \chi_s^n\}$$

and $U(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_p^m)$.

When the number m of samples is a priori bounded, the LWE assumption is known [7, 15] to imply the hardness of LWR for a polynomial ratio $q/p = \mathrm{poly}(\lambda)$. When there is no pre-determined upper bound on the number of samples, the only known reduction [12, Theorem 3.2] from LWE to LWR requires a super-polynomial ratio $q/p = \Omega(\lambda^{\omega(1)})$. However, it is quite plausible that LWR remains hard for $q/p = \mathrm{poly}(\lambda)$ even for an a priori unbounded number of samples. As discussed in [12, 16], as long as $q/p = \Omega(\sqrt{n})$ and q/p is an integer (so that $\lceil (p/q)\cdot U(\mathbb{Z}_q)\rfloor = U(\mathbb{Z}_p)$), LWR may be exponentially hard even for quantum algorithms. We also note that replacing the rounding function $\lfloor\cdot\rceil$ by the floor function is not believed to affect the hardness of the LWR problem [12, Sect. 2]. When evaluating the concrete security of our LWR-based parameter sets, we consider the approach from [3], which models an LWR sample $\left(a, b := \left\lfloor \frac{p}{q}\langle a, s\rangle \right\rceil\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ as an LWE sample via assuming that $\frac{q}{p}\cdot b = \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e$. This LWE sample is then assumed to have uniform noise over the set $\{-\frac{q}{2p}+1, \ldots, \frac{q}{2p}\}$. We note that this is a standard approach for evaluating the concrete security of LWR-based parameter sets and is supported in theory. Specifically, for q divisible by p, [15, Theorem 5] shows a reduction from LWE with uniform errors (in effectively the same interval as above) to LWR for any secret distribution.

Ring-LWE/LWR Assumptions. We now recall the definition of the ring Learning-With-Errors problem [59].

Definition 2.3 (RLWE assumption). Take an integer $q \geq 2$. Let $\Phi(X)$ be a cyclotomic polynomial of degree N and let the rings $\mathcal{R} = \mathbb{Z}[X]/(\Phi(X))$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ_s, χ_e be distributions over $\mathcal{R}$. The *Ring LWE* ($\mathrm{RLWE}_{N,m,q,\chi_s,\chi_e}$) problem consists in distinguishing between the distributions

$$\{(\mathscr{a}, \mathscr{a}\cdot\mathscr{s} + \mathscr{e}) \mid \boldsymbol{a} \sim U(\mathcal{R}_q^m),\ \ \mathscr{s} \hookleftarrow \chi_s,\ \mathscr{e} \sim \chi_e^m\}$$

and $U(\mathcal{R}_q^m \times \mathcal{R}_q^m)$.

The LWR problem has a natural analogue in the ring setting. The *Ring Learning-With-Rounding* (RLWR) problem [12] is defined as follows.

Definition 2.4 (RLWR assumption). Let integers $q > p \geq 2$. Let $\Phi(X)$ be a cyclotomic polynomial of degree N and let the rings $\mathcal{R} = \mathbb{Z}[X]/(\Phi(X))$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ_s be a distribution over $\mathcal{R}$. The *Ring Learning-With-Rounding* ($\mathrm{RLWR}_{N,m,q,p,\chi_s}$) problem consists in distinguishing between the distributions

$$\{(\mathscr{a},\ \lceil (p/q)\cdot(\mathscr{a}\cdot\mathscr{s} \bmod q)\rfloor \bmod p) \mid \mathscr{a} \sim U(\mathcal{R}_q^m),\ \mathscr{s} \hookleftarrow \chi_s\}$$

and $U(\mathcal{R}_q^m \times \mathcal{R}_p^m)$.

We finally recall the definition of the *Generalized Learning-With-Errors* (GLWE) problem (also known as the *Module-Learning-With-Errors* problem) studied in [54] that is useful when discussing the FHEW/TFHE [34, 25] FHE schemes.

Definition 2.5 (GLWE assumption). Take an integer $q \geq 2$ and a rank $k \geq 1$. Let $\Phi(X)$ be a cyclotomic polynomial of degree N and take the rings $\mathcal{R} = \mathbb{Z}[X]/(\Phi(X))$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ_s, χ_e be distributions over $\mathcal{R}$. The *Generalized LWE* (GLWE$_{k,N,m,q,\chi_s,\chi_e}$) problem consists in distinguishing between the distributions

$$\left\{(\mathbf{A}^\top, \mathbf{A}^\top \cdot \mathscr{s} + \mathscr{e}) \mid \mathbf{A} \sim U(\mathcal{R}_q^{k\times m}),\ \ \mathscr{s} \hookleftarrow \chi_s^k,\ \mathscr{e} \sim \chi_e^m\right\}$$

and $U(\mathcal{R}_q^{m\times k} \times \mathcal{R}_q^m)$.

2.2 (Key-Homomorphic) Pseudorandom Functions Based on LWR

In [12] (see also [16]), Banerjee, Peikert, and Rosen implicitly describe a weak pseudorandom function based on the hardness of the LWR problem. This weak PRF maps a uniformly random input $\boldsymbol{a} \in \mathbb{Z}_Q^n$ to the output $\mathrm{wPRF}_{\boldsymbol{s}}(\boldsymbol{a}) = \lceil (p/Q) \cdot (\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q)\rfloor \bmod p$, where $\boldsymbol{s} \in \mathbb{Z}^n$ is the secret key.

By introducing a random oracle $H\colon \{0,1\}^* \to \mathbb{Z}_Q^n$, this weak PRF can be turned into a full PRF by computing $\boldsymbol{a} = H(x) \in \mathbb{Z}_Q^n$ when the PRF is to be evaluated on arbitrary input $x \in \{0,1\}^\ell$. As pointed out in [19], this PRF turns out to be almost key-homomorphic PRF. An explicit security proof was given in [40, Theorem 3.1]. It is precisely defined as follows.

The secret key is a vector $\boldsymbol{s} = (s_1, \ldots, s_n) \sim \chi_s^n$ where each s_i is sampled from a distribution χ_s specified by public parameters. These public parameters also contain the description of a hash function $H : \{0,1\}^* \to \mathbb{Z}_Q^n$ (modeled as a random oracle) and two moduli p and Q where p divides Q. Typically, χ_s is the uniform distribution over $\mathbb{Z}_Q$. Alternatively, χ_s can be a discrete Gaussian distribution with a suitable standard deviation σ or even the uniform binary distribution. We note that in any of these cases, the parameters (n, q, p) can be chosen carefully to protect against all known attacks.

A function evaluation is then defined as

$$x \mapsto \mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq \left\lceil \tfrac{p}{Q} \cdot \big(\langle H(x), \boldsymbol{s}\rangle \bmod Q\big)\right\rfloor \bmod p \qquad (2)$$

and outputs a scalar in $\mathbb{Z}_p$. If we need to output t pseudorandom elements in $\mathbb{Z}_p$, we can extend it as $x \mapsto \mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq (y_1, \ldots, y_t)$, where

$$y_i = \left\lceil \tfrac{p}{Q} \cdot \big(\langle H(x,i), \boldsymbol{s}\rangle \bmod Q\big)\right\rfloor \bmod p \qquad \forall i \in \{1, \ldots, t\}\ .$$

When proving the pseudorandomness of the function (2) (in the random oracle model), the reduction (see [12] or [40, Theorem 3.1]) is given from an instance of LWR where the number of samples is *not* a priori bounded since each queried input x is mapped to a different sample (i.e., the number of samples is as large as the number of evaluation queries).

Therefore we need to choose a super-polynomial $Q/p = \lambda^{\omega(1)}$ if we want to rely on known LWE-to-LWR reductions [12]. Alternatively, one may prefer a more efficient choice of parameters with $Q/p = \mathrm{poly}(\lambda)$ and rely on the plausible hardness of LWR in this parameter regime.[3] In this case, it is recommended in [12] to set Q/p as an integer larger than $\Omega(\sqrt{n})$ if n is the dimension of $\boldsymbol{s}$.

In [12, 16], LWR was conjectured to be exponentially hard when $Q/p = \Omega(\sqrt{n})$ and assuming uniform secret keys (i.e., $\chi_s = U(\mathbb{Z}_Q)$). In order to homomorphically evaluate the PRF using programmable bootstrapping, it is more convenient to sample the seed $\boldsymbol{s}$ from a binary or ternary distribution. In practice, we use the lattice estimator[4] [5] to derive secure parameters. Although the lattice estimator is designed for LWE, we deploy the usual heuristic method of approximating the hardness of LWR by that of LWE with uniform noise in the interval $\left[-\frac{Q}{2p} + 1, \frac{Q}{2p}\right]$. Fortunately, even for the uniform binary distribution $\chi_s = U(\{0,1\})$, we can find 128-bit secure parameters for the range of moduli Q and p that our constructions require.

2.3 TFHE Bootstrapping

An LWE ciphertext encrypting a message $m \in \mathbb{Z}_p$ with respect to secret key $\boldsymbol{s}$ takes the form $(\boldsymbol{a}, b) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e + m\lceil q/p\rfloor) \in \mathbb{Z}_q^{n+1}$. Here, $\boldsymbol{a}$ is sampled uniformly and e is sampled from an error distribution χ_e. As in [58], when $(\boldsymbol{a}, b)$ is an LWE ciphertext with secret key $\boldsymbol{s}$, we denote by $\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b) = b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod q$ the decoding function (a.k.a. phase function) which outputs a noisy encoding $e + m\lceil q/p\rfloor$ of the plaintext m. We can similarly define GLWE ciphertexts by taking $\mathscr{a} \in \mathcal{R}_q^k, \mathscr{s} \in \mathcal{R}_q^k$, error $\mathscr{e} \in \mathcal{R}$ and message $m \in \mathcal{R}_p$.

Programmable Bootstrapping Subroutines. We rely on the following theorem, which is quoted from [58] but is implied by earlier results on the programmable bootstrapping of LWE ciphertexts for negacyclic functions. Note that in this theorem and throughout this paper, q is used to denote the TFHE modulus and Q will denote a smaller modulus dividing q. We will assume that $Q = 2N$ where N is the degree of the TFHE cyclotomic ring.

Theorem 2.6 ([58, Theorem 1]). *Take positive integers n, q and Q such that Q divides q and q is set to a power of 2. There is a bootstrapping procedure* Boot *with the following property: For any LWE ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_Q^{n+1}$ and any function $f\colon \mathbb{Z}_Q \to \mathbb{Z}_q$ such that $f(x + Q/2) = -f(x) \bmod q$, the procedure* Boot$[f](\boldsymbol{a}, b)$ *outputs a ciphertext $(\boldsymbol{c}, d) \in \mathbb{Z}_q^{n+1}$ such that*

$$\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) = f\big(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b)\big) + e \pmod{q}\ ,$$

where $|e| < \beta$, for a noise bound β that only depends on the operations performed by Boot *and not on the input ciphertext $(\boldsymbol{a}, b)$.*

There are three subroutines in TFHE bootstrapping: blind rotation, sample extraction, and key-switching. In what follows, we set $Q = 2N$ in the above theorem. The blind rotation in TFHE takes as input an LWE ciphertext $(\boldsymbol{a}, b) \in \mathbb{Z}_{2N}^{n+1}$ under secret key $\boldsymbol{s} \in \{0,1\}^n$ that "encrypts" a message $\tilde{\mu} \in \mathbb{Z}_{2N}$ and a test polynomial $\mathscr{v}(X)$ whose coefficients encode the outputs of a negacyclic function f in a lookup table. It returns a GLWE ciphertext $\mathscr{c}' \in \left(\mathbb{Z}_q[X]/(X^N+1)\right)^{k+1}$, under secret key with bounded coefficients $\mathscr{s}' \in R^k$, which encrypts the polynomial $X^{-b+\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N} \cdot \mathscr{v}(X)$ whose degree-0 coefficient is $f(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{a}, b))$ when f is negacyclic. This blind rotation operation requires a bootstrapping key in the form of generalized GSW (or GGSW) encryptions [44, 34] of the entries of $\boldsymbol{s}$. For completeness, we overview GGSW in Supplementary Material B. The resulting GLWE ciphertext is then sample-extracted to obtain an LWE ciphertext encrypting the degree-0

[3] We note that, in the statement of [40, Theorem 3.1], the hypothesis $Q/p = \Omega(n^{\omega(1)})$ is only needed if a reduction from LWE is desired via the LWE-to-LWR reduction of [12]. The proof still works under the LWR assumption when Q/p is polynomial.

[4] https://github.com/malb/lattice-estimator

coefficient of $X^{-b+\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod 2N} \cdot \mathscr{v}(X)$. The resulting LWE ciphertext $(\boldsymbol{c}',d') \in \mathbb{Z}_q^{kN+1}$ is encrypted under the secret key $\boldsymbol{s}' \in \mathbb{Z}_q^{kN}$ consisting of the coefficients of $\mathfrak{s}'$. A final key switch leads to an LWE ciphertext $(\boldsymbol{c},d) \in \mathbb{Z}_q^{n+1}$ under the original secret key $\boldsymbol{s}$. To sum up, we have:

$$(\boldsymbol{c},d) \leftarrow \underbrace{\text{KeySwitch} \circ \underbrace{\text{SampleExtract} \circ \underbrace{\text{BlindRotate}(\boldsymbol{a},b)}_{=\text{GLWE}_{\mathfrak{s}'}(X^{-\text{Dec}_{\boldsymbol{s}}(\boldsymbol{a},b)}\cdot \mathscr{v}(X))}}_{=\text{LWE}_{\boldsymbol{s}'}(f(\text{Dec}_{\boldsymbol{s}}(\boldsymbol{a},b)))}}_{=\text{LWE}_{\boldsymbol{s}}(f(\text{Dec}_{\boldsymbol{s}}(\boldsymbol{a},b)))}$$

provided that test polynomial $\mathscr{v}(X) = \sum_{i=0}^{N-1} v_i\, X^i$ is programmed as $v_i = f(i) \in \mathbb{Z}_q$ for some negacyclic function $f\colon\ \mathbb{Z}_{2N} \to \mathbb{Z}_q$. For a more detailed exposition, see [51]. Note that in order to get the output $(\boldsymbol{c},d) \in \mathbb{Z}_q^{n+1}$ back into the domain of Boot, one can simply apply the mod-switching operation given by $\text{ModSwitch}_{q\to 2N}(\boldsymbol{c},d) \triangleq \big(\lceil (2N/q)\cdot \boldsymbol{c} \rfloor, \lceil (2N/q)\cdot d \rfloor\big)$. This is useful when a sequence of multiple PBSes for negacyclic functions with different domains and ranges is required.

3 Homomorphic Evaluation of LWR-based PRF in Depth 2

We now describe the homomorphic evaluation of the standard LWR-based PRF in depth 2. Note that this construction is not as efficient as the depth-1 construction in Section 5 and may be skipped by the reader. Nonetheless, this section describes what can be achieved using known techniques and paves the way for the homomorphic BFV evaluation of the RLWR-based PRF discussed in Section 4. In this section, we set $Q = 2N$ and $\Delta = Q/p$ where p is a plaintext modulus dividing $2N$. Furthermore, the TFHE modulus $q > 2N$ is also assumed to be divisible by $2N$ (which is the case in practice).

In order to homomorphically evaluate the PRF in (2) for a public input x given an encryption of the seed $\boldsymbol{s} \in \mathbb{Z}^n$, the idea is to first compute an input-dependent $\boldsymbol{a} = H(x) \in \mathbb{Z}_Q^n$ and view $(-\boldsymbol{a},0)$ as an LWE ciphertext with a very large noise. Namely, assuming that $\Delta \mid Q$, if we write

$$\begin{aligned}(-\boldsymbol{a},0) = \big(-\boldsymbol{a},\ &-(\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod Q)\\ &+ \Delta \cdot \lceil (\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod Q)/\Delta \rfloor + \underbrace{(\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod \Delta)}_{\in[-\Delta/2,\Delta/2)}\big)\ ,\end{aligned}$$

we can view the term $(\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod \Delta)$ as a noise to clean up using bootstrapping. Using FHEW/TFHE-like schemes, the main difficulty is to do this using negacyclic functions.

A first solution is to use a technique proposed by Liu *et al.* [58, Section 4] which applies Theorem 2.6 to negacyclic functions. This method is based on the homomorphic floor function evaluation technique [58, Algorithm 2] that allows handling an arbitrarily large noise in the input ciphertext. For each slot of pseudorandomness, the resulting homomorphic evaluation algorithm requires three sequential PBSes.

To obtain better efficiency, we can actually use a technique from [60, Algorithm 1] so as to only call Boot twice for each plaintext slot. To do this, we use the negacyclic functions $f_C, f_{\text{eval}} : \mathbb{Z}_Q \to \mathbb{Z}_Q$ defined as

$$f_C(x) = \begin{cases} \frac{\Delta}{4}\cdot\Big(2\left\lfloor \frac{x}{\Delta}\right\rfloor + 1\Big) \bmod Q & \text{if } x \in \big[0, \frac{Q}{2}-1\big] \\ -\frac{\Delta}{4}\cdot\Big(2\left\lfloor \frac{x}{\Delta}\right\rfloor - p + 1\Big) \bmod Q & \text{if } x \in \big[\frac{Q}{2}, Q-1\big] \end{cases}$$

$$f_{\text{eval}}(x) = \begin{cases} \Delta\cdot\Big(\left\lfloor \frac{2x}{\Delta}\right\rfloor \bmod p\Big) & \text{if } x \in \big[0, \frac{Q}{4}-1\big] \\ \Delta\cdot\Big(\left\lfloor \frac{2(Q-x)}{\Delta}\right\rfloor + \frac{p}{2} \bmod p\Big) & \text{if } x \in \big[\frac{3Q}{4}, Q-1\big] \\ -f_{\text{eval}}\big(x-\frac{Q}{2}\big) \bmod Q & \text{if } x \in \big[\frac{Q}{4}, \frac{3Q}{4}-1\big] \end{cases}$$

where $\Delta = Q/p$ and the input $x \in \mathbb{Z}_Q$ is seen as a positive integer in $\{0,\dots,Q-1\}$. We note that f_C is negacyclic since, for each $x \in [Q/2, Q-1]$,

$$\begin{aligned} f_C(x-Q/2) &= \tfrac{Q}{4p}\cdot\Big(2\big\lfloor \tfrac{p}{Q}\cdot\big(x-\tfrac{Q}{2}\big)\big\rfloor + 1\Big) \\ &= \tfrac{Q}{4p}\cdot\Big(2\big\lfloor \tfrac{p}{Q}\cdot x\big\rfloor - p + 1\Big) = -f_C(x)\ . \end{aligned}$$

The negacyclic property of f_{eval} can also be checked in a similar way (with additional cases to consider):

$$\begin{aligned} f_{\text{eval}}(x+Q/2) &= \begin{cases} -f_{\text{eval}}\big((x+Q/2)-\frac{Q}{2}\big) \pmod Q & \text{if } x\in\big[0,\frac{Q}{4}-1\big] \\ \Delta\cdot\Big(\big\lfloor \frac{2(Q-(x+Q/2))}{\Delta}\big\rfloor + \frac{p}{2} \bmod p\Big) & \text{if } x\in\big[\frac{Q}{4},\frac{Q}{2}-1\big] \\ \Delta\cdot\Big(\big\lfloor \frac{2(x+Q/2)}{\Delta}\big\rfloor \bmod p\Big) & \text{if } x\in\big[\frac{Q}{2},\frac{3Q}{4}-1\big] \\ -f_{\text{eval}}\big((x+Q/2)-\frac{Q}{2}\big) \pmod Q & \text{if } x\in\big[\frac{3Q}{4},Q-1\big] \end{cases} \\ &= \begin{cases} -f_{\text{eval}}(x) \pmod Q & \text{if } x\in\big[0,\frac{Q}{4}-1\big] \\ \Delta\cdot\Big(\big\lfloor \frac{2(Q-x)}{\Delta}\big\rfloor + \frac{p}{2} \bmod p\Big) & \text{if } x\in\big[\frac{Q}{4},\frac{Q}{2}-1\big] \\ \Delta\cdot\Big(\big\lfloor \frac{2x}{\Delta}\big\rfloor \bmod p\Big) & \text{if } x\in\big[\frac{Q}{2},\frac{3Q}{4}-1\big] \\ -f_{\text{eval}}(x) \pmod Q & \text{if } x\in\big[\frac{3Q}{4},Q-1\big] \end{cases} \\ &= -f_{\text{eval}}(x) \pmod Q\ . \end{aligned}$$

Again, we assume that Q and p are both powers of 2. In the description hereunder, we also assume that the seed $\boldsymbol{s} \in \mathbb{Z}^n$ of the LWR-based PRF is encrypted in the same way as the bootstrapping key of an TFHE encryption scheme for LWE secret key $\boldsymbol{s}$. We thus assume a bootstrapping key $\mathsf{bsk}[j] = \text{GGSW}_{\mathfrak{s}'}(s_j)$, $1 \le j \le n$ consisting of GGSW encryptions [44, 34] of the bits of seed $\boldsymbol{s}$ under a GGSW secret key $\mathfrak{s}'$. Although we are viewing $\boldsymbol{s}$ as a TFHE LWE key for intuition here, the PRF seed should be distinct from all TFHE keys in any practical application to respect key separation. The evaluation algorithm then goes as follows. Here, as in [60, Section 3], the inner product $\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod Q$ is interpreted as an unsigned element of $\{0,\dots,Q-1\}$ (rather than $\{-Q/2,\dots,Q/2-1\}$) and μ is viewed as an element of $\{0,\dots,p-1\}$.

$\text{Eval}_{\text{pp}}(\mathsf{bsk},x)$: Given public parameters $\text{pp} = (n,Q,p,\beta)$, an evaluation key bsk and an input $x \in \{0,1\}^\ell$, compute the input-dependent vector $\boldsymbol{a} = H(x) \in \mathbb{Z}_Q^n$ and do the following:

1. $(\boldsymbol{c},d) := \text{Boot}[f_C](-\boldsymbol{a},\frac{\Delta}{2}) \pmod Q$
2. $(\bar{\boldsymbol{a}},\bar{b}) = \text{Boot}[f_{\text{eval}}](\boldsymbol{c},d) \pmod Q$

Output the ciphertext $\text{ct} = (\bar{\boldsymbol{a}},\bar{b}) \in \mathbb{Z}_Q^{n+1}$.

The above homomorphic evaluation algorithm outputs an LWE encryption of $\mathsf{PRF}_{\boldsymbol{s}}(x) = \big\lceil \frac{p}{Q}\cdot(\langle \boldsymbol{a},\boldsymbol{s}\rangle \bmod Q)\big\rfloor \bmod p$ (interpreted as an element of $[0,p-1]$) under an LWE secret key which is the PRF seed $\boldsymbol{s}$ itself. In order to obtain an LWE encryption of $\mathsf{PRF}_{\boldsymbol{s}}(x)$ under the LWE secret key $\boldsymbol{s}'$, we can remove the key-switching step in the second call to Boot. We further note that the final modulus switch may be removed.

The proof of the following lemma is adapted from [60, Lemma 3.1], with all details written out.

Lemma 3.1. *Assume that p and Q are both powers of 2 and that $\Delta = Q/p > 4\beta$, where β is the bootstrapping error from Theorem 2.6. For any $x \in \{0,1\}^\ell$,* Eval *outputs a ciphertext $(\bar{\boldsymbol{a}}, \bar{b}) \in \mathbb{Z}_Q^{n+1}$ such that* $\mathrm{Dec}_{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \bar{b}) = \Delta \cdot \mu + e \pmod Q$*, where $|e| < \beta$,*

$$\mu = \left\lfloor \tfrac{p}{Q} \cdot \big(\langle \boldsymbol{a}, \mathbf{s}\rangle \bmod Q\big) \right\rceil \bmod p \tag{3}$$

with $\boldsymbol{a} = H(x) \in \mathbb{Z}_Q^n$.

Proof. We first note that

$$\begin{aligned}(-\boldsymbol{a}, 0) = \big(-\boldsymbol{a}, -(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q) + \Delta \cdot \lfloor(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q)/\Delta\rceil \\ + \underbrace{(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod \Delta)}_{\in[-\Delta/2,\Delta/2)}\big) \in \mathbb{Z}_Q^{n+1}\end{aligned}$$

where $\Delta = Q/p$ and $(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q) \in \{0, \dots, Q-1\}$. Then, before Line 1, we have

$$\begin{aligned}\mathrm{Dec}_{\boldsymbol{s}}(-\boldsymbol{a}, 0) &= \langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q \\ &= \Delta \cdot \underbrace{\lfloor(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod Q)/\Delta\rceil}_{\triangleq \mu} + (\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod \Delta) \pmod Q,\end{aligned}$$

where $\mu \in \{0, \dots, p-1\}$,[5] and we can interpret $(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod \Delta)$ as a very large noise. Initially, we have

$$\mathrm{Dec}_{\boldsymbol{s}}(-\boldsymbol{a}, \tfrac{\Delta}{2}) = \Delta \cdot \mu + \underbrace{\big((\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod \Delta) + \tfrac{\Delta}{2}\big)}_{\triangleq \bar{e}} \pmod Q \tag{4}$$

where $\bar{e} \in [0, \Delta)$. Then, we distinguish two cases.

Case I: $\Delta \cdot \mu + \bar{e} \in [0, Q/2-1]$

We have

$$\begin{aligned} f_C(\Delta \cdot \mu + \bar{e}) &= \tfrac{\Delta}{4} \cdot \left(2\left\lfloor \tfrac{\Delta \cdot \mu + \bar{e}}{\Delta} \right\rfloor + 1\right) \bmod Q \\ &= \tfrac{\Delta}{4} \cdot (2\mu + 1) \bmod Q \ . \end{aligned}$$

After Line 1, we obtain

$$\begin{aligned} \mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) &\equiv f_C\big(\mathrm{Dec}_{\boldsymbol{s}}(-\bar{\boldsymbol{a}}, \tfrac{\Delta}{2})\big) + e_\beta \\ &\equiv f_C(\Delta \cdot \mu + \bar{e}) + e_\beta \equiv \tfrac{\Delta}{4} \cdot (2\mu + 1) + e_\beta \pmod Q \end{aligned}$$

for some $e_\beta \in (-\beta, \beta)$. Moreover, we know that

$$f_C(\Delta \cdot \mu + \bar{e}) \in \left[\tfrac{\Delta}{4}, \tfrac{Q}{4} - \tfrac{\Delta}{4}\right]$$

because $f_C(x) \in [0, Q/4 - \Delta/4]$ for any $x \in [0, Q/2-1]$ and we cannot have $\frac{\Delta}{4} \cdot (2\mu + 1) \in [0, \Delta/4)$ for $\mu \in \{0, \dots, p-1\}$. Since $|e_\beta| < \beta < \Delta/4$, this implies

$$\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) \bmod Q \ = \tfrac{\Delta}{4} \cdot (2\mu + 1) + e_\beta \quad \in [0, Q/4 - 1] \ .$$

By the definition of f_{eval}, this in turn yields

$$\begin{aligned} f_{\mathrm{eval}}\big(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) \bmod Q\big) &\equiv \Delta \cdot \left\lfloor \tfrac{2}{\Delta} \cdot \big(\tfrac{\Delta}{4} \cdot (2\mu + 1) + e_\beta\big) \right\rceil \\ &\equiv \Delta \cdot \left\lfloor \big(\mu + \tfrac{1}{2} + \tfrac{2}{\Delta} \cdot e_\beta\big) \right\rceil \\ &\equiv \Delta \cdot \mu \pmod Q \end{aligned}$$

since $\left|\frac{2}{\Delta} \cdot e_\beta\right| < (2/\Delta) \cdot (\Delta/4) = 1/2$. By Theorem 2.6, after Line 2, we obtain

$$\mathrm{Dec}_{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \bar{b}) \equiv f_{\mathrm{eval}}\big(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d)\big) + e'_\beta \equiv \Delta \cdot \mu + e'_\beta$$

for some $e'_\beta \in (-\beta, \beta)$.

Case II: $\Delta \cdot \mu + \bar{e} \in [Q/2, Q-1]$

We have

$$\begin{aligned} f_C(\Delta \cdot \mu + \bar{e}) &= -\tfrac{\Delta}{4} \cdot \left(2\left\lfloor \tfrac{\Delta \cdot \mu + \bar{e}}{\Delta} \right\rfloor - p + 1\right) \bmod Q \\ &= -\tfrac{\Delta}{4} \cdot (2\mu - p + 1) \bmod Q \\ &= -\tfrac{\Delta}{2} \cdot \mu + \tfrac{Q}{4} - \tfrac{\Delta}{4} \bmod Q \end{aligned}$$

so that, after Line 1,

$$\begin{aligned} \mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) &\equiv f_C(\Delta \cdot \mu + \bar{e}) + e_\beta \\ &\equiv -\tfrac{\Delta}{2} \cdot \mu + \tfrac{Q}{4} - \tfrac{\Delta}{4} + e_\beta \pmod Q \end{aligned} \tag{5}$$

for some $e_\beta \in (-\beta, \beta)$.

Also, for any $x \in [Q/2, Q-1]$, we have $f_C(x) \in [(3Q + \Delta)/4, Q - \frac{\Delta}{4}]$, so that $f_C(x) + e_\beta \in [3Q/4, Q-1]$ whenever $e_\beta \in (-\Delta/4, \Delta/4)$ and the rightmost side of (5) thus lives in $[3Q/4, Q-1]$.

Since $f_C(\Delta \cdot \mu + \bar{e}) = -\frac{\Delta}{2} \cdot \mu + \frac{Q}{4} - \frac{\Delta}{4} \pmod Q$, we have (over $\mathbb{Q}$)

$$\begin{aligned} \tfrac{2}{\Delta} \cdot \big(Q - f_C(\Delta \cdot \mu + \bar{e}) - e_\beta\big) &= \tfrac{2}{\Delta} \cdot \big(\tfrac{\Delta}{2} \cdot \mu + \tfrac{3Q}{4} + \tfrac{\Delta}{4} - e_\beta + k \cdot Q\big) \\ &= \mu + \tfrac{3p}{2} + \tfrac{1}{2} - \underbrace{\tfrac{2}{\Delta} \cdot e_\beta}_{\in(-\frac{1}{2}, \frac{1}{2})} + 2k \cdot p \end{aligned}$$

for some integer $k \in \mathbb{Z}$. By rounding, it comes that

$$\left\lfloor \tfrac{2}{\Delta} \cdot \big(Q - f_C(\Delta \cdot \mu + \bar{e}) - e_\beta\big) \right\rceil = \mu + \tfrac{3p}{2} + 2k \cdot p$$

(still over $\mathbb{Q}$) and

$$\begin{aligned} \left\lfloor \tfrac{2}{\Delta} \cdot \big(Q - f_C(\Delta \cdot \mu + \bar{e}) - e_\beta\big) \right\rceil + \tfrac{p}{2} &\equiv \mu + 2(k+1) \cdot p \\ &\equiv \mu \pmod p \ . \end{aligned}$$

Given that $\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) \bmod Q \in [3Q/4, Q-1]$ after Line 1, we have

$$\begin{aligned} &f_{\mathrm{eval}}\big(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d) \bmod Q\big) = f_{\mathrm{eval}}\big(f_C(\Delta \cdot \mu + \bar{e}) + e_\beta\big) \\ &\quad = \Delta \cdot \Big(\left\lfloor \tfrac{2}{\Delta} \cdot \big(Q - f_C(\Delta \cdot \mu + \bar{e}) - e_\beta\big) \right\rceil + \tfrac{p}{2} \bmod p\Big) \bmod Q \\ &\quad = \Delta \cdot \mu \bmod Q \ . \end{aligned}$$

Therefore, after Line 2, we get

$$\mathrm{Dec}_{\boldsymbol{s}}(\bar{\boldsymbol{a}}, \bar{b}) \equiv f_{\mathrm{eval}}\big(\mathrm{Dec}_{\boldsymbol{s}}(\boldsymbol{c}, d)\big) + e'_\beta \equiv \Delta \cdot \mu + e'_\beta$$

for some $e'_\beta \in (-\beta, \beta)$, as claimed.

□

We need to assume that $Q = 2N$, where N is the ring dimension (i.e., the degree of the cyclotomic polynomial $X^N + 1$ used in the GGSW scheme encrypting the PRF seed $\boldsymbol{s}$), in order to evaluate the negacyclic functions using look-up tables of reasonable size. As an example, suppose that $Q = 2 \times 2048$, $p = 2^5$ and $n = 761$ with a binary secret key (which is the case for the `PARAM_MESSAGE_2_CARRY_2` parameters from TFHE-rs). In this case, the lattice estimator suggests around 200 bits of security for the $\mathrm{LWR}_{n,m,2N,p,U(\{0,1\})}$ for

[5] Recall that, in this section, elements of $\mathbb{Z}_Q$ are viewed as unsigned integers with a representative in $[0, Q-1]$.

unbounded number of samples m, meaning that the LWR problem of interest is concretely hard.

Remark 3.2. We note that Theorem 2.6 applies to the bootstrapping algorithm of [55], which does not require secret keys to be small. This allows homomorphically evaluating the PRF described in (2) when its secret key is sampled from a wide discrete Gaussian distribution (rather than a uniform binary/ternary distribution). The methodology is the same as above.

4 Extension to RLWR-based PRFs Using BFV

The approach of Section 3 extends to homomorphically evaluate the ring analogue of the PRF recalled in Section 2.2. We assume a random oracle $H\colon \{0,1\}^\ell \to \mathcal{R}_q$ that ranges over the ring $\mathcal{R}_q$. Note that this section offers an alternative view of the homomorphic PRF evaluation in [31]. The difference in interpretation is that we bootstrap $(-\mathscr{a}, 0)$ directly, interpreting it as a noisy ciphertext whereas [31] scalar multiply an encryption of the seed $\mathscr{s}$ by $\mathscr{a}$ and then homomorphically round the result.

Recall that, in the notations of Theorem 2.3, the BFV FHE [36] involves ciphertexts of the form $(\mathscr{a}, \mathscr{a} \cdot \mathscr{s} + \Delta \cdot m + \text{noise})$, where $\mathscr{a} \sim U(\mathcal{R}_q)$ is a random ring element and $\mathscr{s} \sim \chi_s$ is the secret key sampled from some distribution over $\mathcal{R}$. The standard bootstrapping of BFV can be seen as a restricted programmable bootstrapping for the function $f(x) = \Delta \cdot \lceil x/\Delta \rfloor$ that only refreshes the input ciphertext. We can use the random oracle to encode the input x as a ring element $\mathscr{a} = H(x) \in \mathcal{R}_q$ and interpret $(-\mathscr{a}, 0) \in \mathcal{R}_q^2$ as a noisy BFV ciphertext of the form

$$(-\mathscr{a}, 0) = \big(-\mathscr{a},\ -\mathscr{a} \cdot \mathscr{s}\ +\ \Delta \cdot \mathscr{Q}\ +\ \underbrace{\mathscr{r}}_{\in [-\Delta/2, \Delta/2)}\big) \in \mathcal{R}_q^2\,,$$

where $\mathscr{Q}$ is the quotient obtained by Euclidean division and $\mathscr{r}$ is the remainder. By applying non-programmable bootstrapping techniques for BFV, we can homomorphically compute a low-noise encryption $(\mathscr{c}, \mathscr{d}) \in \mathcal{R}_q^2$ of the same plaintext $\mathscr{Q}$. Note that in order to enable the BFV bootstrapping of the high noise ciphertext, one should set q to be the intermediate modulus to avoid the modulus switching step [39]. Then, we obtain that

$$\mathscr{Q} = \big\lceil ((p/q) \cdot \mathscr{a} \cdot \mathscr{s} \bmod q) \big\rfloor = \big\lceil (p/q) \cdot (\mathscr{a} \cdot \mathscr{s} \bmod q) \big\rfloor \bmod p\,,$$

so that the bootstrapping algorithm outputs a BFV encryption $(\mathscr{c}, \mathscr{d})$ of the RLWR-based PRF $\big\lceil (p/q) \cdot (\mathscr{a} \cdot \mathscr{s} \bmod q) \big\rfloor$.

Using the bootstrapping techniques of BFV, we can thus obtain many pseudorandom slots in one bootstrap achieving very high *amortized* throughput. On the downside, BFV bootstrapping incurs a much higher latency (typically at least in the 10's of seconds) than TFHE. This latency may be prohibitive in applications such as transciphering.

5 Modified PRFs Supporting Homomorphic Evaluation Using Depth-1 Bootstrapping

In Section 3, the TFHE evaluator has to perform two sequential programmable bootstraps for each plaintext slot in order to evaluate non-negacyclic functions.

In this section, we modify the LWR-based PRF in such a way that it can be evaluated using only one PBS per plaintext slot, approximately halving the computation time. In this modified construction, we assume again that $Q = 2N$ where N is the ring dimension used in TFHE.

We start from the previous approach and view $(-\boldsymbol{a}, 0) \in \mathbb{Z}_{2N}^{n+1}$ as a highly noisy ciphertext that "decrypts" to $\big\lfloor (\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N)/\Delta \big\rfloor$ for $\Delta = 2N/p$ (in this case, we view the noise as a positive integer in $[0, \Delta)$ and use the floor function). In an attempt to achieve this decryption functionality, we apply Theorem 2.6 with the negacyclic function $f\colon \mathbb{Z}_{2N} \to \mathbb{Z}_q$ defined by:

$$\begin{aligned} f(x) &= \begin{cases} \Delta' \cdot \big\lfloor \frac{p}{N} \cdot x \big\rfloor \bmod q & \text{if } x \in \big[0, N-1\big] \\ -\Delta' \cdot \big\lfloor \frac{p}{N} \cdot (x - N) \big\rfloor \bmod q & \text{if } x \in \big[N, 2N-1\big] \end{cases} \\ &= (-1)^{\mathrm{msb}(x)} \cdot \Delta' \cdot \big\lfloor \tfrac{p}{N} \cdot (x \bmod N) \big\rfloor \end{aligned} \tag{6}$$

where $\Delta' = q/2N \in \mathbb{Z}$. By Theorem 2.6, the $\mathrm{Eval}_{\mathrm{pp}}$ evaluation algorithm outputs a ciphertext encrypting

$$\mathsf{PRF}_{\boldsymbol{s}}(x) = (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N)} \Big\lfloor \tfrac{p}{N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod N\big) \Big\rfloor \bmod p \tag{7}$$

where $\boldsymbol{a} \triangleq \boldsymbol{a}(x) = H(x) \in (\mathbb{Z}_{2N})^n$ and $\boldsymbol{s} \in \{0,1\}^n$ is the LWE secret key. Note that we hash x onto $\mathbb{Z}_{2N}$ even though the inner product inside the rounding function is reduced modulo N. Another difference with the original construction is that the underlying rounding function is the floor function $\lfloor \cdot \rfloor$ (whereas the initial PRF can use any rounding function like floor, ceiling or the nearest integer although its depth-2 evaluation works for the $\lceil \cdot \rfloor$ rounding function). We note that this rounding function was used recently in [40] for similarly small moduli, and that there is no reason to suspect that the use of $\lfloor \cdot \rfloor$ affects hardness [12, Sect. 2]. We later show that one can use the nearest integer rounding function if desired. Although the function (7) is not the standard PRF considered in [19, 40], we can still prove it pseudorandom via a reduction from the pseudorandomness of the *standard* LWR-based PRF.

In more detail, the $\mathrm{Eval}_{\mathrm{pp}}$ algorithm takes as input the bootstrapping keys $\mathsf{bsk}[j] = \mathrm{GGSW}_{\mathscr{s}'}(s_j)$, $1 \le j \le n$, each of which is a GGSW encryption (with ring dimension N) of the seed entry s_j under GLWE secret key $\mathscr{s}'$. We form the test polynomial $\mathscr{v}(X) = \sum_{i=0}^{N-1} v_i\, X^i \in \mathbb{Z}_p[X]/(X^N + 1)$ with coefficients defined as $v_i = \lfloor i \cdot p/N \rfloor \bmod p$ for each $i \in [0, N-1]$. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$. Then, due to the congruence $X^N = -1 \pmod{X^N + 1}$, we have

$$\begin{aligned} X^{-\sum_{j=1}^n s_j \cdot a_j \pmod{2N}} \cdot \mathscr{v}(X) = (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N)} \cdot \\ X^{-(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod N)} \cdot \mathscr{v}(X) \pmod{X^N + 1} \end{aligned} \tag{8}$$

since

$$\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N = (\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod N) + b \cdot N\,,$$

where $b = \mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N) \in \{0,1\}$. In the right-hand-side member of (8), the degree-0 coefficient is thus

$$\begin{aligned} &(-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N)} \cdot v_{\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod N} \\ &\quad = (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N)} \cdot \big\lfloor (\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod N) \cdot p/N \big\rfloor \bmod p \end{aligned}$$

which is the correct evaluation for $\boldsymbol{a} = H(x) \in \mathbb{Z}_{2N}^n$ in (7). Note that the blind rotation and sample extraction subroutines on the ciphertext $(-\boldsymbol{a}, 0)$ exactly run this procedure in the encrypted domain. In particular, the *phase* of $(-\boldsymbol{a}, 0)$ (i.e., $\langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod 2N$) picks out the correct coefficient of the test polynomial. To summarize, we have:

$\mathsf{Eval}_{\mathsf{pp}}(\mathsf{bsk}, x)$: Given public parameters $\mathsf{pp} = (n, N, p, q, \beta)$, an evaluation key bsk and an input $x \in \{0,1\}^{\ell}$, compute the input-dependent vector $\boldsymbol{a} = H(x) \in \mathbb{Z}_{2N}^{n}$ and output $\mathrm{SampleExtract} \circ \mathrm{BlindRotate}[f](-\boldsymbol{a}, 0)$ i.e., $\mathrm{Boot}[f]$ without the keyswitch.

Note that removing the keyswitch outputs an LWE ciphertext of the PRF encrypted under the LWE secret key $\boldsymbol{s}'$ consisting of the coefficients in $\mathfrak{s}'$. One can then keyswitch this ciphertext to the TFHE secret key in the wider application if required.

Remark 5.1. Although we have presented the case where the PRF key has the same form as the LWE secret key used in TFHE, other alternatives are available. Since the modified LWR PRF is as secure as an LWR PRF, the required LWR dimension n_{LWR} can be smaller than the LWE dimension n used in TFHE. This allows us to use a $\mathsf{bsk}[j]$ for $1 \le j \le n_{\mathrm{LWR}}$ in our construction which can significantly increase efficiency as fewer blind rotation steps are carried out. Furthermore, since we do not perform any modulus switch from q to $2N$ unlike in TFHE, there is a possibility to reduce N while increasing the module rank for efficiency. When doing this, the hardness of the corresponding LWR problem must be checked since a smaller N induces a smaller deterministically generated LWR noise (of magnitude $< N/p$). Note that it would be useful in terms of memory requirement to reuse the TFHE bootstrapping key for PRF evaluation, i.e. to set the PRF seed to be the TFHE LWE key, but this would violate the principle of key separation.

5.1 Pseudorandomness of the Modified PRF

We have shown that one can evaluate the function

$$\mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N)} \left\lfloor \tfrac{p}{N} \cdot \left(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\right)\right\rfloor \pmod{p}$$

with depth-1 programmable bootstrapping. However, we must also show that $\mathsf{PRF}_{\boldsymbol{s}}$ is indeed pseudorandom as it is a *modified version* of the standard LWR PRF described in Section 2.2. As a standard TFHE parameter choice, we assume that $N = 2^{\ell_N}$ and $p = 2^{\ell_p}$ where $k \triangleq \ell_N - \ell_p > 0$.

Consider any $y \in \mathbb{Z}$ bounded in absolute value by $2N \cdot B$ for appropriately large B. It is easy to see that $y \bmod 2N$ is simply the lowest $\ell_N + 1$ bits of $y + 2NB$. Further, $y \bmod N$ is the lowest ℓ_N bits of $y + 2NB$. We can additionally interpret the operation on y described by

$$\left\lfloor \tfrac{2p}{2N} \cdot (y \bmod 2N)\right\rfloor \bmod 2p \tag{9}$$

in terms of operations on bits too. In particular, one can think of the above operation as taking the ℓ_N+1 bottom bits of $y+2NB$ and then dropping the k least significant bits. This leaves $\ell_N + 1 - k = \ell_p + 1$ bits which represents an integer modulo $2p$. This interpretation implies:

Observation 5.2. *For $N > p$ both powers-of-two and any $y \in \mathbb{Z}$,*

$$\mathrm{msb}(y \bmod 2N) = \mathrm{msb}\Big(\left\lfloor \tfrac{2p}{2N} \cdot (y \bmod 2N)\right\rfloor \bmod 2p\Big) .$$

Changing $(2N, 2p)$ to (N, p) in Eq. (9) changes the operation to "take ℓ_N bottom bits of $y + 2NB$ and then drop the k least significant bits". Therefore, we have the following:

Observation 5.3. *For $N > p$ both powers-of-two and any $y \in \mathbb{Z}$, one can compute $\left\lfloor \frac{p}{N} \cdot (y \bmod N)\right\rfloor \bmod p$ from the bits of the quantity in* (9) *by simply dropping the most significant bit.*

With these two observations, we prove that the modified $\mathsf{PRF}_{\boldsymbol{s}}$ is as secure as a standard LWR-based PRF (denoted as $G_{\boldsymbol{s}}$), which is identical to the one recalled in Section 2.2 except that the rounding function $\lceil \cdot \rfloor$ is replaced by $\lfloor \cdot \rfloor$.

Lemma 5.4. *Assume $p = 2^{\ell_p}$ and $N = 2^{\ell_N}$ are powers-of-two such that $k \triangleq \ell_N - \ell_p > 0$ and let $\mathcal{S}$ be a distribution with support on $\mathbb{Z}^n$. Take a random function $H\colon \{0,1\}^* \to (\mathbb{Z}_{2N})^n$ and assume that $G_{\boldsymbol{s}}\colon \{0,1\}^* \to \mathbb{Z}_{2p}$,*

$$G_{\boldsymbol{s}}(x) \triangleq \left\lfloor \tfrac{2p}{2N} \cdot \left(\langle H(x), \boldsymbol{s}\rangle \bmod 2N\right)\right\rfloor \bmod 2p$$

is a pseudorandom function for seeds $\boldsymbol{s} \sim \mathcal{S}$. Then, the function $\mathsf{PRF}_{\boldsymbol{s}}$: $\{0,1\}^* \to \mathbb{Z}_p$,

$$\mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq (-1)^{\mathrm{msb}(\langle H(x), \boldsymbol{s}\rangle \bmod 2N)} \cdot \left\lfloor \tfrac{p}{N} \cdot \left(\langle H(x), \boldsymbol{s}\rangle \bmod N\right)\right\rfloor \bmod p$$

is also a pseudorandom function for seeds $\boldsymbol{s} \sim \mathcal{S}$.

Proof. We describe a reduction $\mathcal{A}$ that attempts to build a PPT PRF distinguisher for G from any PPT distinguisher $\mathcal{D}$ for PRF. The reduction is as follows:

- When $\mathcal{D}$ wishes to query its oracle on input x, $\mathcal{A}$ forwards the request on to its challenger, receiving $g \in \mathbb{Z}_{2p}$ in response.
- Define $g_0 \triangleq \mathrm{msb}(g)$ and g' to be the integer in $\{0, \dots, p-1\}$ resulting from dropping the MSB of g. $\mathcal{A}$ sends $f \triangleq (-1)^{g_0} \cdot g' \bmod p$ back to $\mathcal{D}$ in response to the query x.
- $\mathcal{A}$ ultimately outputs whatever $\mathcal{D}$ does.

When $\mathcal{A}$'s challenger is returning uniform values for g, $\mathcal{A}$'s response f that is sent to $\mathcal{D}$ is clearly uniform. On the other hand, when $\mathcal{A}$'s challenger is using G_{s} to compute g, we can use Theorem 5.2 to show that the exponent of (-1) is correct and Theorem 5.3 to show that the remaining term (i.e., g') is correctly computed for $\mathsf{PRF}_{\boldsymbol{s}}$. Therefore, $\mathcal{A}$ perfectly simulates $\mathcal{D}$'s PRF challenger which implies that $\mathcal{D}$'s advantage against the pseudorandomness of $\mathsf{PRF}_{\boldsymbol{s}}$ is equal to that of $\mathcal{A}$'s advantage against $G_{\boldsymbol{s}}$. By assumption on the pseudorandomness of $G_{\boldsymbol{s}}$, $\mathsf{PRF}_{\boldsymbol{s}}$ must also be pseudorandom. □

5.2 Replacing Floor with Nearest Integer Rounding

It is of course possible to evaluate the modified PRF in depth one when the rounding function is $\lceil \cdot \rfloor$ instead of $\lfloor \cdot \rfloor$. This only requires to modify the coefficients of the test polynomial $v(X)$ accordingly. In the interpretation of $(-\boldsymbol{a}, 0)$ as a noisy ciphertext, we need to replace $\lfloor \cdot \rfloor$ by $\lceil \cdot \rfloor$ in the negacyclic function of (6) and go back to our view of the "noise" as an integer in $[-\Delta/2, \Delta/2)$.

However, we need to slightly modify the argument proving the pseudorandomness of the resulting function. We may replace the definition of $\mathsf{PRF}_{\boldsymbol{s}}$ by

$$\mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq (-1)^{\mathrm{msb}(\langle H(x), \boldsymbol{s}\rangle \bmod 2N)} \cdot \left\lceil \tfrac{p}{N}\left(\langle H(x), \boldsymbol{s}\rangle \bmod N\right)\right\rfloor \bmod p \tag{10}$$

in Theorem 5.4 whilst changing the parameters (p, N) in the definition of $G_{\boldsymbol{s}}$. To see how, note that we can write $\lceil y \rfloor = \lfloor y \rfloor + b_y$ where b_y is the bit of y just below the fixed binary point. Essentially, the reduction in the proof needs an extra bit in order to convert the floor function to the rounding function (i.e., to simulate the function in Eq. (10)). In order to gain access to this bit, one needs to increase p in the definition of G_s by a factor of 2. The conclusion is the following lemma.

Lemma 5.5. *Assume $p = 2^{\ell_p}$ and $N = 2^{\ell_N}$ are powers-of-two such that $k \triangleq \ell_N - \ell_p - 1 > 0$ and let $\mathcal{S}$ be a distribution with support on $\mathbb{Z}^n$. Take a random function $H\colon \{0,1\}^* \to (\mathbb{Z}_{2N})^n$ and assume that $G_{\boldsymbol{s}} : \{0,1\}^* \to \mathbb{Z}_{4p}$,*

$$G_{\boldsymbol{s}}(x) \triangleq \left\lfloor \tfrac{4p}{2N} \cdot \big(\langle H(x), \boldsymbol{s}\rangle \bmod 2N\big)\right\rfloor \bmod 4p$$

is a pseudorandom function for seeds $\boldsymbol{s} \sim \mathcal{S}$. Then, the function $\mathsf{PRF}_{\boldsymbol{s}} : \{0,1\}^* \to \mathbb{Z}_p$,

$$\mathsf{PRF}_{\boldsymbol{s}}(x) \triangleq (-1)^{\mathrm{msb}(\langle H(x), \boldsymbol{s}\rangle \bmod 2N)} \cdot \left\lceil \tfrac{p}{N} \cdot \big(\langle H(x), \boldsymbol{s}\rangle \bmod N\big)\right\rfloor \bmod p$$

is also a pseudorandom function for seeds $\boldsymbol{s} \sim \mathcal{S}$. □

5.3 Reducing the Range for Padding

In TFHE and wider applications, it is often useful to have one or more padding bits in the plaintext space [26]. A common practice is to always leave the most significant bit of a plaintext as 0 when this plaintext has to be involved in further homomorphic computations. So far, we have not considered this issue.[6]

In the depth-2 construction of Section 3, a simple solution is to modify the test polynomial of the second PBS and shift the bits of all coefficients to the right. In the depth-1 case, we cannot do this since it would unsuitably interfere with the $(-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N)}$ factor during the blind rotation.

In the depth-1 case, we can address this problem by considering yet another modified PRF. As before, we assume that the full plaintext space has a power-of-two modulus $p = 2^{\ell_p}$. However, we now introduce a usable plaintext modulus $p' = 2^{\ell_{p'}} < p$ meaning that we have $\ell_p - \ell_{p'}$ padding bits (that should be set to 0). The modified PRF can then be described as

$$\begin{aligned}\mathsf{PRF}_{\boldsymbol{s}}(x) &\triangleq (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N)} \cdot \left(\left\lfloor \tfrac{p'}{2N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\big)\right\rfloor + \tfrac{1}{2}\right) \\ &\quad + \tfrac{p'-1}{2} \bmod p \qquad (11)\\ &= (-1)^{\mathrm{msb}(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N)} \cdot \left\lfloor \tfrac{p'}{2N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\big)\right\rfloor \\ &\quad + \tfrac{p'}{2} - \mathrm{msb}\big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N\big) \qquad (12)\end{aligned}$$

which ranges over $[0, p'-1]$. To evaluate this function, we can first apply Theorem 2.6 with $Q = 2N$ to the function

$$\begin{aligned} f\colon\ &\mathbb{Z}_{2N} \to \mathbb{Z}_q, \\ &x \mapsto f(x) \triangleq (-1)^{\mathrm{msb}(x)} \cdot \tfrac{q}{p} \cdot \left(\left\lfloor \tfrac{p'}{2N} \cdot (x \bmod N)\right\rfloor + \tfrac{1}{2}\right)\end{aligned}$$

which is negacyclic since

$$f(x+N) = (-1)\cdot f(x) = -f(x) \bmod 2N \qquad \forall x \in [0, N-1] \ .$$

Then, after the PBS, we can add the term $\frac{q}{p} \cdot \frac{p'-1}{2}$ to the evaluated ciphertext.[7] Due to the additive homomorphism, this yields an encryption of the correct PRF value (11).

We now argue the pseudorandomness of the function in (11). Note that for any $\boldsymbol{a}, \boldsymbol{s} \in \mathbb{Z}^n$,

$$\begin{aligned}\left\lfloor \tfrac{p'/2}{N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\big)\right\rfloor \bmod \tfrac{p'}{2} &= \left\lfloor \tfrac{p'/2}{N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\big)\right\rfloor \bmod p \\ &= \left\lfloor \tfrac{p'/2}{N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod N\big)\right\rfloor\end{aligned}$$

as the modular reduction $\bmod(p'/2)$ and p is inconsequential. By Theorem 5.3, if $N > p'/2$, one can compute the above by dropping the most significant bit of

$$\left\lfloor \tfrac{p'}{2N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N\big)\right\rfloor \ . \qquad (13)$$

Furthermore, by Theorem 5.2, if $N > p'/2$,

$$\mathrm{msb}\big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N\big) = \mathrm{msb}\Big(\left\lfloor \tfrac{p'}{2N} \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s}\rangle \bmod 2N\big)\right\rfloor\Big) \ . \qquad (14)$$

To prove the pseudorandomness property, we rely on the following lemma whose proof is similar to that of the un-padded case.

Lemma 5.6. *Assume $p' = 2^{\ell_{p'}}$ and $N = 2^{\ell_N}$ are powers of two such that $N > p'/2$ and let $\mathcal{S}$ a distribution with support on $\mathbb{Z}^n$. Let a random oracle $H\colon \{0,1\}^* \to (\mathbb{Z}_{2N})^n$ and assume that $G_{\boldsymbol{s}} : \{0,1\}^* \to \mathbb{Z}_{p'}$,*

$$G_{\boldsymbol{s}}(x) \triangleq \left\lfloor \tfrac{p'}{2N} \cdot \big(\langle H(x), \boldsymbol{s}\rangle \bmod 2N\big)\right\rfloor$$

is a pseudorandom function for $\boldsymbol{s} \sim \mathcal{S}$. Then, the function

$$\mathsf{PRF}'_{\boldsymbol{s}} : \{0,1\}^* \to \left\{\pm\tfrac{1}{2}, \pm\big(1+\tfrac{1}{2}\big), \pm\big(2+\tfrac{1}{2}\big), \ldots, \pm\big(\tfrac{p'}{2} - 1 + \tfrac{1}{2}\big)\right\}$$

$$\mathsf{PRF}'_{\boldsymbol{s}}(x) \triangleq (-1)^{\mathrm{msb}(\langle H(x), \boldsymbol{s}\rangle \bmod 2N)} \cdot \left(\left\lfloor \tfrac{p'}{2N} \cdot \big(\langle H(x), \boldsymbol{s}\rangle \bmod N\big)\right\rfloor + \tfrac{1}{2}\right)$$

is a pseudorandom function where $\boldsymbol{s} \sim \mathcal{S}$.

Proof. We describe a reduction $\mathcal{A}$ that attempts to build a PPT PRF distinguisher for G from any PPT distinguisher $\mathcal{D}$ for PRF'. The reduction is as follows:

- When $\mathcal{D}$ wishes to query its oracle on input x, $\mathcal{A}$ forwards the request on to its challenger, receiving $g \in \mathbb{Z}_{p'}$ in response.
- Define $g_0 \triangleq \mathrm{msb}(g) \in \{0,1\}$ and let g' be the integer in $\{0, \ldots, p'/2 - 1\}$ resulting from dropping the MSB of g. $\mathcal{A}$ sends $f \triangleq (-1)^{g_0} \cdot (g' + \frac{1}{2})$ back to $\mathcal{D}$ in response to the query x.
- $\mathcal{A}$ ultimately outputs whatever $\mathcal{D}$ does.

When $\mathcal{A}$'s challenger is returning uniform values for g, $\mathcal{A}$'s response f that is sent to $\mathcal{D}$ is clearly uniform in the appropriate range as $\mathcal{A}$'s operations are invertible. On the other hand, when $\mathcal{A}$'s challenger is using $G_{\boldsymbol{s}}$ to compute g, we can use Eq. (14) to show that the exponent of (-1) is correct and Theorem 5.3 with the parametrization in (13) to show that the remaining term (i.e., g') is correctly computed for $\mathsf{PRF}'_{\boldsymbol{s}}$. Therefore, $\mathcal{A}$ perfectly simulates $\mathcal{D}$'s PRF challenger which implies that $\mathcal{D}$'s advantage against

[6] In the application to transciphering in Supplementary Material C, it is not necessary to keep the padding bit clear and we can use the full precision of the plaintext space.

[7] This does not quite correspond to adding $\frac{p'}{2} - \frac{1}{2}$ to the plaintext since $1/2$ is not defined modulo p. However, it still provides an encryption of the correct PRF evaluation (11) after the final addition.

the pseudorandomness of PRF'_s is equal to that of $\mathcal{A}$'s advantage against G_s. By assumption on the pseudorandomness of G_s, PRF'_s must also be pseudorandom. □

We complete the proof of pseudorandomness for the function defined in (11) by additively shifting the pseudorandom function PRF'_s from the above lemma.

6 Implementation and Performance

In order to test our depth-1 construction in practice, we use the TFHE-rs library (v0.6.1) using an AWS hpc7a.96xlarge instance with 4th Gen AMD EPYC processor, 768 GiB total RAM and using AVX512 on a single thread. We also provide benchmarks run on a 2022 Apple Macbook Pro with an Apple M2 chip and 8 GB RAM. In particular, we test the latency (specifically, the time required to perform the blind rotation step) of the homomorphic evaluation of the PRF in Section 5.1. To ensure the practical relevance of our results, we use practical TFHE parameter sets `PARAM_MESSAGE_2_CARRY_2` (where the bootstrapping key has size 23.9MB) and `PARAM_MESSAGE_1_CARRY_1` (with bootstrapping key size 11MB) instead of some bespoke parameter set. Our benchmarks do not include the computation of the hash value $H(x)$ and therefore mimic cases where the hash values have been precomputed e.g., transciphering where the server stores the hashes. In any case, the time taken to hash will be negligible compared to the runtime of homomorphic PRF evaluations, especially for short 128-bit or 256-bit inputs. Note that the plaintext space for `PARAM_MESSAGE_2_CARRY_2` is effectively $\mathbb{Z}_p$ for $p = 2^5$ as there are 2 "carry" bits, 2 "message" bits and a padding bit (equaling 5 bits in total). Note that the nomenclature arises from the fact that TFHE-rs is designed to implement large integer arithmetic. Further, the ring dimension used is $N = 2048 = 2^{11}$. Using the optimization outlined in Theorem 5.1 and Theorem 5.4, we require that $\mathrm{LWR}_{n_{\mathrm{LWR}},m,Q=2N,2p,U(\{0,1\})}$ is hard. Using the lattice estimator for unbounded m and modeling LWR as LWE with uniform rounding noise, we conclude that we may choose $n_{\mathrm{LWR}} = 445$ for an estimated 128 bits of security. Taking n_{LWE} to be the TFHE LWE dimension leads to a PRF evaluation key that has a size of approximately $n_{\mathrm{LWR}}/n_{\mathrm{LWE}} = 445/742 \approx 60\%$ of the size of a regular bootstrapping key. Making this choice leads to a latency of 6.0328 ms (averaged over 60 seconds worth of trials). Stated differently, we obtain a throughput of around 829 encrypted pseudorandom bits on a single thread. Naturally, one can increase this throughput by using multiple threads.

For the `PARAM_MESSAGE_1_CARRY_1` parameter set, the plaintext space is effectively set to $p = 2^3$ and the ring dimension is $N = 512$ leading to $n_{\mathrm{LWR}} = 409$. The PRF evaluation key size in this case is also approximately $n_{\mathrm{LWR}}/n_{\mathrm{LWE}} = 409/702 \approx 60\%$ of the bootstrapping key. The latency in the depth-1 construction is around 2.8029 ms (averaged over 60 seconds) leading to around 1070 pseudorandom bits per second on a single thread. The results are summarized in Table 1.

Further Optimization. As mentioned in Remark 5.1, our conditions on N are different to those in TFHE parameter selection and reducing the size of N can offer improvements in efficiency. In particular, we do not have to worry about any modulus switching error from q to $2N$ in our construction. We do however have to worry about choosing N and n_{LWR} so that LWR holds with respect to moduli $2N$ and $2p$. We describe our strategy for optimizing parameters next, defining k to be the module rank (as defined in Definition 2.5) in the TFHE GLWE assumption. Suppose that we are initially using parameters (N, k, n_{LWR}). Then, we can move to some $k', N' = N/2^{\kappa}$ (for some integer $\kappa < \log_2(N)$) and set n'_{LWR} such that the LWR assumption in Theorem 5.4 holds.

Table 1: Single threaded experimental results. The first reported result is on a hpc7a.96xlarge instance whereas the second is on an Apple Macbook Pro.

Parameter set	`MESSAGE_1_CARRY_1`	`MESSAGE_2_CARRY_2`
Plaintext bits	3	5
Latency (ms)	2.803 / 3.714	6.033 / 8.187
Throughput (bits/s)	**1070** / 808	829 / 611
Bootstrap Key	11.0 MB	23.9 MB
PRF Eval Key	**6.4 MB**	13.9 MB

Next, we must ensure that the new parameters are chosen so that the output of our construction looks like a bootstrapped TFHE `PARAM_MESSAGE_X_CARRY_X` ciphertext, particularly in terms of the error size. A detail is that these parameter sets use the more efficient "keyswitch then blind rotate" [14] pattern: i.e., the output to the PBS is a $(k \cdot N + 1)$-dimensional LWE ciphertext. On the other hand, the output to our optimized scheme would be a $(k' \cdot N' + 1)$-dimensional LWE ciphertext. We will always assume that whenever $(k', N') \neq (k, N)$, we use distinct secret keys. In other words, we do not share a secret key between a GLWE in dimensions (k', N') and (k, N). This choice may be overly conservative, but may potentially allow for a less heuristic security guarantee. Then, to summarize we pick parameters to ensure that a blind rotation followed by a "$k'N'$ to kN" keyswitch results in a ciphertext under the correct key, with the noise level of a bootstrapped ciphertext. We note that when using the "blind rotate then keyswitch" pattern, the $k'N'$ to kN keyswitch is unnecessary as one would simply keyswitch directly down to the LWE dimension to obtain a bootstrapped ciphertext. However, this pattern generally leads to an overall less efficient FHE application.

Using the optimization techniques from [14], we obtain new parameters and run benchmarks. Unfortunately, the optimized `PARAM_MESSAGE_1_CARRY_1` setting did not improve throughput, even when ignoring the $k'N'$ to kN keyswitch. Therefore, we do not report on this parameter set. However, the optimization and improved performance of the `PARAM_MESSAGE_2_CARRY_2` is reported in Table 2. A good choice for the dimension N' in terms of latency and key size appears to be 512. At this dimension, the throughput increases by 16% (or 44%) on the hpc7a.96xlarge (respectively, laptop) whereas the evaluation key shrinks by around 5 MB or 36% compared to results in Table 1. As can be seen in the latter table, when N' becomes small, the LWR dimension increases dramatically hindering performance. Note that optimizing without the $k'N'$ to kN keyswitch does not appear to change the throughput here either. In particular, removing the keyswitch still does not allow us to pick $k'N' < kN$ or improve the blind rotation decomposition parameters (which are already optimal in `PARAM_MESSAGE_2_CARRY_2`).

We note that the key compression techniques of [55] could also be applied to compress the size of the PRF evaluation key. However, due to the tight noise constraints considered when optimizing the parameters, we do not expect their techniques to be directly applicable since they result in additional noise in the PRF evaluation key. Instead, to utilize these techniques, it would be necessary to use larger parameters, as in [55, 4]

Table 2: Single threaded experimental results with "optimized" parameters for `PARAM_MESSAGE_2_CARRY_2`. The first reported result is on a hpc7a.96xlarge instance whereas the second is on an Apple Macbook Pro.

Parameter set	`Opt-2-2-256`	`Opt-2-2-512`	`Opt-2-2-1024`
$(N', k', n'_{\mathrm{LWR}})$	$(256, 8, 980)$	$(512, 4, 455)$	$(1024, 2, 455)$
Plaintext bits	5	5	5
Latency (ms)	14.267 / 13.388	5.205 / 5.675	5.156 / 6.379
Throughput (bits/s)	350 / 373	961 / 881	**970** / 784
PRF Eval Key	17.2 MB	**8.9 MB**	10.7 MB

Comparison With the State of the Art. We can compare the experimental results of our technique with other state-of-the-art solutions. In our case, we consider two of our results: the `MESSAGE_1_CARRY_1` parameter set with performance measured on the hpc7a.96xlarge and also the `Opt-2-2-512` parameter set measured on the Apple Macbook Pro. For comparison, we focus only on other TFHE-based solutions, and do not consider works based on BFV or CKKS. In particular, we compare against Transistor [13], and Trivium and Kreyvium (which were evaluated in the TFHE setting in [10]).

We are interested in comparing the performance in multiple ways: considering whether a setup phase is required, the latency and throughput, and also the size of any additional evaluation key material beyond that typically used in TFHE.

Table 3: Comparison of our techniques and existing state of the art techniques based on the TFHE scheme.

Technique	Setup?	Latency (ms)	Throughput	Eval Key
`MESSAGE_1_CARRY_1`	no	2.803	1070	6.4 MB
`Opt-2-2-512`	no	5.675	881	8.9 MB
Transistor [13]	no	251	65	780 B
Trivium [10]	yes	121	529	36.4 MB
Kreyvium [10]	yes	150	427	36.4 MB

The latency and throughput of our solution outperforms both Transistor, and Trivium/Kreyvium (which also requires a warm-up phase in the range of 2-3s). We note that Transistor requires less public material (780 B) due to their requirement for only 96 LWE-encrypted ciphertexts, which is more compact than the n_{LWR} GGSW ciphertexts required in our solution. However, we note that the additional key material required in our solution is smaller than the associated bootstrapping key used in TFHE. Moreover, our experimental parameter sets were optimized with latency in mind: it is possible to explore a trade-off between the size of the additional required key material and latency. Further, we note that approach relies on standard LWR assumptions as opposed to bespoke block/stream cipher security.

Acknowledgments

We thank Chris Peikert and Damien Stehlé for their comments/corrections regarding related works.

References

[1] Shweta Agrawal, Shafi Goldwasser, and Saleet Mossel. 2021. Deniable fully homomorphic encryption from learning with errors. In *Advances in Cryptology – CRYPTO 2021, Part II (Lecture Notes in Computer Science, Vol. 12826)*, T. Malkin and C. Peikert (Eds.). Springer, 641–670. doi:10.1007/978-3-030-84245-1_22

[2] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. 2022. Round-optimal lattice-based threshold signatures, revisited. In *Automata, Languages, and Programming (ICALP 2022) (LIPIcs, Vol. 229)*, M. Bojanczyk, E. Merelli, and D. P. Woodruff (Eds.). Dagstuhl Publishing, 8:1–8:20. doi:10.4230/LIPICS.ICALP.2022.8

[3] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. 2018. Estimate all the LWE, NTRU schemes!. In *Security and Cryptography for Networks (SCN 2018) (Lecture Notes in Computer Science, Vol. 11035)*, D. Catalano and R. De Prisco (Eds.). Springer, 351–367. doi:10.1007/978-3-319-98113-0_19

[4] Martin R. Albrecht, Alex Davidson, Amit Deo, and Daniel Gardham. 2024. Crypto dark matter on the torus - Oblivious PRFs from shallow PRFs and TFHE. In *Advances in Cryptology – EUROCRYPT 2024, Part VI (Lecture Notes in Computer Science, Vol. 14656)*, M. Joye and G. Leander (Eds.). Springer, 447–476. doi:10.1007/978-3-031-58751-1_16

[5] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of learning with errors. *J. Math. Cryptol.* 9, 3 (2015), 169–203. doi:10.1515/jmc-2015-0016

[6] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. 2015. Ciphers for MPC and FHE. In *Advances in Cryptology – EUROCRYPT 2015, Part I (Lecture Notes in Computer Science, Vol. 9056)*, E. Oswald and M. Fischlin (Eds.). Springer, 430–454. doi:10.1007/978-3-662-46800-5_17

[7] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. 2013. Learning with rounding, revisited. In *Advances in Cryptology – CRYPTO 2013, Part I (Lecture Notes in Computer Science, Vol. 8042)*, R. Canetti and J. A. Garay (Eds.). Springer, 57–74. doi:10.1007/978-3-642-40041-4_4

[8] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. 2022. Chaghri: A FHE-friendly block cipher. In *2022 ACM SIGSAC Conference on Computer and Communications Security*, H. Yin et al. (Eds.). ACM Press, 139–150. doi:10.1145/3548606.3559364

[9] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. 2023. HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In *Advances in Cryptology – CRYPTO 2023, Part IV (Lecture Notes in Computer Science, Vol. 14084)*, H. Handschuh and A. Lysyanskaya (Eds.). Springer, 37–69. doi:10.1007/978-3-031-38551-3_2

[10] Thibault Balenbois, Jean-Baptiste Orfila, and Nigel P. Smart. 2023. Trivial transciphering with Trivium and TFHE. In *11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2023)*, M. Brenner, A. Costache, and K. Rohloff (Eds.). ACM Press, 69–78. doi:10.1145/3605759.3625255

[11] Abhishek Banerjee and Chris Peikert. 2014. New and improved key-homomorphic pseudorandom functions. In *Advances in Cryptology – CRYPTO 2014 (Lecture Notes in Computer Science, Vol. 8616)*, J. A. Garay and R. Gennaro (Eds.). Springer, 353–370. doi:10.1007/978-3-662-44371-2_20

[12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. 2012. Pseudorandom functions and lattices. In *Advances in Cryptology – EUROCRYPT 2012 (Lecture Notes in Computer Science, Vol. 7237)*, D. Pointcheval and T. Johansson (Eds.). Springer, 719–737. doi:10.1007/978-3-642-29011-4_42

[13] Jules Baudrin, Sonia Belaïd, Nicolas Bon, Christina Boura, Anne Canteaut, Gaëtan Leurent, Pascal Paillier, Léo Perrin, Matthieu Rivain, Yann Rotella, and Samuel Tap. 2025. Transistor: A TFHE-friendly stream cipher. In *Advances in Cryptology – CRYPTO 2025, Part V (Lecture Notes in Computer Science, Vol. 16004)*, S. F. Kamara and Y. Tauman Kalai (Eds.). Springer, 522–555. doi:10.1007/978-3-032-01901-1_17

[14] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2023. Parameter optimization and larger precision for (T)FHE. *J. Cryptol.* 36, 3 (2023), 28. doi:10.1007/S00145-023-09463-5

[15] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. 2016. On the hardness of learning with rounding over small modulus. In *Theory of Cryptography (TCC 2016), Part I (Lecture Notes in Computer Science, Vol. 9562)*, E. Kushilevitz and T. Malkin (Eds.). Springer, 209–224. doi:10.1007/978-3-662-49096-9_9

[16] Andrej Bogdanov and Alon Rosen. 2017. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography*, Y. Lindell (Ed.). Springer, Chapter 3, 79–158. doi:10.1007/978-3-319-57048-8_3

[17] Nicolas Bon, David Pointcheval, and Matthieu Rivain. 2024. Optimized homomorphic evaluation of Boolean functions. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2024, 3 (2024), 302–341. doi:10.46586/tches.v2024.i3.302-341

[18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. 2018. Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology – CRYPTO 2018, Part I (Lecture Notes in Computer Science, Vol. 10991)*, H. Shacham and A. Boldyreva (Eds.). Springer, 565–596. doi:10.1007/978-3-319-96884-1_19

[19] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. 2013. Key homomorphic PRFs and their applications. In *Advances in Cryptology – CRYPTO 2013, Part I (Lecture Notes in Computer Science, Vol. 8042)*, R. Canetti and J. A. Garay (Eds.). Springer, 410–428. doi:10.1007/978-3-642-40041-4_23

[20] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology – CRYPTO 2012 (Lecture Notes in Computer Science, Vol. 7417)*, R. Safavi-Naini and R. Canetti (Eds.). Springer, 868–886. doi:10.1007/978-3-642-32009-5_50

[21] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. 2019. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *Theory of Cryptography (TCC 2019), Part II (Lecture Notes in Computer Science, Vol. 11892)*, D. Hofheinz and A. Rosen (Eds.). Springer, 407–437. doi:10.1007/978-3-030-36033-7_16

[22] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *3rd Innovations in Theoretical Computer Science (ITCS 2012)*, S. Goldwasser (Ed.). ACM Press, 309–325. doi:10.1145/2090236.2090262

[23] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. 2016. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In *Fast Software Encryption (FSE 2016) (Lecture Notes in Computer Science, Vol. 9783)*, T. Peyrin (Ed.). Springer, 313–333. doi:10.1007/978-3-662-52993-5_16

[24] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017, Part I (Lecture Notes in Computer Science, Vol. 10624)*, T. Takagi and T. Peyrin (Eds.). Springer, 409–437. doi:10.1007/978-3-319-70694-8_15

[25] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* 33, 1 (2020), 34–91. doi:10.1007/s00145-019-09319-x

[26] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2020. CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE. In *8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2020)*, M. Brenner and T. Lepoint (Eds.). Leibniz Universität IT Services, 57–63. doi:10.25835/0072999

[27] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private information retrieval. *J. ACM* 45, 6 (1998), 965–981. doi:10.1145/293347.293350

[28] Carlos Cid, John Petter Indrøy, and Håvard Raddum. 2022. FASTA: A stream cipher for fast FHE evaluation. In *Topics in Cryptology – CT-RSA 2022 (Lecture Notes in Computer Science, Vol. 13161)*, S. D. Galbraith (Ed.). Springer, 451–483. doi:10.1007/978-3-030-95312-6_19

[29] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. 2022. Towards case-optimized hybrid homomorphic encryption: Featuring the Elisabeth stream cipher. In *Advances in Cryptology – ASIACRYPT 2022, Part III (Lecture Notes in Computer Science, Vol. 13793)*, S. Agrawal and D. Lin (Eds.). Springer, 32–67. doi:10.1007/978-3-031-22969-5_2

[30] Eric Crockett and Chris Peikert. 2016. $\Lambda\circ\lambda$: Functional lattice cryptography. In *2016 ACM SIGSAC Conference on Computer and Communications Security*, E. R. Weippl et al. (Eds.). ACM Press, 993–1005. doi:10.1145/2976749.2978402

[31] Eric Crockett, Chris Peikert, and Chad Sharp. 2018. ALCHEMY: A Language and Compiler for Homomorphic Encryption Made easY. In *2018 ACM SIGSAC Conference on Computer and Communications Security*, D. Lie et al. (Eds.). ACM Press, 1020–1037. doi:10.1145/3243734.3243828

[32] Morten Dahl, Clément Danjou, Daniel Demmler, Tore Frederiksen, Peter Ivanov, Marc Joye, Dragos Rotaru, Nigel Smart, and Louis Tremblay Thibault. 2023. *fhEVM: Confidential EVM smart contracts using fully homomorphic encryption.* White Paper. Zama. https://github.com/zama-ai/fhevm/raw/main/fhevm-whitepaper.pdf

[33] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. 2023. Pasta: A case for hybrid homomorphic encryption. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023, 3 (2023), 30–73. doi:10.46586/TCHES.V2023.I3.30-73

[34] Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT 2015, Part I (Lecture Notes in Computer Science, Vol. 9056)*, E. Oswald and M. Fischlin (Eds.). Springer, 617–640. doi:10.1007/978-3-662-46800-5_24

[35] Ehsan Ebrahimi and Anshu Yadav. 2024. Strongly secure universal thresholdizer. In *Advances in Cryptology – ASIACRYPT 2024, Part III (Lecture Notes in Computer Science, Vol. 15486)*, K.-M. Chung and Y. Sasaki (Eds.). Springer, 207–239. doi:10.1007/978-981-96-0891-1_7

[36] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144. https://ia.cr/2012/144

[37] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Charalampos Papamanthou. 2024. ThorPIR: Single server PIR via homomorphic Thorp shuffles. In *2024 ACM SIGSAC Conference on Computer and Communications Security*, D. Lie and E. Kirda (Eds.). ACM Press, 1448–1462. doi:10.1145/3658644.3690326

[38] Pierre-Alain Fouque, Benjamin Hadjibeyli, and Paul Kirchner. 2016. Homomorphic evaluation of lattice-based symmetric encryption schemes. In *Computing and Combinatorics (COCOON 2016) (Lecture Notes in Computer Science, Vol. 9797)*, T. N. Dinh and M. T. Thai (Eds.). Springer, 269–280. doi:10.1007/978-3-319-42634-1_22

[39] Robin Geelen and Frederik Vercauteren. 2023. Bootstrapping for BGV and BFV revisited. *J. Cryptol.* 36, 2 (2023), 12. doi:10.1007/S00145-023-09454-6

[40] Matthias Geihs and Hart Montgomery. 2024. LaKey: Efficient lattice-based distributed PRFs enable scalable distributed key management. In *33rd USENIX Security Symposium*, D. Balzarotti and W. Xu (Eds.). USENIX, 4319–4335. https://dl.acm.org/doi/10.5555/3698900.3699142

[41] Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme.* Ph. D. Dissertation. Stanford University. https://crypto.stanford.edu/craig

[42] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. 2015. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *J. Cryptol.* 28, 4 (2015), 820–843. doi:10.1007/S00145-014-9184-Y

[43] Craig Gentry, Shai Halevi, and Nigel P. Smart. 2012. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology – CRYPTO 2012 (Lecture Notes in Computer Science, Vol. 7417)*, R. Safavi-Naini and R. Canetti (Eds.). Springer, 850–867. doi:10.1007/978-3-642-32009-5_49

[44] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013, Part I (Lecture Notes in Computer Science, Vol. 8042)*, R. Canetti and J. A. Garay (Eds.). Springer, 75–92. doi:10.1007/978-3-642-40041-4_5

[45] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. 2023. Cryptanalysis of Elisabeth-4. In *Advances in Cryptology – ASIACRYPT 2023, Part III (Lecture Notes in Computer Science, Vol. 14440)*, J. Guo and R. Steinfeld (Eds.). Springer, 256–284. doi:10.1007/978-981-99-8727-6_9

[46] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. 1986. How to construct random functions. *J. ACM* 33, 4 (1986), 792–807. doi:10.1145/6490.6503

[47] Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang. 2023. Cryptanalysis of symmetric primitives over rings and a key recovery attack on Rubato. In *Advances in Cryptology – CRYPTO 2023, Part III (Lecture Notes in Computer Science, Vol. 14083)*, H. Handschuh and A. Lysyanskaya (Eds.). Springer, 305–339. doi:10.1007/978-3-031-38548-3_11

[48] Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Jooyoung Lee, and Mincheol Son. 2022. Rubato: Noisy ciphers for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2022, Part I (Lecture Notes in Computer Science, Vol. 13275)*, O. Dunkelman and S. Dziembowski (Eds.). Springer, 581–610. doi:10.1007/978-3-031-06944-4_20

[49] Jiahui He, Kai Hu, Hao Lei, and Meiqin Wang. 2024. Massive superpoly recovery with a meet-in-the-middle framework - Improved cube attacks on Trivium and Kreyvium. In *Advances in Cryptology – EUROCRYPT 2024, Part I (Lecture Notes in Computer Science, Vol. 14651)*, M. Joye and G. Leander (Eds.). Springer, 368–397. doi:10.1007/978-3-031-58716-0_13

[50] ISO/IEC 29192-3:2012. 2012. Information Technology – Security Techniques – Lightweight Cryptography, Part 3: Stream Ciphers. https://www.iso.org/standard/56426.html

[51] Marc Joye. 2022. SoK: Fully Homomorphic encryption over the [discretized] torus. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022, 4 (2022), 661–692. doi:10.46586/TCHES.V2022.I4.661-692

[52] Marc Joye. 2024. TFHE public-key encryption revisited. In *Topics in Cryptology – CT-RSA 2024 (Lecture Notes in Computer Science, Vol. 14643)*, E. Oswald (Ed.). Springer, 277–291. doi:10.1007/978-3-031-58868-6_11

[53] Jonathan Katz and Yehuda Lindell. 2020. *Introduction to Modern Cryptography* (3rd ed.). Chapman and Hall/CRC. doi:10.1201/9781351133036

[54] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* 75, 3 (2015), 565–599. doi:10.1007/S10623-014-9938-4

[55] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. 2023. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Advances in Cryptology – EUROCRYPT 2023, Part III (Lecture Notes in Computer Science, Vol. 14006)*, C. Hazay and M. Stam (Eds.). Springer, 227–256. doi:10.1007/978-3-031-30620-4_8

[56] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. 2023. Coefficient grouping: Breaking Chaghri and more. In *Advances in Cryptology – EUROCRYPT 2023, Part IV (Lecture Notes in Computer Science, Vol. 14007)*, C. Hazay and M. Stam (Eds.). Springer, 287–317. doi:10.1007/978-3-031-30634-1_10

[57] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. 2025. BitGC: Garbled circuits with 1 bit per gate. In *Advances in Cryptology – EUROCRYPT 2025, Part VII (Lecture Notes in Computer Science, Vol. 15607)*, S. Fehr and P.-A. Fouque (Eds.). Springer, 437–466. doi:10.1007/978-3-031-91098-2_16

[58] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. 2022. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. In *Advances in Cryptology – ASIACRYPT 2022, Part II (Lecture Notes in Computer Science, Vol. 13792)*, S. Agrawal and D. Lin (Eds.). Springer, 130–160. doi:10.1007/978-3-031-22966-4_5

[59] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On ideal lattices and learning with errors over rings. *J. ACM* 60, 6 (2013), 43:1–43:35. doi:10.1145/2535925

[60] Shihe Ma, Tairong Huang, Anyu Wang, Qixian Zhou, and Xiaoyun Wang. 2024. Fast and accurate: Efficient full-domain functional bootstrap and digit decomposition for homomorphic computation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2024, 1 (2024), 592–616. doi:10.46586/tches.v2024.i1.592-616

[61] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. 2019. Improved filter permutators for efficient FHE: Better instances and implementations. In *Progress in Cryptology – INDOCRYPT 2019 (Lecture Notes in Computer Science, Vol. 11898)*, F. Hao, S. Ruj, and S. Sen Gupta (Eds.). Springer, 68–91. doi:10.1007/978-3-030-35423-7_4

[62] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. 2016. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *Advances in Cryptology – EUROCRYPT 2016, Part I (Lecture Notes in Computer Science, Vol. 9665)*, M. Fischlin and J.-S. Coron (Eds.). Springer, 311–343. doi:10.1007/978-3-662-49890-3_13

[63] Parker Newton and Silas Richelson. 2023. A lower bound for proving hardness of learning with rounding with polynomial modulus. In *Advances in Cryptology – CRYPTO 2023, Part V (Lecture Notes in Computer Science, Vol. 14085)*, H. Handschuh and A. Lysyanskaya (Eds.). Springer, 805–835. doi:10.1007/978-3-031-38554-4_26

[64] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34:1–34:40. doi:10.1145/1568318.1568324

[65] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, R. A. DeMillo et al. (Eds.). Academic Press, 165–179. https://people.csail.mit.edu/rivest/pubs/RAD78.pdf

[66] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. 2023. A homomorphic AES evaluation in less than 30 seconds by means of TFHE. In *11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2023)*, M. Brenner, A. Costache, and K. Rohloff (Eds.). ACM Press, 79–90. doi:10.1145/3605759.3625260

[67] Benqiang Wei, Ruida Wang, Zhihao Li, Qinju Liu, and Xianhui Lu. 2023. Fregata: Faster homomorphic evaluation of AES via TFHE. In *Information Security (ISC 2023) (Lecture Notes in Computer Science, Vol. 14411)*, E. Athanasopoulos and B. Mennink (Eds.). Springer, 392–412. doi:10.1007/978-3-031-49187-0_20

Supplementary Material

A Pseudorandom Functions

In this section, we recall the standard definition of pseudorandom function families.

Definition A.1. Let λ be a security parameter and let $\mathcal{K} = \mathcal{K}(\lambda)$, $\mathcal{X} = \mathcal{X}(\lambda)$, $\mathcal{Y} = \mathcal{Y}(\lambda)$ denote a key-space, an input space and a range respectively. An efficiently computable function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is pseudorandom if no probabilistic polynomial time distinguisher has a non-negligible advantage as defined below. Let Ω be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. The advantage of a distinguisher $\mathcal{D}$ making Q evaluation queries is defined as

$$\mathbf{Adv}_Q^{\mathcal{D},\mathrm{prf}}(\lambda) := \left|\Pr[\mathcal{D}^{\mathsf{PRF}(K,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{F(\cdot)}(1^\lambda) = 1]\right| ,$$

where the probability is taken over the random choice of $K \hookleftarrow U(\mathcal{K})$ and $F \hookleftarrow U(\Omega)$ and the coin tosses of $\mathcal{D}$.

B GGSW Ciphertexts

Generalized GSW (GGSW) encryption is a natural extension of the scheme by Gentry, Sahai, and Waters [44] (in its ring version) to higher ranks. GGSW ciphertexts play a central role in the programmable bootstrapping of TFHE as they enable the external product of certain ciphertexts (as defined below). In particular, the bootstrapping keys are GGSW encryptions of private-key components.

The simplest way to view GGSW ciphertexts is through gadget decomposition of GLWE ciphertexts. Given a gadget vector $\mathscr{g} = (\mathscr{g}_1, \ldots, \mathscr{g}_\ell) \in \mathscr{R}_q^\ell$ and a GLWE ciphertext $\mathscr{c} \leftarrow \mathrm{GLWE}_{\mathscr{s}}(\mu) \in \mathscr{R}_q^{k+1}$ under private key $\mathscr{s} = (\mathscr{s}_1, \ldots, \mathscr{s}_k) \in \mathscr{R}^k$, the corresponding gadget GLWE ciphertext (usually indicated with a $'$) is defined as

$$\mathrm{GLWE}'_{\mathscr{s}}(\mu) \leftarrow \big(\mathrm{GLWE}_{\mathscr{s}}(\mathscr{g}_1 \cdot \mu), \ldots, \mathrm{GLWE}_{\mathscr{s}}(\mathscr{g}_\ell \cdot \mu)\big) \ .$$

This leveled encryption gives rise to a GGSW ciphertext; i.e.,

$$\mathrm{GGSW}_{\mathscr{s}}(\mu) \leftarrow \big(\mathrm{GLWE}'_{\mathscr{s}}(-\mathscr{s}_1 \cdot \mu), \ldots, \mathrm{GLWE}'_{\mathscr{s}}(-\mathscr{s}_k \cdot \mu), \\ \mathrm{GLWE}'_{\mathscr{s}}(\mu)\big) \ .$$

Importantly, the external product of a GLWE ciphertext and a GGSW ciphertext, denoted with $\circledast$, satisfies

$$\mathrm{GLWE}_{\mathscr{s}}(\mu_1) \circledast \mathrm{GGSW}_{\mathscr{s}}(\mu_2) = \mathrm{GLWE}_{\mathscr{s}}(\mu_1 \cdot \mu_2 + \mathscr{e}_1 \cdot \mu_2)$$

where $\mathscr{e}_1$ is the noise error present in $\mathrm{GLWE}_{\mathscr{s}}(\mu_1)$. The output of the external product is therefore a GLWE encryption of $\mu_1 \cdot \mu_2$ provided that message μ_2 is "small" so that $\|\mathscr{e}_1 \cdot \mu_2\|_\infty \approx \|\mathscr{e}_1\|_\infty$. This is the case for the TFHE bootstrapping keys, which are GGSW encryptions of key bits (i.e., values in $\{0,1\}$).

C On Transciphering Using TFHE

Consider a client wishing to send large amounts of data to the cloud that will eventually be used in an FHE application—think for example of image processing over encrypted data. Instead of sending FHE encryptions of the data on the cloud, transciphering allows the transmission of symmetric-key encryptions to dramatically reduce cloud bandwidth requirements. Specifically, the symmetric encryptions sent on the cloud do not need to be any larger than the original data, whereas FHE ciphertexts would be much larger. As an example of this application, a symmetric encryption of a message M can take the form $\big(x, M \oplus \mathsf{PRF}_k(x)\big)$, where x is a random string of sufficient length chosen by the sender.[8] Then, in order to obtain an FHE encryption of M, the cloud can use an FHE encryption of k to compute an FHE encryption of $\mathsf{PRF}_k(x)$. From this encryption of $\mathsf{PRF}_k(x)$, the cloud can *homomorphically* subtract $\mathsf{PRF}_k(x)$ to obtain an FHE encryption of M that can be computed on further. Note that other forms of symmetric encryption (e.g., certain block-cipher modes of operation) may also be used provided that PRF inputs remain public.

As a concrete example, we may consider our depth-1 construction of Section 5.1 (or Section 5.2). This yields a symmetric encryption scheme where the sender chooses a random $x \hookleftarrow U(\{0,1\}^\lambda)$ and encrypts $\boldsymbol{M} \in \mathbb{Z}_p^m$ using PRF secret key $k \triangleq \boldsymbol{s} \hookleftarrow U(\{0,1\}^n)$ by computing $(x, \boldsymbol{c}) = \big(x, \boldsymbol{M} + \mathsf{PRF}_{\boldsymbol{s}}(x) \bmod p\big)$, where

$$\forall i \in [m] \ : \ \big(\mathsf{PRF}_{\boldsymbol{s}}(x)\big)_i \triangleq (-1)^{\mathrm{msb}(\langle H(x,i),\boldsymbol{s}\rangle \bmod 2N)} \\ \cdot \left\lfloor \tfrac{p}{N} \cdot \big(\langle H(x,i), \boldsymbol{s}\rangle \bmod N\big) \right\rceil , \quad (^{**})$$

for a hash function $H\colon \{0,1\}^* \to \mathbb{Z}_p^n$ modeled as a random oracle. Then, the encryptor sends the ciphertext $(x, \boldsymbol{c})$ for storage. We assume that the evaluator has a copy of the corresponding public

[8] This construction satisfies the definition of CPA security (see, e.g., [53, Chapter 3]) for secret-key encryption schemes assuming that the underlying PRF is pseudorandom.

bootstrapping key $\mathsf{bsk} = \mathrm{GGSW}_{\mathfrak{s}'}(\boldsymbol{s})$ for some GLWE secret key $\mathfrak{s}' \in \mathcal{R}^{\kappa}$. From $(x, \boldsymbol{c})$ and bsk, the evaluator can compute

$$\left(\mathbf{A}, \boldsymbol{b} = \mathbf{A}^{\top}\bar{\mathbf{s}} + \boldsymbol{e} + \Delta \cdot \mathsf{PRF}_{\boldsymbol{s}}(x)\right) = \mathrm{Eval}_{\mathrm{pp}}(\mathsf{bsk}, x) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m \ ,$$

where $\bar{\mathbf{s}} \in \mathbb{Z}_q^n$ and $\|\boldsymbol{e}\|_\infty \leq \beta$ for the bound β of Theorem 2.6. Then, the evaluator obtains

$$(-\mathbf{A}, -\boldsymbol{b} + \Delta \cdot \boldsymbol{c} \bmod q)$$

which is an encryption of $\boldsymbol{M} \in \mathbb{Z}_p^m$ under the secret key $\bar{\mathbf{s}}$ since $(0, \Delta \cdot \boldsymbol{c})$ is a (trivial) encryption of $\boldsymbol{c}$.

We also observe that, as in stream-cipher-based solutions, we can dynamically decide to change the number m of plaintext blocks if the need arises without changing anything to the parameters or the key material. It only requires to update the number of indexes i when the encryptor computes (**).

C.1 Comparison to the Naive Solution

The main advantage of the proposed solution is that the encryption of m plaintexts in $\mathbb{Z}_p$ only requires sending a random seed x (typically, a 256-bit value) along with m values modulo p. This is much better than the plain solution that would send a matrix $\mathbf{A}$ of $m \times n$ entries modulo q plus m values modulo q (typically, n is of the order of 1000). This is even better than the folklore improved solution consisting in sending a seed σ for building matrix $\mathbf{A}$ along with m values modulo q. Our solution trades modulo-q values against modulo-p values, where $p \ll q$; typically, $p = 2^4$ and $q = 2^{64}$. Compared to the improved solution, this saves $m \cdot \log_2 q/p$ bits of transmission. For example, for (1024×1024) 8-bit gray-scale images, with $p = 16$ and $q = 2^{64}$ (and thus $m = 2^{21}$), this results in saving more than 10^8 bits per encrypted image.

C.2 Comparison to Transciphering LWR

Assume here that $q/(2p) \in \mathbb{Z}$. One can interpret an LWR instance $\left(\boldsymbol{a}, b = \lceil \frac{p}{q} \cdot \boldsymbol{a}^{\top}\boldsymbol{s} \rfloor\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ as an LWE instance

$$\left(\boldsymbol{a}, \tfrac{q}{p} \cdot b\right) = \left(\boldsymbol{a}, \boldsymbol{a}^{\top}\boldsymbol{s} + e \bmod q\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \ .$$

In particular, the error e in the latter satisfies $-q/(2p) < e \leq q/(2p)$. Therefore, yet another way to reduce bandwidth/storage on a server is for the client to send symmetric LWR encryptions [38] of $\boldsymbol{M} \in \mathbb{Z}_p^m$ taking the form

$$c_i = M_i + \left\lceil \tfrac{p}{q} \cdot H(x,i)^{\top}\boldsymbol{s} \right\rfloor \in \mathbb{Z}_p \text{ for } i \in \{1, \ldots, m\} \ .$$

These can then be interpreted as (potentially maximal noise) symmetric TFHE encryptions $d_i = \frac{q}{p}c_i = H(x,i)^{\top}\boldsymbol{s} + e_i + \frac{q}{p}M_i$. When M_i is required for some computation, the server may simply bootstrap d_i in order to get a small-noise TFHE ciphertext.

Setting $q = 2N$, we very nearly recover our proposed solution using the modified PRF. However, the performance is subtly different. A first difference is that bootstrapping the ciphertext d_i requires two sequential bootstraps in the case that there is no padding bit in M_i. This issue does not arise in our modified PRF solution. A second difference is that the symmetric-key cipher here depends on an LWR assumption with moduli $(p, 2N)$ whereas the modified PRF method uses an LWR assumption with moduli $(2p, 2N)$ which may require to choose a larger LWR dimension. However, this increase in dimension is expected to be less than a factor of 2 meaning that our modified PRF method will likely be more computationally efficient overall. As a realistic example, we may take parameters $N = 1024$, $p = 32$ (Section 6). The required LWR dimension for moduli $(2p, 2N)$ is 455 for around 128 bits of security. This dimension decreases modestly to around 435 when using moduli $(p, 2N)$.