Key-Homomorphic Pseudorandom Functions from LWE with a Small Modulus

Sam Kim
Stanford University
skim13@cs.stanford.edu

Abstract

Pseudorandom functions (PRFs) are fundamental objects in cryptography that play a central role in symmetric-key cryptography. Although PRFs can be constructed from one-way functions generically, these black-box constructions are usually inefficient and require deep circuits to evaluate compared to direct PRF constructions that rely on specific algebraic assumptions. From lattices, one can directly construct PRFs from the Learning with Errors (LWE) assumption (or its ring variant) using the result of Banerjee, Peikert, and Rosen (Eurocrypt 2012) and its subsequent works. However, all existing PRFs in this line of work rely on the hardness of the LWE problem where the associated modulus is super-polynomial in the security parameter.

In this work, we provide two new PRF constructions from the LWE problem. In each of these constructions, each focuses on either minimizing the depth of its evaluation circuit or providing key-homomorphism while relying on the hardness of the LWE problem with either a polynomial modulus or nearly polynomial modulus. Along the way, we introduce a new variant of the LWE problem called the Learning with Rounding and Errors (LWRE) problem. We show that for certain settings of parameters, the LWRE problem is as hard as the LWE problem. We then show that the hardness of the LWRE problem naturally induces a pseudorandom synthesizer that can be used to construct a low-depth PRF. The techniques that we introduce to study the LWRE problem can then be used to derive variants of existing key-homomorphic PRFs whose security can be reduced from the hardness of the LWE problem with a much smaller modulus.

1 Introduction

A pseudorandom function (PRF) [GGM86] is a deterministic function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that satisfies a specific security property: for a randomly sampled key $k \xleftarrow{R} \mathcal{K}$, the output of the function $F(k, \cdot)$ is computationally indistinguishable from those of a truly random function. PRFs are fundamental objects in cryptography that serve as a basis for symmetric cryptography. Even beyond symmetric cryptography, PRFs serve as one of the core building blocks for many advanced cryptographic constructions and protocols.

In theory, PRFs can be constructed from any one-way function via the transformation of [HILL99, GGM86]. However, the PRFs that are constructed from one-way functions in a blackbox way are generally inefficient. Furthermore, the transformation of [HILL99, GGM86] is inherently *sequential* and therefore, the resulting PRFs require deep circuits to evaluate. For practical deployment, this is problematic as symmetric objects like PRFs are often deployed inside designated hardware devices similar to how modern blockciphers, such as AES, are incorporated into many modern processors.

For these settings, it is important for PRFs to exhibit *low depth* evaluation circuits that require few computing cycles to evaluate using multiple cores or processing units.

For these reasons, constructing low-depth pseudorandom functions from standard cryptographic assumptions have been a highly active area of research. Starting from the seminal work of Naor and Reingold [NR99], there have been great progress in constructing low-depth pseudorandom functions from *group-based* assumptions like the Decisional Diffie-Hellman (DDH) assumption [NR99, ABP15a, ABP15b]. However, constructing low-depth PRFs from standard lattice assumptions such as the Learning with Errors (LWE) assumption [Reg09] has been surprisingly more difficult. Indeed, a low-depth PRF from LWE was constructed by a breakthrough result of Banerjee, Peikert, and Rosen [BPR12], but only after the realization of seemingly more powerful primitives such as (lattice-based) identity-based encryption [GPV08, ABB10a, ABB10b, CHKP10] and fully homomorphic encryption [Gen09, BV11b, BV11a].

Key-homomorphic PRFs. Since the work of [BPR12], the study of lattice-based PRFs has become a highly productive area of research. There have been a sequence of results that further improve the constructions of [BPR12] with various trade-offs in the parameters [BLMR13, BP14, DS15, Mon18, JKP18]. A long sequence of results also show how to construct PRFs with useful algebraic properties such as key-homomorphic PRFs [BLMR13, BP14, BV15], constrained PRFs [BV15, BKM17, CC17, BTVW17], and even watermarkable PRFs [KW17, CC17, QWZ18, KW19] from LWE.

A special family of PRFs that are particularly useful for practical applications are key-homomorphic PRFs. The concept was first introduced by Naor, Pinkas, and Reingold [NPR99] and it was first formalized as a cryptographic primitive by Boneh et al. [BLMR13]. We say that a pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is *key-homomorphic* if the key-space $(\mathcal{K}, \oplus)$ and the range of the PRF $(\mathcal{Y}, \otimes)$ exhibit group structures such that for any two keys k_1, k_2 and input $x \in \mathcal{X}$, we have $F(k_1 \oplus k_2, x) = F(k_1, x) \otimes F(k_2, x)$. Key-homomorphic PRFs have many useful applications in symmetric cryptography and give rise to distributed PRFs, symmetric-key proxy re-encryption, and updatable encryption. The study of updatable encryption, in particular, have recently gained a considerable amount of traction [BLMR13, EPRS17, LT18, KLR19, BDGJ19]. Most of these existing proposals for updatable encryption rely on key-homomorphic PRFs or use direct updatable encryption constructions that take advantage of similar algebraic structures.

LWE modulus. Despite significant progress in our understanding of lattice-based PRFs as described above, all existing direct PRF constructions suffer from one caveat: the modulus q, which defines the underlying ring for the PRF, must be set to be super-polynomial in the security parameter. The need for a large modulus q has several disadvantages. The first and immediate disadvantage is *efficiency*. A lattice-based PRF is generally defined with respect to a set of public matrices in $\mathbb{Z}_q^{n \times m}$ and a secret vector in $\mathbb{Z}_q^n$ for some suitable choice of parameters n and m. A bigger modulus q means that more space is required to store these values and more time is needed to evaluate the PRF.

Another disadvantage of a large modulus q is related to the *quality* of the LWE assumption that is needed to prove security. Generally, PRFs that are defined with a super-polynomial modulus q relies on the hardness of the LWE problem with a super-polynomial noise-to-modulus ratio. This means that the security of the PRF can only be based on the hardness of solving worst-case lattice problems with a super-polynomial approximation factor, which is a significantly stronger assumption than what is required by many other lattice-based cryptographic constructions. In fact, today, we can base seemingly stronger primitives like fully-homomorphic encryption [BV14, AP14] and attribute-based encryption [GV15] (when restricted to NC^1 computations) on the hardness of approximating worst-case lattice problems with only polynomial approximation factors. This clearly demonstrates that our current understanding of lattice-based PRFs is still quite limited.

Many existing lattice-based PRFs including the original constructions of [BPR12] are, in fact, conjectured to be secure even when they are instantiated with much smaller values of the modulus q. However, their formal proof of security has been elusive for many years. An important question in the study of lattice-based PRFs is whether there exist direct lattice-based PRF constructions that rely on a polynomial modulus q that still exhibit some of the useful features not satisfied by the generic constructions [HILL99, GGM86] such as low-depth evaluation circuits or key-homomorphism.

1.1 Our Contributions

In this work, we present two PRF constructions from the hardness of the LWE problem with a small modulus q. For our first construction, we focus on minimizing the depth of the evaluation circuit while in our second construction, we focus on constructing a key-homomorphic PRF. In both settings, our goal is to construct lattice-based PRFs that work over small moduli q.

1.1.1 Low-depth PRF

In our first PRF construction, our main focus is on minimizing the size of the modulus q while also minimizing the depth of the PRF evaluation circuit. We provide an overview of the construction in Section 2 and briefly discuss our approach below. The resulting PRF can be instantiated with a range of parameter settings that are determined by a trade-off between the depth of the evaluation circuit and the associated modulus q. We consider two types of parameter settings that provide different levels of security.

- *Theoretical security*: In this setting, we guarantee that any adversary has at most a negligible advantage in breaking the PRF. For this level of security, we can set the parameters of our PRF such that the modulus is polynomial in the security parameter $q = \tilde{O}(\lambda^{2.5})$ and the depth of the evaluation circuit is in NC^2.

- 2^λ*-security*: In this setting, we guarantee that an adversary's advantage in breaking the PRF degrades exponentially in the security parameter. For this level of security, we can set the parameters of our PRF such that the depth of the evaluation circuit is $\tilde{O}(\lambda/\log q)$. In particular, setting $q = 2^{\tilde{O}(\sqrt{\lambda})}$, the PRF evaluation can be done in depth $\tilde{O}(\sqrt{\lambda})$. Previously, all lattice-based PRFs either required that the depth of the evaluation circuit is at least $\tilde{O}(\lambda)$ or the modulus q to be at least $2^{\tilde{O}(\lambda)}$.

We provide a comparison of the size of the modulus q and the depth of the evaluation circuit that is needed for our PRF with those of existing LWE-based PRFs in Table 1. We further discuss how to interpret our parameter settings in Section 1.2 and how to concretely instantiate them in Section 4.3.

Synthesizers and LWRE. The main intermediate object that we use to construct our PRF is a pseudorandom synthesizer (Definition 4.5), which was first introduced by Naor and Reingold [NR99]. They showed that a pseudorandom synthesizer that can be computed by a low-depth circuit can be used to construct a PRF that can also be computed by a low-depth circuit. The work of Banerjee, Peikert, and Rosen [BPR12] first showed that such pseudorandom synthesizers can be constructed

Construction	Size of Modulus	Evaluation Depth
[BPR12, GGM]	$\lambda^{\Omega(1)}$	$\Omega(\lambda \log \lambda)$
[BPR12, Synthesizer]	$2^{\Omega(\lambda)}$	$\Omega(\log^2 \lambda)$
[BPR12, Direct]	$2^{\Omega(\lambda \log \lambda)}$	$\Omega(\log^2 \lambda)$
[BLMR13]	$2^{\Omega(\lambda \log \lambda)}$	$\Omega(\log^2 \lambda)$
[BP14]	$2^{\Omega(\lambda)}$	$\Omega(\log^2 \lambda)$
[DS15]	$2^{\Omega(\lambda)}$	$\Omega(\log^{1+o(1)} \lambda)$
[Mon18]	$2^{\Omega(\lambda)}$	$\Omega(\log^2 \lambda)$
[JKP18]	$2^{\Omega(\lambda)}$	$\Omega(\log^{1+o(1)} \lambda)$
This work: Synthesizer-based	$2^{\Omega(\sqrt{\lambda})}$	$\Omega(\sqrt{\lambda} \log \lambda)$
This work: BP-based	$\lambda^{\Omega(1)}$	$\Omega(\lambda^2 \log \lambda)$

Table 1: Comparison of the PRF constructions in this work and the existing PRF constructions based on LWE. For each of the PRF constructions, the table denotes the size of the modulus q that is needed to prove 2^λ-security from LWE and the depth of the evaluation circuit that is needed to evaluate the PRFs.

from a natural variant of the LWE problem called the Learning with Rounding (LWR) problem. They showed that the hardness of the LWR problem can be reduced from the hardness of the LWE problem when the modulus q is set to be very large.

To construct a pseudorandom synthesizer from a small-modulus LWE problem, we introduce yet another variant of the LWE problem called the Learning with Rounding *and* Errors (LWRE) problem whose hardness naturally induces a pseudorandom synthesizer. The challenger for an LWRE problem chains multiple samples of the LWR and LWE problems together such that the error terms that are involved in each of the LWE samples are derived from the "noiseless" LWR samples. This specific way of chaining LWR and LWE samples together allows us to reduce the hardness of LWRE from the hardness of the LWE problem with a much smaller modulus q. We provide an overview of the LWRE problem and the reduction from LWE in Section 2. We precisely formulate the LWRE problem in Definition 4.1 and provide the formal reduction in Section A.

The LWRE problem and our synthesizer construction naturally extend to the ring setting as well. In Definition 5, we formulate the Ring-LWRE problem similarly to how the Ring-LWE and Ring-LWR problems are defined. Then, in Construction 5.4, we show how to construct a pseudorandom synthesizer from the Ring-LWRE problem.

1.1.2 Key-homomorphic PRF

For our second construction, we focus on constructing a key-homomorphic PRF with a small modulus q. Specifically, we provide a key-homomorphic PRF whose security (either theoretical or 2^λ-security) can be based on the hardness of LWE with a polynomial size q without relying on random oracles. All previous key-homomorphic PRFs from lattices either relied on LWE with a super-polynomial modulus q [BLMR13, BP14, BV15] or random oracles [NPR99, BLMR13]. As in previous LWE-based key-homomorphic PRFs, our construction is "almost" key-homomorphic in that the homomorphism on the PRF keys hold subject to some small rounding error, which does not significantly impact its usefulness to applications.

Our construction relies on the same chaining technique that is used to construct our first PRF. This time, instead of chaining multiple LWE and LWR samples together as was done in our first construction, we chain multiple instances of the existing lattice-based PRFs themselves. For most existing PRF constructions that are based on LWE, their proof of security proceeds in two steps:

1. Define a "noisy" variant of the deterministic PRF function whose security can be based on the hardness of the LWE problem.

2. Show that the deterministic PRF function and its "noisy" variant have the same input-output behavior with overwhelming probability over the randomness used to sample the noise.

Generally, in order to show that the deterministic PRF and its "noisy" variant are statistically indistinguishable in step 2 above, the modulus q has to be set to be super-polynomial in the security parameter.

To reduce the need for a large modulus q in step 2, we chain multiple instances of the deterministic PRF and its noisy variant. Namely, our PRF construction consists of many noisy variants of these PRFs that are chained together such that the noise that is needed to evaluate the noisy PRF in a chain is derived from a PRF in the previous level of the chain. By setting the initial PRF at the start of the chain to be the original deterministic PRF, the entire evaluation of the chained PRF can be made to be deterministic.

This simple way of chaining multiple instances of deterministic and noisy variants of PRFs allows us to prove the security of the final PRF from the hardness of LWE with a much smaller modulus q. In fact, when we chain multiple instances of a key-homomorphic PRF, the resulting PRF is also key-homomorphic. Instantiating the chain with the Banerjee-Peikert key-homomorphic PRF [BP14] results in a key-homomorphic PRF that works over a polynomial modulus q. We provide a detailed overview of our technique in Section 2 and provide the formal construction and its security proof in Section 6.

1.2 Discussions

Regarding theoretical and 2^λ-security. The reason why we present our results with respect to both theoretical and 2^λ-security is due to the fact that the generic PRF constructions [GGM86] can already be used to construct a low-depth PRF that provides asymptotically equivalent level of security. Note that using a length-doubling pseudorandom generator, the Goldwasser, Goldreich, and Micali [GGM86] construction can be used to provide a PRF that can be evaluated in depth linear in the input size of the PRF. One way to achieve a poly-logarithmic depth PRF using the GGM construction is to first hash the input using a universal hash function into a domain of size $2^{\omega(\log \lambda)}$ and then apply the PRF on the hash of the message. As the hashed outputs are poly-logarithmic in length, the PRF can be evaluated in poly-logarithmic depth. At the same time, as long as the adversary is bounded to making a polynomial number of evaluation queries on the PRF, a collision on the hash function occurs with negligible probability and therefore, any efficient adversary can have at most negligible advantage in breaking the PRF. As a length-doubling PRG can be constructed from LWE with a polynomial modulus q, this gives an LWE-based PRF with both small evaluation depth and small modulus q. Of course, the actual security of this final PRF is quite poor since the probability that an adversary forces a collision on the hash function is barely negligible.

Therefore, the way to view our low-depth PRF is to consider its parameters when they are set to provide 2^λ-security. In this setting, our PRF provides security under the condition that $d \log q = \tilde{\Omega}(\lambda)$ where d is the depth of the evaluation circuit. When setting $\tilde{\Omega}(\log q) = \sqrt{\lambda}$, the evaluation circuit has depth that scales with $\sqrt{\lambda}$. This means that setting $\lambda = 128$ and ignoring arithmetic and vector operations, our PRF can be evaluated by a circuit with depth ≈ 11 that works over a ≈ 11-bit modulus. The GGM PRF, on the other hand, requires a circuit with depth at least $\lambda = 128$, which is prohibitive for practical use, while the existing lattice-based PRFs require $7 \approx \log \lambda$ circuit depth, but must operate over at least a 128-bit modulus. We discuss concrete instantiation of our scheme in Section 4.3.

We note that for key-homomorphic PRFs, no construction that works over a polynomial modulus was previously known. Therefore, our second PRF construction can be viewed as the first key-homomorphic PRF that works over a polynomial modulus independent of whether it provides theoretical or 2^λ-security.

On the chaining method and blockciphers. The pseudorandom synthesizers or PRFs in this work consist of many repeated rounds of computation that are chained together in such a way that the output of each round of computation is affected by the output of the previous round. This way of chaining multiple rounds of computation is reminiscent of the structure of many existing blockciphers such as DES or AES, which also consist of many rounds of bit transformations that are chained together. Interestingly, chaining in blockciphers and chaining in our work seem to serve completely opposite roles. In blockcipher design, chaining is generally used to achieve the effect of *diffusion*, which guarantees that a small change to the input to the blockcipher significantly alters its final output. This assures that no correlation can be efficiently detected between the input and the output of the blockcipher. In this work, chaining is used to actually *prevent* diffusion. Namely, chaining guarantees that some small error that is induced by the PRF evaluation does not affect the final output of the PRF. This allows us to switch from the real PRF evaluation to the "noisy" PRF evaluation in our hybrid security argument such that we can embed an LWE challenge to the output of the PRF.

1.3 Other Related Work

PRF cascades. The techniques that are used in this work are conceptually similar to PRF cascading, which is the process of chaining multiple small-domain PRFs to construct large-domain PRFs. The technique was first introduced by Bellare et al. [BCK96b] and was further studied by Boneh et al. [BMR10]. PRF cascades serve as a basis for NMAC and HMAC PRFs [BCK96a, Bel06].

LWR for bounded number of samples. There have been a sequence of results that study the hardness of the Learning with Rounding problem when the number of samples are a priori bounded. Alwen et al. [AKPW13] first showed that such variant of LWR is as hard as LWE even when the modulus q is set to be polynomial in the security parameter. Bogdanov et al. [BGM$^+$16] improved the statistical analysis of this reduction using Rényi divergence. Alperin-Sherif and Apon [AA16] further improved these previous results such that the reduction from LWE to LWR is sample-preserving.

2 Overview

In this section, we provide a high level overview of the main techniques that we use in this work. For the full details of our main results and proofs, we refer the readers to Sections 4 and 6.

We divide the overview into three parts. In Section 2.1, we provide additional background on existing results on constructing lattice-based PRFs. In Section 2.2, we provide an overview of our synthesizer construction from a new computational problem called the Learning with Rounding and Errors (LWRE) problem. Then, in Section 2.3, we show how the technique that we use to prove the security of our synthesizer-based PRF can be applied to the parameters for existing lattice-based key-homomorphic PRFs.

2.1 Background on Lattice PRFs via Synthesizers

The main intermediate primitive that we use to construct our first lattice-based PRF is a special family of pseudorandom generators called *pseudorandom synthesizers* [NR99]. A pseudorandom synthesizer over a domain $\mathcal{D}$ is a two-to-one function $S : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ such that for any (a priori *unbounded*) polynomial number of inputs $a_1, \ldots, a_\ell \xleftarrow{R} \mathcal{D}$, and $b_1, \ldots, b_\ell \xleftarrow{R} \mathcal{D}$, the set of ℓ^2 elements $\{S(a_i, b_j)\}_{i,j\in[\ell]}$ are computationally indistinguishable from uniformly random elements $\{u_{i,j}\}_{i,j\in[\ell]} \xleftarrow{R} \mathcal{D}^{\ell^2}$.

A pseudorandom synthesizer $S : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ induces a PRF F with key space $\mathcal{D}^{2\ell}$, domain $\{0,1\}^\ell$, and range $\mathcal{D}$ for any positive power-of-two integer ℓ as follows:

- Define a PRF key to consist of 2ℓ uniformly random elements in $\mathcal{D}$:

$$\begin{pmatrix} s_{1,0} & s_{2,0} & \cdots & s_{\ell,0} \\ s_{1,1} & s_{2,1} & & s_{\ell,1} \end{pmatrix}.$$

- To evaluate the PRF on an input $x \in \{0,1\}^\ell$, compress the subset of the elements $s_{1,x_1}, \ldots, s_{\ell,x_\ell}$ into a single element of $\mathcal{D}$ by iteratively applying the synthesizer:

$$S\Big(\cdots S\big(S(s_{1,x_1}, s_{2,x_2}), S(s_{3,x_3}, s_{4,x_4})\big), \cdots S(s_{\ell-1,x_{\ell-1}}, s_{\ell,x_\ell}) \cdots\Big).$$

The pseudorandomness of the output of the PRF can roughly be argued as follows. Since each of the ℓ elements $s_{1,x_1}, \ldots, s_{\ell,x_\ell} \in \mathcal{D}$ that are part of the PRF key are sampled uniformly at random, the compressed $\ell/2$ elements $S(s_{1,x_1}, s_{2,x_2}), \ldots, S(x_{\ell-1,x_{\ell-1}}, x_{\ell,x_\ell})$ are computationally indistinguishable from random elements in $\mathcal{D}$. This, in turn, implies that the compression of these $\ell/2$ elements into $\ell/4$ elements are pseudorandom. This argument can be applied iteratively to show that the final output of the PRF is computationally indistinguishable from uniform in $\mathcal{D}$.

LWE and Synthesizers. As pseudorandom synthesizers imply pseudorandom functions, one can naturally hope to construct a pseudorandom synthesizer from the LWE problem [Reg09]. Recall that the $\mathsf{LWE}_{n,q,\chi}$ assumption, parameterized by positive integers n, q and a B-bounded error distribution χ, states that for any (a priori unbounded) $m = \mathsf{poly}(\lambda)$, if we sample a uniformly random secret vector $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, uniformly random public vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \xleftarrow{R} \mathbb{Z}_q^n$, "small" error terms $e_1, \ldots, e_\ell \leftarrow \chi^m$, and uniformly random values $u_1, \ldots, u_m \xleftarrow{R} \mathbb{Z}_q$, the following distributions are computationally indistinguishable:

$$\begin{array}{ccc}(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s}\rangle + e_1) & \approx_c & (\mathbf{a}_1, u_1) \\ (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s}\rangle + e_2) & \approx_c & (\mathbf{a}_2, u_2) \\ \vdots & & \vdots \\ (\mathbf{a}_m, \langle \mathbf{a}_m, \mathbf{s}\rangle + e_m) & \approx_c & (\mathbf{a}_m, u_m).\end{array}$$

Given the $\mathsf{LWE}_{n,q,\chi}$ assumption, it is natural to define a (randomized) pseudorandom synthesizer $S : \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_q^{n\times n}$ as follows

$$S(\mathbf{S}, \mathbf{A}) = \mathbf{S} \cdot \mathbf{A} + \mathbf{E},$$

where the error matrix $\mathbf{E} \leftarrow \chi^{n\times n}$ is sampled randomly by the synthesizer S. It is easy to show via a standard hybrid argument that for any set of matrices $\mathbf{S}_1, \ldots, \mathbf{S}_\ell \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times n}$, $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times n}$, and $\mathbf{E}_{1,1}, \ldots, \mathbf{E}_{\ell,\ell} \leftarrow \chi^{n\times n}$, the pairwise applications of the synthesizer $S(\mathbf{S}_i, \mathbf{A}_j) = \mathbf{S}_i \cdot \mathbf{A}_j + \mathbf{E}_{i,j}$ for all $i, j \in [n]$ result in pseudorandom matrices.

Learning with Rounding. The problem with the synthesizer construction above is that the synthesizer must be randomized. Namely, in order to argue that the synthesizer's outputs are pseudorandom, the evaluation algorithm must flip random coins and sample independent error matrices $\mathbf{E}_{i,j} \leftarrow \chi^{n\times n}$ for each execution $S(\mathbf{S}_i, \mathbf{A}_j)$ for $i, j \in [\ell]$. Otherwise, if the error matrices are derived from an additional input to the synthesizer, then the error matrices for each evaluation of the synthesizer $S(\mathbf{S}_i, \mathbf{A}_j)$ for $i, j \in [\ell]$ must inevitably be correlated and hence, the security of the synthesizer cannot be shown from the hardness of $\mathsf{LWE}_{n,q,\chi}$.

Banerjee, Peikert, and Rosen [BPR12] provided a way to overcome this obstacle by introducing a way of derandomizing errors in LWE samples. The idea is quite simple and elegant: instead of adding small random error terms $e \leftarrow \chi$ to each inner product $\langle \mathbf{a}, \mathbf{s}\rangle \in \mathbb{Z}_q$, one can deterministically round it to one of $p < q$ partitions or "buckets" in $\mathbb{Z}_q$. Concretely, the idea can be implemented by applying the *modular rounding* operation $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ to the inner product $\langle \mathbf{a}, \mathbf{s}\rangle$, which maps $\langle \mathbf{a}, \mathbf{s}\rangle \mapsto \lfloor \langle \mathbf{a}, \mathbf{s}\rangle \cdot p/q \rceil$. Intuitively, adding a small noise term $e \leftarrow \chi$ to the inner product of $\langle \mathbf{a}, \mathbf{s}\rangle$ in the $\mathsf{LWE}_{n,q,\chi}$ problem *blinds* its low-ordered bits from a distinguisher. Therefore, applying the modular rounding operation to $\langle \mathbf{a}, \mathbf{s}\rangle$, which *removes* the low-ordered bits (albeit deterministically), should make the task of distinguishing it from random just as hard.

With this intuition, [BPR12] introduced a new computational problem called the Learning with Rounding (LWR) problem. For parameters $n, q, p \in \mathbb{N}$ where $p < q$, the $\mathsf{LWR}_{n,q,p}$ problem asks an adversary to distinguish the distributions:

$$\begin{array}{ccc}(\mathbf{a}_1, \lfloor \langle \mathbf{a}_1, \mathbf{s}\rangle \rceil_p) & \approx_c & (\mathbf{a}_1, u_1) \\ (\mathbf{a}_2, \lfloor \langle \mathbf{a}_2, \mathbf{s}\rangle \rceil_p) & \approx_c & (\mathbf{a}_2, u_2) \\ \vdots & & \vdots \\ (\mathbf{a}_m, \lfloor \langle \mathbf{a}_m, \mathbf{s}\rangle \rceil_p) & \approx_c & (\mathbf{a}_m, u_m),\end{array}$$

where $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{a}_1, \ldots, \mathbf{a}_m \xleftarrow{\text{R}} \mathbb{Z}_q^n$, and $u_1, \ldots, u_m \xleftarrow{\text{R}} \mathbb{Z}_p$. The hardness of the LWR problem then induces a deterministic pseudorandom synthesizer $S : \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$ that is defined as follows:

$$S(\mathbf{S}, \mathbf{A}) = \lfloor \mathbf{S} \cdot \mathbf{A} \rceil_p,$$

where the modular rounding is done component-wise to each entry of the matrix $\mathbf{S} \cdot \mathbf{A} \in \mathbb{Z}_q^{n\times n}$.[1]

Reducing LWE to LWR. Now, the remaining question is whether the LWR problem can formally be shown to be as hard as the LWE problem. The work of [BPR12] gave a positive answer to this question. They showed that for any B-bounded distribution χ and moduli q and p for which $q = 2Bp\lambda^{\omega(1)}$, the $\mathsf{LWR}_{n,q,p}$ problem is as hard as the $\mathsf{LWE}_{n,q,\chi}$ problem.[2] Given an adversary for the $\mathsf{LWR}_{n,q,p}$ problem $\mathcal{A}$, one can construct a simple algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve $\mathsf{LWE}_{n,q,\chi}$. Specifically, on input an $\mathsf{LWE}_{n,q,\chi}$ challenge $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, algorithm $\mathcal{B}$ simply provides $(\mathbf{a}_1, \lfloor b_1 \rceil_p), \ldots, (\mathbf{a}_m, \lfloor b_m \rceil_p)$ to $\mathcal{A}$.

- If the values $b_1, \ldots, b_m \in \mathbb{Z}_q$ are noisy inner products $b_1, = \langle \mathbf{a}_1, \mathbf{s}\rangle + e_1, \ldots, b_m = \langle \mathbf{a}_m, \mathbf{s}\rangle + e_m$, then we have
$$\lfloor b_i \rceil_p = \lfloor \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \rceil_p = \lfloor \langle \mathbf{a}_i, \mathbf{s}\rangle \rceil_p$$
for all $i \in [m]$ except with negligible probability over $\mathbf{a}_1, \ldots, \mathbf{a}_m \xleftarrow{\text{R}} \mathbb{Z}_q^n$.

- If the values $b_1, \ldots, b_m$ are uniformly random in $\mathbb{Z}_q$, then the values $\lfloor b_1 \rceil_p, \ldots, \lfloor b_m \rceil_p$ are also uniform in $\mathbb{Z}_p$.

Hence, the algorithm $\mathcal{B}$ statistically simulates the correct distribution of an $\mathsf{LWR}_{n,q,p}$ challenge for $\mathcal{A}$ and therefore, can solve the $\mathsf{LWE}_{n,q,\chi}$ problem with essentially the same advantage as $\mathcal{A}$.

The apparent limitation of this reduction is the need for the modulus q to be super-polynomial in the security parameter. If q is only polynomial, then with noticeable probability, the inner product $\langle \mathbf{a}_i, \mathbf{s}\rangle \in \mathbb{Z}_q$ for any $i \in [m]$ lands on a rounding "borderline" set

$$\begin{aligned} \mathsf{Borderline}_B &= [-B, B] + q/p \cdot (\mathbb{Z} + 1/2) \\ &= \{\ v \in \mathbb{Z}_q \mid \exists\ e \in [-B, B] \text{ such that } \lfloor v \rceil_p \neq \lfloor v + e \rceil_p\ \}, \end{aligned}$$

and hence $\lfloor \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \rceil_p \neq \lfloor \langle \mathbf{a}_i, \mathbf{s}\rangle \rceil_p$. In this case, one cannot guarantee that an adversary $\mathcal{A}$ for the $\mathsf{LWR}_{n,q,p}$ problem correctly distinguishes the samples $(\mathbf{a}_1, \lfloor \langle \mathbf{a}_1, \mathbf{s}\rangle + e_1 \rceil_p), \ldots, (\mathbf{a}_m, \lfloor \langle \mathbf{a}_m, \mathbf{s}\rangle + e_m \rceil_p)$ from purely random samples.

2.2 Learning with Rounding and Errors.

Chaining LWE Samples. We get around the limitation of the reduction above by using what we call the *chaining method*. To demonstrate the idea, let us consider a challenge oracle $\mathcal{O}_{\tau,\mathbf{S}}$ that chains multiple $\mathsf{LWE}_{n,q,\chi}$ samples together. The oracle $\mathcal{O}_{\tau,\mathbf{S}}$ is parameterized by a chaining parameter $\tau \in \mathbb{N}$, a set of secret vectors $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n\times\tau}$, and is defined with respect to a sampling algorithm $D_\chi : \{0,1\}^{\lfloor \log p \rfloor} \to \mathbb{Z}$, which takes in as input $\lfloor \log p \rfloor$ random coins and samples an error value $e \in \mathbb{Z}$ according to the B-bounded error distribution χ.

[1] Note that the synthesizer maps matrices in $\mathbb{Z}_q^{n\times n}$ to matrices in $\mathbb{Z}_p^{n\times n}$ for $p < q$ and hence violates the original syntax of a synthesizer. This is a minor technicality, which can be addressed in multiple ways. One option is to have a sequence of rounding moduli $p_1, \ldots, p_{\log \ell}$ such that the synthesizer can be applied iteratively. Another option is to take a pseudorandom generator $G : \mathbb{Z}_p^{n\times n} \to \mathbb{Z}_q^{n\times n}$ and define $S(\mathbf{S}, \mathbf{A}) = G(\lfloor \mathbf{S} \cdot \mathbf{A} \rceil_p)$. Finally, one can define the synthesizer $S(\mathbf{S}, \mathbf{A}) = \lfloor \mathbf{S} \cdot \mathbf{A} \rceil_p$ with respect to non-square matrices $S : \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_p^{m\times m}$ such that the sets $\mathbb{Z}_q^{m\times n}$, $\mathbb{Z}_q^{n\times m}$, and $\mathbb{Z}_p^{m\times m}$ have the same cardinality $|\mathbb{Z}_q^{m\times n}| = |\mathbb{Z}_q^{n\times m}| = |\mathbb{Z}_p^{m\times m}|$.

[2] For the reduction to succeed with probability $1 - 2^{-\lambda}$, we must set $q \geq 2Bp \cdot 2^\lambda$. For simplicity throughout the overview, we restrict to the "negligible vs. non-negligible" type security as opposed to 2^λ-security. See Section 1.2 for further discussions.

- $\mathcal{O}_{\tau,\mathbf{S}}$: On its invocation, the oracle samples a public vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and an error term $e_1 \leftarrow \chi$. Then, for $1 \leq i < \tau$, it iteratively computes:
  - $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
  - $e_{i+1} \leftarrow D_\chi(r_i)$.

  It then returns $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p)$.

In words, the oracle $\mathcal{O}_{\tau,\mathbf{S}}$ generates (the rounding of) an $\mathsf{LWE}_{n,q,\chi}$ sample $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle + e_1 \rceil_p)$ and uses $r_1 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle + e_1 \rceil_p$ as the random coins to sample $e_2 \leftarrow D_\chi(r_1)$. It then computes $r_2 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_2 \rangle + e_2 \rceil_p$ and uses r_2 to sample the next error term $e_3 \leftarrow D_\chi(r_2)$ for the next iteration. The oracle iterates this procedure for τ steps and finally returns $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p)$.

Now, suppose that p divides q. Then a hybrid argument shows that assuming the hardness of $\mathsf{LWE}_{n,q,\chi}$, a sample $(\mathbf{a}, b) \leftarrow \mathcal{O}_{\tau,\mathbf{S}}$ is computationally indistinguishable from uniform in $\mathbb{Z}_q^n \times \mathbb{Z}_p$. Specifically, we can argue that the first term (random coins) $r_1 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle + e_1 \rceil_p$ is computationally indistinguishable from uniform in $\{0,1\}^{\lfloor \log p \rfloor}$ by the hardness of $\mathsf{LWE}_{n,q,\chi}$. Then, since r_1 is uniform, the error term $e_2 \leftarrow D_\chi(r_1)$ is correctly distributed according to χ, which implies that $r_2 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_2 \rangle + e_2 \rceil_p$ is also computationally indistinguishable from uniform. Continuing this argument for τ iterations, we can prove that the final output $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p)$ is computationally indistinguishable from uniform in $\mathbb{Z}_q^n \times \mathbb{Z}_p$.

Chaining LWR and LWE Samples. So far, it seems as if we have not made much progress. Although the oracle $\mathcal{O}_{\tau,\mathbf{s}}$ returns an "LWR looking" sample $(\mathbf{a}, b) \in \mathbb{Z}_q \times \mathbb{Z}_p$, it must still randomly sample the initial noise term $e_1 \leftarrow \chi$, which makes it useless for constructing a deterministic pseudorandom synthesizer. Our key observation, however, is that when the chaining parameter τ is big enough, then the initial error term e_1 does not affect the final output $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p)$ with overwhelming probability. In other words, e_1 can always be set to be 0 without negatively impacting the pseudorandomness of $\mathcal{O}_{\tau,\mathbf{S}}$.

To see this, consider the following modification of the oracle $\mathcal{O}_{\tau,\mathbf{S}}$:

- $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$: On its invocation, the oracle samples a public vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and initializes $e_1 = 0$. Then, for $1 \leq i < \tau$, it iteratively computes:
  - $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
  - $e_{i+1} \leftarrow D_\chi(r_i)$.

  It returns $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p)$.

In contrast to $\mathcal{O}_{\tau,\mathbf{S}}$, the oracle $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$ derives the first set of random coins r_1 from an errorless $\mathsf{LWR}_{n,q,p}$ sample $r_1 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle \rceil_p$. It then uses r_1 to sample the error term $e_2 \leftarrow D_\chi(r_1)$ for the next $\mathsf{LWE}_{n,q,\chi}$ sample to derive $r_2 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_2 \rangle + e_2 \rceil_p$, and it continues this procedure for τ iterations. As the oracle $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$ is, in effect, chaining $\mathsf{LWR}_{n,q,p}$ and $\mathsf{LWE}_{n,q,\chi}$ samples together, we refer to $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$ as the *Learning with (both) Rounding and Errors* (LWRE) oracle.

We claim that even when q is small, as long as the chaining parameter τ is big enough, the samples that are output by the oracles $\mathcal{O}_{\tau,\mathbf{S}}$ and $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$ are identical except with negligible probability. For simplicity, let us fix the modulus to be $q = 4Bp$ such that for $u \xleftarrow{\text{R}} \mathbb{Z}_q$, $e_1, e_2 \leftarrow \chi$, we have

$$\Pr[\lfloor u + e_1 \rceil_p \neq \lfloor u + e_2 \rceil_p] \leq \frac{1}{2}. \tag{2.1}$$

Now, consider a transcript of an execution of the oracles $\mathcal{O}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ for $\mathbf{S} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times\tau}$, $\mathbf{a} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, and any fixed error value $e_1 \in [-B, B]$:

$$
\begin{array}{ll}
\mathcal{O}_{\tau,\mathbf{S}}: & \mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}: \\
\quad r_1 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle + e_1 \rceil_p & \quad \tilde{r}_1 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_1 \rangle + 0 \rceil_p \\
\quad r_2 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_2 \rangle + e_2 \rceil_p & \quad \tilde{r}_2 \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_2 \rangle + \tilde{e}_2 \rceil_p \\
\qquad \vdots & \qquad \vdots \\
\quad r_\tau \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p & \quad \tilde{r}_\tau \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + \tilde{e}_\tau \rceil_p
\end{array}
$$

We make the following observations:

1. Since the sampler D_χ is deterministic, if there exists an index $1 \le i^* \le \tau$ for which $r_{i^*} = \tilde{r}_{i^*}$, then this implies that $r_i = \tilde{r}_i$ for all $i^* \le i \le \tau$.

2. Since the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_\tau$ are sampled uniformly at random from $\mathbb{Z}_q^n$, the inner products $\langle \mathbf{a}, \mathbf{s}_i \rangle$ for any $1 \le i \le \tau$ are distributed statistically close to uniform in $\mathbb{Z}_q$.[3] Therefore, using (2.1), we have
$$\Pr\left[\lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p \neq \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + \tilde{e}_i \rceil_p\right] \le \frac{1}{2} + \mathsf{negl},$$
for any $1 \le i \le \tau$.

These observations imply that unless all of the inner products $\langle \mathbf{a}, \mathbf{s}_1 \rangle, \ldots, \langle \mathbf{a}, \mathbf{s}_\tau \rangle$ land on the "borderline" set of $\mathbb{Z}_q$, the samples of $\mathcal{O}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ coincide for $\mathbf{a} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. Furthermore, such bad event occurs with probability at most $\approx 1/2^\tau$. Hence, even for very small values of the chaining parameter $\tau = \omega(\log \lambda)$, with overwhelming probability over the matrix $\mathbf{S} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times\tau}$, no information about the initial error term e_1 is revealed from a single sample of $\mathcal{O}_{\tau,\mathbf{S}}$ or $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$. This can be extended to argue that no information about e_1 is leaked from any polynomial number of samples via the union bound. Hence, for $\tau = \omega(\log \lambda)$, any polynomial number of samples from $\mathcal{O}_{\tau,\mathbf{S}}$ or $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ are statistically indistinguishable.

In the discussion above, we set the modulus $q = 4Bp$ purely for simplicity. If we set q to be slightly greater than $2Bp$ (by a polynomial factor), the chaining parameter τ can be set to be any super-constant function $\tau = \omega(1)$ to guarantee that the output of the oracles $\mathcal{O}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ are statistically indistinguishable.

Synthesizer from LWRE. The oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ naturally induces a computational problem. Namely, for a set of parameters n, q, p, χ, and τ, we define the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem that asks an adversary to distinguish the samples $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \leftarrow \mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ from uniformly random samples $(\mathbf{a}_1, \hat{b}_1), \ldots, (\mathbf{a}_m, \hat{b}_m) \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \times \mathbb{Z}_p$. Using the ideas highlighted above, we can show that for small values of q and τ, the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ is at least as hard as the $\mathsf{LWE}_{n,m,q,\chi}$ problem. Specifically, we can first show that the oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ is statistically indistinguishable from $\mathcal{O}_{\tau,\mathbf{S}}$. Then, via a hybrid argument, we can show that the oracle $\mathcal{O}_{\tau,\mathbf{S}}$ is computationally indistinguishable from a uniform sampler over $\mathbb{Z}_q^n \times \mathbb{Z}_p$ by the hardness of $\mathsf{LWE}_{n,q,\chi}$.

[3]When the modulus q is prime, then the inner product $\langle \mathbf{a}, \mathbf{s}_i \rangle$ is certainly uniform in $\mathbb{Z}_q$. Even when q is composite (i.e. q is divisible by p), under mild requirements on q, the inner product $\langle \mathbf{a}, \mathbf{s}_i \rangle$ is statistically close to uniform in $\mathbb{Z}_q$.

The $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem naturally induces a pseudorandom synthesizer. One can first define an "almost" pseudorandom synthesizer $\mathcal{G} : \mathbb{Z}_q^n \times \mathbb{Z}_q^{n\times\tau} \to \mathbb{Z}_p$ that emulates the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ oracle as follows:

- $\mathcal{G}(\mathbf{a}, \mathbf{S})$: On input $\mathbf{a} \in \mathbb{Z}_q^n$ and $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n\times\tau}$, the LWRE function sets $e_1 = 0$ and computes for $i = 1, \ldots, \tau - 1$:
  1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$,
  2. $e_{i+1} \leftarrow D_\chi(r_i)$.

  It then sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$ and returns $b \in \mathbb{Z}_p$.

It is easy to see that as long as the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is hard, then for any $\ell = \mathsf{poly}(\lambda)$, $\mathbf{a}_1, \ldots, \mathbf{a}_\ell \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{S}_1, \ldots, \mathbf{S}_\ell \leftarrow \mathbb{Z}_q^{n\times\tau}$, and $u_{1,1}, \ldots, u_{\ell,\ell} \xleftarrow{\text{R}} \mathbb{Z}_p$, we have

$$\Big\{\mathcal{G}(\mathbf{a}_i, \mathbf{S}_j)\Big\}_{i,j\in[\ell]} \quad \approx_c \quad \Big\{u_{i,j}\Big\}_{i,j\in[\ell]} \quad \in \mathbb{Z}_p^{\ell^2}.$$

Furthermore, this indistinguishability holds even for small values of the chaining parameter $\tau = \omega(1)$ and hence, the function $\mathcal{G}$ can be computed by shallow circuits.

The only reason why $\mathcal{G} : \mathbb{Z}_q^n \times \mathbb{Z}_q^{n\times\tau} \to \mathbb{Z}_p$ is not a pseudorandom synthesizer is that the cardinality of the sets $\mathbb{Z}_q^n$, $\mathbb{Z}_q^{n\times\tau}$, and $\mathbb{Z}_p$ are different. However, this can easily be fixed by defining a synthesizer $S : (\mathbb{Z}_q^n)^{\ell_1} \times (\mathbb{Z}_q^{n\times\tau})^{\ell_2} \to (\mathbb{Z}_p)^{\ell_1\times\ell_2}$ that takes in a set of ℓ_1 vectors $(\mathbf{a}_1, \ldots, \mathbf{a}_\ell) \in (\mathbb{Z}_q^n)^{\ell_1}$, and ℓ_2 matrices $(\mathbf{S}_1, \ldots, \mathbf{S}_\tau) \in (\mathbb{Z}_q^{n\times\tau})^{\ell_2}$, and then returns $\{\mathcal{G}(\mathbf{a}_i, \mathbf{S}_j)\}_{i,j\in[\ell]}$. The parameters ℓ_1 and ℓ_2 can be set to be any positive integers such that

$$\big|(\mathbb{Z}_q^n)^{\ell_1}\big| = \big|(\mathbb{Z}_q^{n\times\tau})^{\ell_2}\big| = \big|\mathbb{Z}_p^{\ell_1\times\ell_2}\big|,$$

which makes S to be a two-to-one function over a fixed domain. The PRF that is induced by the synthesizer $S : (\mathbb{Z}_q^n)^{\ell_1} \times (\mathbb{Z}_q^{n\times\tau})^{\ell_2} \to (\mathbb{Z}_p)^{\ell_1\times\ell_2}$ corresponds to our first PRF construction.

We note that for practical implementation of the synthesizer, the large PRF key can be derived from a λ-bit PRG seed. Furthermore, the discrete Gaussian sampler D_χ can always be replaced by a suitable look-up table with pre-computed Gaussian noise as the modulus p is small. Therefore, the synthesizer can be implemented quite efficiently as it simply consists of τ inner products of two vectors modulo a small integer and their rounding. We refer to Section 4.3 for a discussion on the parameters and implementations.

2.3 Chaining Key-Homomorphic PRFs

The method of chaining multiple LWR/LWE samples can also be applied directly to existing lattice-based PRF constructions to improve their parameters. Furthermore, when applied to existing key-homomorphic PRFs, the resulting PRF is also key-homomorphic. Here, we demonstrate the idea with the PRF construction of [BLMR13] as it can be described quite compactly. In the technical section (Section 6), we show how to chain the PRF construction of [BP14] as it is a generalization of the previous PRF constructions of [BPR12, BLMR13] and it also allows us to set the parameters such that the underlying modulus q is only polynomial in the security parameter.

Background on BLMR [BLMR13]. Recall that the BLMR PRF is defined with respect to two public binary matrices $\mathbf{A}_0, \mathbf{A}_1 \in \{0,1\}^{n\times n}$ for a suitable choice of $n = \mathsf{poly}(\lambda)$. A PRF key is set to be a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and the PRF evaluation for an input $x \in \{0,1\}^\ell$ is defined to be the rounded matrix product

$$F^{(\mathsf{BLMR})}(\mathbf{s}, x) = \Big\lfloor \mathbf{s}^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \Big\rceil_p .$$

We can reason about the security of the BLMR PRF by considering its "noisy" variant that is defined as follows:

$F^{(\mathsf{noise})}(\mathbf{s}, x)$:

1. Sample error vectors $\mathbf{e}_1, \ldots, \mathbf{e}_\ell \leftarrow \chi^n$,
2. Return the vector

$$\Big\lfloor \Big(\big((\mathbf{s}^\intercal \mathbf{A}_{x_1} + \mathbf{e}_1^\intercal) \cdot \mathbf{A}_{x_2} + \mathbf{e}_2^\intercal \big) \cdots \Big) \cdot \mathbf{A}_{x_\ell} + \mathbf{e}_\ell^\intercal \Big\rceil_p$$
$$= \Big\lfloor \mathbf{s}^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} + \underbrace{\sum_{i=1}^{\ell-1} \mathbf{e}_i^\intercal \prod_{j=i+1}^{\ell-1} \mathbf{A}_{x_j} + \mathbf{e}_\ell^\intercal}_{\mathbf{e}^*} \Big\rceil_p .$$

Since the error vectors $\mathbf{e}_1, \ldots, \mathbf{e}_\ell \leftarrow \chi^n$ has small norm and the matrices $\mathbf{A}_0, \mathbf{A}_1 \in \{0,1\}^{n \times n}$ are binary, the error term $\mathbf{e}^*$ is also small. Therefore, if the modulus q is sufficiently big, then with overwhelming probability, the error vector $\mathbf{e}^*$ is "rounded away" and does not contribute to the final output of the PRF. This shows that when the modulus q is big, the evaluations of the functions $F(\mathbf{s}, \cdot)$ and $F^{(\mathsf{noise})}(\mathbf{s}, \cdot)$ are statistically indistinguishable.

Now, it is easy to show that $F^{(\mathsf{noise})}(\mathbf{s}, \cdot)$ is computationally indistinguishable from a truly random function using the hardness of the $\mathsf{LWE}_{n,q,\chi}$ problem.[4] We can first argue that the vector $\mathbf{s}_1^\intercal \mathbf{A}_{1,x_1} + \mathbf{e}_1^\intercal$ is computationally indistinguishable from a uniformly random vector $\mathbf{s}_2 \xleftarrow{\text{R}} \mathbb{Z}_q^n$. This implies that the vector $\mathbf{s}_2^\intercal \mathbf{A}_{2,x_2} + \mathbf{e}_2^\intercal$ is computationally indistinguishable from a random vector $\mathbf{s}_3 \xleftarrow{\text{R}} \mathbb{Z}_q^n$. We can repeat the argument for ℓ steps to prove that the final output of the PRF is computationally indistinguishable from a uniformly random output.

Chaining BLMR. For the security argument of the BLMR PRF to be valid, it is crucial that the modulus q is large enough such that the error term $\mathbf{e}^*$ rounds away with the modular rounding operation. Specifically, the modulus q must be set to be greater than the maximum possible norm of the error term $\|\mathbf{e}^*\|$ by a super-polynomial factor in the security parameter.

To prevent this blow-up in the size of q, we can chain multiple instances of the functions $F^{(\mathsf{BLMR})}$ and $F^{(\mathsf{noise})}$ together. Consider the following chained PRF $F^{(\mathsf{chain})} : \mathbb{Z}_q^{n \times \tau} \times \{0,1\}^\ell \to \mathbb{Z}_p^n$:

$F^{(\mathsf{chain})}\big(\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau), x\big)$:

1. Evaluate $r_2 \leftarrow F^{(\mathsf{BLMR})}(\mathbf{s}_1, x)$.
2. For $i = 2, \ldots, \tau - 1$, compute:
   - $r_{i+1} \leftarrow F^{(\mathsf{noise})}(\mathbf{s}_i, x; r_i)$.

[4] Technically, the reduction uses the hardness of the *non-uniform* variant of the LWE problem [BLMR13] where the challenger samples the public vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m$ uniformly at random from $\{0,1\}^n$ as opposed to sampling them from $\mathbb{Z}_q^n$. The work of [BLMR13] shows that this variant of the LWE problem is as hard as the traditional version of LWE for suitable choices of parameters.

3. Return $F^{(\mathsf{noise})}(\mathbf{s}_\tau, x; r_\tau)$.

In words, the chained PRF $F^{(\mathsf{chain})}(\mathbf{s}, x)$ evaluates the "random coins" that are needed to evaluate the randomized PRF $F^{(\mathsf{noise})}(\mathbf{s}_i, x)$ from the previous execution of $F^{(\mathsf{noise})}(\mathbf{s}_{i-1}, x)$. The initial random coins are derived from the errorless BLMR PRF $F^{(\mathsf{BLMR})}(\mathbf{s}_1, x)$.

We can prove that the chained PRF $F^{(\mathsf{chain})}$ is secure using the same argument that was used to show the hardness of the LWRE problem. Namely, we first argue that even if the modulus q is greater than the maximum possible norm of the error term $\|\mathbf{e}^*\|$ only by a polynomial factor, for any two random coins r_i, r_i', we have $F^{(\mathsf{noise})}(\mathbf{s}_i, x; r_i) = F^{(\mathsf{noise})}(\mathbf{s}_i, x; r_i')$ with noticeable probability. Therefore, if we set τ to be sufficiently big, then we can replace $F^{(\mathsf{BLMR})}(\mathbf{s}_1, \cdot)$ with the randomized function $F^{(\mathsf{noise})}(\mathbf{s}_1, \cdot)$ without changing the output of the PRF. Now, we can use the fact that $F^{(\mathsf{noise})}(\mathbf{s}_1, \cdot)$ is computationally indistinguishable from a truly random function to argue that the function $F^{(\mathsf{chain})}(\mathbf{S}, \cdot)$ is computationally indistinguishable from a truly random function.

The parameters for the chained PRF $F^{(\mathsf{chain})}$ provide a trade-off between the depth of the evaluation circuit and the size of the modulus q (and therefore, the quality of the LWE assumption). If we set $\tau = 1$, then we recover the original BLMR PRF, which requires very large values of the modulus q. As we increase τ, the modulus q can be set to be arbitrarily close to the maximum possible value of the error vector $\|\mathbf{e}^*\|$ for $F^{(\mathsf{noise})}(\mathbf{s}, x)$. For the Banerjee-Peikert PRF [BP14], the maximum possible value of the error vector $\|\mathbf{e}^*\|$ can be made to be only polynomial in the security parameter, thereby allowing us to set q to be a polynomial function of the security parameter.

Key-homomorphism. The BLMR PRF is key-homomorphic because the modular rounding operation is an almost linear operation. Namely, for any input $x \in \{0,1\}^\ell$ and any two keys $\mathbf{s}, \tilde{\mathbf{s}} \in \mathbb{Z}_q^n$, we have

$$\begin{aligned} F^{(\mathsf{BLMR})}(\mathbf{s}, x) + F^{(\mathsf{BLMR})}(\tilde{\mathbf{s}}, x) &= \Big\lfloor \mathbf{s}^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \Big\rceil_p + \Big\lfloor \tilde{\mathbf{s}}^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \Big\rceil_p \\ &\approx \Big\lfloor (\mathbf{s} + \tilde{\mathbf{s}})^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \Big\rceil_p \\ &= F^{(\mathsf{BLMR})}(\mathbf{s} + \tilde{\mathbf{s}}, x). \end{aligned}$$

Due to this algebraic structure, the "noisy" variant of the BLMR PRF is also key-homomorphic. Namely, for any input $x \in \{0,1\}^\ell$, any two keys $\mathbf{s}, \tilde{\mathbf{s}} \in \mathbb{Z}_q^n$, and any *random coins* $r, \tilde{r}, r'$ that is used to sample the noise, we have

$$\begin{aligned} F^{(\mathsf{noise})}(\mathbf{s}, x; r) + F^{(\mathsf{noise})}(\tilde{\mathbf{s}}, x; \tilde{r}) &= \Big\lfloor (\mathbf{s} + \tilde{\mathbf{s}})^\intercal \prod_{i=1}^{\ell} \mathbf{A}_{x_i} + (\mathbf{e}^* + \tilde{\mathbf{e}}^*) \Big\rceil_p \\ &\approx F^{(\mathsf{noise})}(\mathbf{s} + \tilde{\mathbf{s}}, x; r'), \end{aligned}$$

Now, note that the final output of the chained PRF $F^{(\mathsf{chain})}$ on an input $x \in \{0,1\}^\ell$ and a key $(\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n \times \tau}$ with chaining parameter τ is simply the output of the noisy PRF $F^{(\mathsf{noise})}(\mathbf{s}_\tau, x; r_\tau)$ where r_τ is the randomness that is derived from the previous execution of $r_\tau \leftarrow F^{(\mathsf{noise})}(\mathbf{s}_{\tau-1}, x; r_{\tau-1})$.

Therefore, for any input $x \in \{0,1\}^\ell$, and two keys $\mathbf{S}, \tilde{\mathbf{S}} \in \mathbb{Z}_q^{n\times\tau}$, we can show that

$$\begin{aligned} F^{(\mathsf{chain})}(\mathbf{S}, x) + F^{(\mathsf{chain})}(\tilde{\mathbf{S}}, x) &= F^{(\mathsf{noise})}(\mathbf{s}_\tau, x; r_\tau) + F^{(\mathsf{noise})}(\tilde{\mathbf{s}}_\tau, x; \tilde{r}_\tau) \\ &\approx F^{(\mathsf{noise})}(\mathbf{s}_\tau + \tilde{\mathbf{s}}_\tau, x) \\ &\approx F^{(\mathsf{chain})}(\mathbf{S} + \tilde{\mathbf{S}}, x) \end{aligned}$$

Specifically, we can show that for a suitable choice of the modulus q, the resulting PRF is *2-almost* key-homomorphic in that for any input $x \in \{0,1\}^\ell$ and two keys $\mathbf{S}, \tilde{\mathbf{S}} \in \mathbb{Z}_q^{n\times\tau}$, there exists an error vector $\boldsymbol{\eta} \in [0,2]^n$ such that

$$F^{(\mathsf{chain})}(\mathbf{S}, x) + F^{(\mathsf{chain})}(\tilde{\mathbf{S}}, x) = F^{(\mathsf{chain})}(\mathbf{S} + \tilde{\mathbf{S}}, x) + \boldsymbol{\eta}.$$

The exact argument to show that the chained BLMR PRF is 2-almost key-homomorphic can be used to show that the chained BP PRF [BP14] is also 2-almost key-homomorphic. We provide the formal details in Section 6.

3 Preliminaries

Basic notations. Unless specified otherwise, we use λ to denote the security parameter. We say a function $f(\lambda)$ is negligible in λ, denoted by $\mathsf{negl}(\lambda)$, if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We say that a function $f(\lambda)$ is noticeable in λ if $f(\lambda) = \Omega(1/\lambda^c)$ for some $c \in \mathbb{N}$. We say that an event happens with overwhelming probability if its complement happens with negligible probability. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We use $\mathsf{poly}(\lambda)$ to denote a quantity whose value is bounded by a fixed polynomial in λ.

For an integer $n \geq 1$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled from $\mathcal{D}$; for a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is sampled uniformly from S. For a positive integer B, we say that a distribution $\mathcal{D}$ over $\mathbb{Z}$ is B-bounded if $\Pr[x \leftarrow \mathcal{D} \wedge |x| > B]$ is negligible. Finally, we write $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions mapping from a domain $\mathcal{X}$ to a range $\mathcal{Y}$.

Vectors and matrices. We use bold lowercase letters (*e.g.*, $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (*e.g.*, $\mathbf{A}, \mathbf{B}$) to denote matrices. Throughout this work, we always use the infinity norm for vectors and matrices. Therefore, for a vector $\mathbf{x}$, we write $\|\mathbf{x}\|$ to denote $\max_i |x_i|$. Similarly, for a matrix $\mathbf{A}$, we write $\|\mathbf{A}\|$ to denote $\max_{i,j} |A_{i,j}|$. If $\mathbf{x} \in \mathbb{Z}^n$ and $\mathbf{A} \in \mathbb{Z}^{n\times m}$, then $\|\mathbf{x}\mathbf{A}\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$.

Modular rounding. For an integer $p \leq q$, we define the modular "rounding" function

$$\lfloor\cdot\rceil_p \colon \mathbb{Z}_q \to \mathbb{Z}_p \text{ that maps } x \to \lfloor (p/q) \cdot x \rceil$$

and extend it coordinate-wise to matrices and vectors over $\mathbb{Z}_q$. Here, the operation $\lfloor\cdot\rceil$ is the integer rounding operation over the real numbers. It can be readily checked that for any two values $x, y \in \mathbb{Z}_q$, there exists some $\eta \in \{0,1\}$ such that $\lfloor x\rceil_p + \lfloor y\rceil_p = \lfloor x+y\rceil_p + \eta$.

Bit-decomposition. Let n and q be positive integers. Then we define the "gadget matrix" $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n\times n\cdot\lceil\log q\rceil}$ where $\mathbf{g} = (1, 2, 4, \ldots, 2^{\lceil\log q\rceil - 1})$. We define the inverse bit-decomposition function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n\times m} \to \mathbb{Z}_q^{n\lceil\log q\rceil\times m}$ which expands each entry $x \in \mathbb{Z}_q$ in the input matrix into a column of size $\lceil\log q\rceil$ that consists of the bits of the binary representation of x.

3.1 Learning with Errors

In this section, we define the Learning with Errors (LWE) problem [Reg09] and the Ring-LWE (RLWE) problem [LPR10].

Learning with Errors. We define the LWE problem with respect to the real and ideal oracles.

Definition 3.1 (Learning with Errors)**.** Let $\lambda \in \mathbb{N}$ be the security parameter. Then the *learning with errors* (LWE) problem is parameterized by a dimension $n = n(\lambda)$, modulus $q = q(\lambda)$, and error distribution $\chi = \chi(\lambda)$. It is defined with respect to the real and ideal oracles $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ and $\mathcal{O}^{(\mathsf{ideal})}$ that are defined as follows:

- $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$: The real oracle is parameterized by a vector $\mathbf{s} \in \mathbb{Z}_q^n$. On its invocation, the oracle samples a random vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, and an error term $e \leftarrow \chi$. It sets $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

- $\mathcal{O}^{(\mathsf{ideal})}$: On its invocation, the ideal oracle samples a random vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, random element $u \xleftarrow{\text{R}} \mathbb{Z}_q$, and returns $(\mathbf{a}, u) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

The $\mathsf{LWE}_{n,q,\chi}$ problem is to distinguish the oracles $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ and $\mathcal{O}^{(\mathsf{ideal})}$. More precisely, we define an adversary $\mathcal{A}$'s distinguishing advantage $\mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A})$ as the probability

$$\mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{ideal})}}(1^\lambda) = 1\right] \right|,$$

where $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$. The $\mathsf{LWE}_{n,q,\chi}$ assumption state that for any efficient adversary $\mathcal{A}$, its distinguishing advantage is negligible $\mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}) = \mathsf{negl}(\lambda)$.

Compactly, the $\mathsf{LWE}_{n,q,\chi}$ assumption states that for any $m = \mathsf{poly}(\lambda)$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, the noisy vector-matrix product $(\mathbf{A}, \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal)$ is computationally indistinguishable from $(\mathbf{A}, \mathbf{u}^\intercal)$. It follows from a standard hybrid argument that for any $m, \ell = \mathsf{poly}(\lambda)$, $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{\ell \times n}$, $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{E} \leftarrow \chi^{\ell \times m}$, and $\mathbf{U} \xleftarrow{\text{R}} \mathbb{Z}_q^{\ell \times m}$, the noisy matrix product $(\mathbf{A}, \mathbf{S} \cdot \mathbf{A} + \mathbf{E})$ is computationally indistinguishable from $(\mathbf{A}, \mathbf{U})$ by the $\mathsf{LWE}_{n,q,\chi}$ assumption.

Let $n = \mathsf{poly}(\lambda)$ and χ be a B-bounded discrete Gaussian distribution. Then the $\mathsf{LWE}_{n,q,\chi}$ assumption is true assuming that various worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices are hard to approximate to within a factor of $\tilde{O}(n \cdot q/B)$ by a quantum algorithm [Reg09]. Similar reductions of $\mathsf{LWE}_{n,q,\chi}$ to the *classical* hardness of approximating worst-case lattice problems are also known [Pei09, ACPS09, MM11, MP12, BLP+13].

Ring LWE. For simplicity of exposition, we use a special case of the ring-LWE problem throughout this work. However, all of our definitions and results on ring-LWE and ring-LWRE (Section 5) can be extended to the more general form as defined in [LPR10, PRSD17].[5] Throughout the paper, we always let R denote the cyclotomic polynomial ring $R = \mathbb{Z}[X]/(X^n + 1)$ for a power-of-two integer n. For any integer modulus q, we let R_q denote the quotient ring $R_q = R/qR \cong \mathbb{Z}_q[X]/(X^n + 1)$. A ring element of R is a polynomial in X and therefore, can be represented as a vector of the

[5]The main techniques that we use in this work are independent of the underlying ring of the Learning with Errors problem and therefore, they can be applied to other "algebraically structured" LWE variants that exist in the literature [SSTX09, BGV14, LS15, RSW18, BBPS18, RSSS17].

integer coefficients of the polynomial. Similarly, a ring element in R_q can be represented as a vector of coefficients in $\mathbb{Z}_q$. For any element $b \in R$, we define its norm $\|b\|$ as the norm of its vector representation. For $p < q$ and any $b \in R_q$, we let $\lfloor b \rceil_p \in R_p$ to denote the modular rounding of each coefficient of b as a polynomial in X.

Definition 3.2 (Ring-LWE)**.** Let $\lambda \in \mathbb{N}$ be the security parameter. Then the *ring learning with errors* (RLWE) problem is parameterized by a power-of-two integer $n = n(\lambda)$, any modulus $q = q(\lambda)$, and an error distribution $\chi = \chi(\lambda)$ over the cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$. It is defined with respect to the real and ideal oracles $\mathcal{O}_s^{(\mathsf{rlwe})}$ and $\mathcal{O}^{(\mathsf{ideal})}$ that are defined as follows:

- $\mathcal{O}_s^{(\mathsf{rlwe})}$: The real oracle is parameterized by a ring element $s \in R_q$. On its invocation, the oracle samples a random vector $a \xleftarrow{\text{R}} R_q$, and an error term $e \leftarrow \chi$. It sets $b = a \cdot s + e$, and returns $(a, b) \in R_q \times R_q$.

- $\mathcal{O}^{(\mathsf{ideal})}$: On its invocation, the ideal oracle samples random elements $a, b \xleftarrow{\text{R}} R_q$, and returns $(a, b) \in R_q \times R_q$.

The $\mathsf{RLWE}_{n,q,\chi}$ problem is to distinguish the oracles $\mathcal{O}_s^{(\mathsf{rlwe})}$ and $\mathcal{O}^{(\mathsf{ideal})}$. More precisely, we define an adversary $\mathcal{A}$'s distinguishing advantage $\mathsf{Adv}_{\mathsf{RLWE}}(n, q, \chi, \mathcal{A})$ as the probability

$$\mathsf{Adv}_{\mathsf{RLWE}}(n, q, \chi, \mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{O}_s^{(\mathsf{rlwe})}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{ideal})}}(1^\lambda) = 1\right] \right|,$$

where $s \xleftarrow{\text{R}} R_q$. The $\mathsf{RLWE}_{n,q,\chi}$ assumption states that for any efficient adversary $\mathcal{A}$, its distinguishing advantage is negligible $\mathsf{Adv}_{\mathsf{RLWE}}(n, q, \chi, \mathcal{A}) = \mathsf{negl}(\lambda)$.

Let $n = \mathsf{poly}(\lambda)$ and χ be a B-bounded discrete Gaussian distribution over R. Then the $\mathsf{LWE}_{n,q,\chi}$ assumption is true assuming that certain worst-case lattice problems such as SVP on n-dimensional ideal lattices are hard to approximate to within $\mathsf{poly}(n) \cdot q/B$ by a quantum algorithm [LPR10, LPR13, LS15].

3.2 Elementary Number Theory

In this section, we state and prove an elementary fact in number theory that we use for our technical sections. Specifically, we analyze the distribution of the inner product of two uniformly random vectors $\langle \mathbf{a}, \mathbf{s} \rangle$ where $\mathbf{a}, \mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ for some positive integers n and q. When n is sufficiently big, then with overwhelming probability, one of the components of $\mathbf{a}$ (or $\mathbf{s}$) will be a multiplicative unit in $\mathbb{Z}_q$ and therefore, the inner product $\langle \mathbf{a}, \mathbf{s} \rangle$ will be uniform in $\mathbb{Z}_q$. We formally state and prove this elementary fact in the lemma below. We first recall a general fact of the Euler totient function.

Fact 3.3 ([Lan00])**.** Let $\varphi : \mathbb{Z} \to \mathbb{Z}$ be the Euler totient function. Then, there exists a constant c such that for any $q > 264$, we have $q/\varphi(q) = c \cdot \log \log q$.

The following lemma follows immediately from Fact 3.3.

Lemma 3.4. *Let $n = n(\lambda)$ and $q = q(\lambda)$ be positive integers such that $n = \Omega(\lambda \log \log q)$. Then, for any element $d \in \mathbb{Z}_q$, we have*

$$\Pr[\ \langle \mathbf{a}, \mathbf{s} \rangle = d\ ] \leq 1/q + 2^{-\lambda},$$

for $\mathbf{a}, \mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$.

Proof. If the vector $\mathbf{a} \in \mathbb{Z}_q^n$ has at least one component that is a multiplicative unit in $\mathbb{Z}_q$, then since $\mathbf{s}$ is sampled uniformly at random from $\mathbb{Z}_q^n$, we have $\langle \mathbf{a}, \mathbf{s} \rangle = d$ with probability $1/q$. By Fact 3.3, the probability that all components of $\mathbf{a}$ is *not* a unit in $\mathbb{Z}_q$ is bounded by the probability

$$\left(1 - \frac{1}{c \cdot \log \log q}\right)^n$$

for some constant c. Setting $\lambda = \Omega(n/\log \log q)$, this probability is bounded by $e^{-\lambda} < 2^{-\lambda}$. The lemma follows. □

3.3 Pseudorandom Functions and Key-Homomorphic PRFs

In this section, we formally define pseudorandom functions [GGM86] and key-homomorphic PRFs [BLMR13].

Definition 3.5 (Pseudorandom Functions [GGM86])**.** A *pseudorandom function* for a key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently computable deterministic function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for any efficient adversary $\mathcal{A}$, we have

$$\left| \Pr\left[k \xleftarrow{\text{R}} \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1\right] - \Pr\left[f \xleftarrow{\text{R}} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}] : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1\right] \right| = \mathsf{negl}(\lambda),$$

We generally refer to the experiment where the adversary $\mathcal{A}$ is given oracle access to the real PRF $F(k, \cdot)$ as the *real* PRF experiment. Analogously, we refer to the experiment where the adversary $\mathcal{A}$ is given oracle access to a truly random function $f(\cdot)$ as the *ideal* PRF experiment.

Key-homomorphic PRFs are special family of pseudorandom function that satisfy an additional algebraic property. Specifically, for a key-homomorphic PRF, the key space $\mathcal{K}$ and the range $\mathcal{Y}$ of the PRF exhibit certain group structures such that its evaluation on any fixed input $x \in \mathcal{X}$ is homomorphic with respect to these group structures. Formally, we define a key-homomorphic PRF as follows.

Definition 3.6 (Key-Homomorphic PRFs [NPR99, BLMR13])**.** Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, \otimes)$ be groups. Then, an efficiently computable deterministic function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a *key-homomorphic* PRF if

- F is a secure PRF (Definition 3.5).
- For every key $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$, we have $F(k_1, x) \otimes F(k_2, x) = F(k_1 \oplus k_2, x)$.

In this work, we will work with a slight relaxation of the notion of key-homomorphic PRFs. Namely, instead of requiring that the PRF outputs are perfectly homomorphic with respect to the PRF keys, we require that they are "almost" homomorphic in that $F(k_1, x) \otimes F(k_2, x) \approx F(k_1 \oplus k_2, x)$. Precisely, we define an almost key-homomorphic PRF as follows.

Definition 3.7 (Almost Key-Homomorphic PRFs [BLMR13])**.** Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, \otimes)$ be groups and let m and p be positive integers. Then, an efficiently computable deterministic function $F : \mathcal{K} \times \mathcal{X} \to \mathbb{Z}_p^m$ is a γ-almost key-homomorphic PRF if

- F is a secure PRF (Definition 3.5).
- For every key $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$, there exists a vector $\mathbf{e} \in [0, \gamma]^m$ such that

$$F(k_1, x) + F(k_2, x) = F(k_1 \oplus k_2, x) + \mathbf{e} \pmod{p}.$$

Naor et al. [NPR99] and Boneh et al. [BLMR13] gave a number of applications of (almost) key-homomorphic PRFs including distributed PRFs, symmetric-key proxy re-encryption, updatable encryption, and PRFs secure against related-key attacks.

4 Learning with Rounding and Errors

In this section, we present our new lattice-based synthesizer construction. We first define the Learning with Rounding and Errors (LWRE) problem in Section 4.1. We then show how to use the LWRE problem to construct a synthesizer in Section 4.2 and discuss its parameters in Section 4.3. We show that the LWRE problem is as hard as the standard LWE problem for suitable choices of parameters in Section A.

4.1 Learning with Rounding and Errors

Definition 4.1 (Learning with Rounding and Errors)**.** Let $\lambda \in \mathbb{N}$ be the security parameter. The *learning with rounding and errors* (LWRE) problem is defined with respect to the parameters

- LWE parameters (n, q, χ),
- Rounding modulus $p \in \mathbb{N}$ such that $p < q$,
- Chaining parameter $\tau \in \mathbb{N}$,

that are defined as functions of λ. Additionally, let $D_\chi : \{0,1\}^{\lfloor \log p \rfloor} \to \mathbb{Z}$ be a sampling algorithm for the error distribution χ. Then, we define the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ real and ideal oracles $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{ideal})}$ as follows:

- $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$: The real oracle is defined with respect to a chaining parameter $\tau \in \mathbb{N}$, and a secret matrix $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n \times \tau}$. On its invocation, the real oracle samples a vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and initializes $e_1 = 0$. Then, for $1 \le i < \tau$, it iteratively computes:

  1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
  2. $e_{i+1} \leftarrow D_\chi(r_i)$.

  It then sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.

- $\mathcal{O}^{(\mathsf{ideal})}$: On its invocation, the ideal oracle samples a random vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, a random element $u \xleftarrow{\text{R}} \mathbb{Z}_p$, and returns $(\mathbf{a}, u) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.

The $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is to distinguish the oracles $\mathcal{O}^{(\mathsf{lwre})}_{\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{ideal})}$ for $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \tau}$. More precisely, we define an adversary $\mathcal{A}$'s distinguishing advantage $\mathsf{Adv}_{\mathsf{LWRE}}(n, q, p, \chi, \tau, \mathcal{A})$ as the probability

$$\mathsf{Adv}_{\mathsf{LWRE}}(n, q, p, \chi, \tau, \mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{ideal})}}(1^\lambda) = 1\right] \right|,$$

for $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \tau}$.

It is easy to see that when $\tau = 1$, the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is identical to the standard Learning with Rounding problem [BPR12]. Hence, the reduction in [BPR12] immediately shows that for $\tau = 1$, if the modulus q is sufficiently large such that $q = 2Bpn^{\omega(1)}$, then the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is as hard as the $\mathsf{LWE}_{n,q,\chi}$ problem. We show that when τ is set to be larger, then the modulus q can be set to be significantly smaller.

Theorem 4.2. *Let λ be the security parameter and n, q, p, χ, τ be a set of parameters for the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem such that p divides q. Then, for any efficient adversary $\mathcal{A}$ making at most Q number of oracle calls, we have*

$$\mathsf{Adv}_{\mathsf{LWRE}}(n, q, p, \chi, \tau, \mathcal{A}) \leq Q\big(2Bp/q + 1/2^{\lambda}\big)^{\tau} + \tau \cdot \mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}).$$

In particular, if $Q(2Bp/q + 1/2^{\lambda})^{\tau} = \mathsf{negl}(\lambda)$, then the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is as hard as the $\mathsf{LWE}_{n,q,\chi}$ problem.

We provide the proof of the theorem in Section A. We provide the high-level ideas of the proof in Section 2.

Remark 4.3. The $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem is well-defined only when there exists a concrete sampler $D_\chi : \{0,1\}^{\lfloor \log p \rfloor} \to \mathbb{Z}$, which uses at most $\lfloor \log p \rfloor$ random bits to sample from χ. For the discrete Gaussian distribution (over $\mathbb{Z}$) with Gaussian parameter $\sigma > \sqrt{n}$, there exist Gaussian samplers (i.e., [GPV08]) that require $O(\log \lambda)$ random bits. Therefore, one can always set $p = \mathsf{poly}(\lambda)$ to be big enough such that $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ is well defined for the discrete Gaussian distribution.

To set p to be even smaller, one can alternatively use a pseudorandom generator (PRG) to stretch the random coins that are needed by the sampler. A single element in $\mathbb{Z}_p$ for $p = \mathsf{poly}(\lambda)$ is not large enough to serve as a seed for a PRG. However, one can modify the oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ such that it samples multiples vectors $\mathbf{a}_1, \ldots, \mathbf{a}_\ell \xleftarrow{\text{R}} \mathbb{Z}_q^n$ for some $\ell = \mathsf{poly}(\lambda)$ and then derives a vector of elements $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,\ell}) \in \mathbb{Z}_p^\ell$ that can serve as a seed for any (lattice-based) pseudorandom generator.

Remark 4.4. We note that in Theorem 4.2, we impose the requirement that p perfectly divides q. This requirement is needed purely to guarantee that the rounding $\lfloor r \rceil_p$ of a uniformly random element $r \xleftarrow{\text{R}} \mathbb{Z}_q$ results in a uniformly random element in $\mathbb{Z}_p$. However, even when q is not perfectly divisible by p (i.e. q is prime), the rounding $\lfloor r \rceil_p$ of $r \xleftarrow{\text{R}} \mathbb{Z}_q$ still results in a highly unpredictable element in $\mathbb{Z}_p$. Therefore, by modifying the oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ such that it applies a randomness extractor after each of the modular rounding operation, one can remove the requirement on the structure of q with respect to p.

4.2 Pseudorandom Synthesizers from LWRE

In this section, we construct our new pseudorandom synthesizer from the hardness of the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem. A pseudorandom synthesizer over a domain $\mathcal{D}$ is a function $S : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ satisfying a specific pseudorandomness property as formulated below. We first recall the formal definition as presented in [NR99]. As observed in [BPR12], one can also relax the traditional definition of a pseudorandom synthesizer by allowing the synthesizer function $S : \mathcal{D}_1 \times \mathcal{D}_1 \to \mathcal{D}_2$ to have differing domain $\mathcal{D}_1$ and range $\mathcal{D}_2$. A synthesizer satisfying this relaxed definition still induces a PRF as long as the function can be applied iteratively. For this work, we restrict to the original definition for a simpler presentation.

Definition 4.5 (Pseudorandom Synthesizer [NR99])**.** Let $\mathcal{D}$ be a finite set. An efficiently computable function $S : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ is a secure *pseudorandom synthesizer* if for *any* polynomial $\ell = \ell(\lambda)$, the following distributions are computational indistinguishable

$$\{S(a_i, b_j)\}_{i,j \in [\ell]} \quad \approx_{\mathsf{c}} \quad \{u_{i,j}\}_{i,j \in [\ell]},$$

where $(a_1, \ldots, a_\ell) \xleftarrow{R} \mathcal{D}^\ell$, $(b_1, \ldots, b_\ell) \xleftarrow{R} \mathcal{D}^\ell$, and $(u_{i,j})_{i,j \in [\ell]} \leftarrow \mathcal{D}^{\ell \times \ell}$.

Naor and Reingold [NR99] showed that a secure pseudorandom synthesizer induces a secure pseudorandom function. Furthermore, if the synthesizer can be computed by a low-depth circuit, then the final PRF can also be evaluated by a low-depth circuit. We formally state their result in the following theorem.

Theorem 4.6 ([NR99, BPR12])**.** *Suppose that there exists a pseudorandom synthesizer $S : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ over a finite set $\mathcal{D}$ that is computable by a circuit of size s and depth d. Then, for any $\ell = \mathsf{poly}(\lambda)$, there exists a pseudorandom function $F : \mathcal{D}^{2\ell} \times \{0,1\}^\ell \to \mathcal{D}$ with key space $\mathcal{D}^{2\ell}$, domain $\{0,1\}^\ell$, and range $\mathcal{D}$ that is computable by a circuit of size $O(s\ell)$ and depth $O(d \log \ell)$.*

As was shown in [BPR12], the hardness of the Learning with Rounding (LWR) problem (Definition 4.1 for $\tau = 1$) naturally induces a secure pseudorandom synthesizer $S : \mathbb{Z}_q^{n \times n} \to \mathbb{Z}_q^{n \times n} \to \mathbb{Z}_p^{n \times n}$ that can be compactly defined as $S(\mathbf{S}, \mathbf{A}) = \lfloor \mathbf{S} \cdot \mathbf{A} \rceil_p$.[6] One can naturally extend this construction to $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ for $\tau > 1$. Unfortunately, as the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ requires the chaining of many samples, the synthesizer does not exhibit a compact description like the LWR synthesizer. We describe the LWRE synthesizer in two steps. We first define an *LWRE function*, which satisfies the security requirement of a pseudorandom synthesizer, but does not satisfy the strict restriction on the domain and range of a synthesizer. Then, we show how to modify the LWRE function to achieve a synthesizer that satisfies Definition 4.5.

Definition 4.7 (LWRE Function)**.** Let λ be the security parameter and let n, q, p, χ, τ be a set of LWRE parameters. Then, we define the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ function $\mathcal{G}_{n,q,p,\chi,\tau} : \mathbb{Z}_q^n \times \mathbb{Z}_q^{n \times \tau} \to \mathbb{Z}_p$ as follows:

- $\mathcal{G}_{n,q,p,\chi,\tau}(\mathbf{a}, \mathbf{S})$: On input $\mathbf{a} \in \mathbb{Z}_q^n$ and $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n \times \tau}$, the LWRE function sets $e_1 = 0$ and computes for $1 \le i < \tau$:

  1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$,
  2. $e_{i+1} \leftarrow D(r_i)$.

  It then sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$ and returns $b \in \mathbb{Z}_p$.

For any $\ell = \mathsf{poly}(\lambda)$, we can use a standard hybrid argument to show that for $\mathbf{a}_1, \ldots, \mathbf{a}_\ell \xleftarrow{R} \mathbb{Z}_q^n$ and $\mathbf{S}_1, \ldots, \mathbf{S}_\ell \xleftarrow{R} \mathbb{Z}_q^{n \times \tau}$, the set of elements $\{\mathcal{G}_{n,q,p,\chi,\tau}(\mathbf{a}_i, \mathbf{S}_j)\}_{i,j \in [\ell]}$ are computationally indistinguishable from ℓ^2 uniformly random elements in $\mathbb{Z}_p$. It readily follows that for any $\ell_1, \ell_2 = \mathsf{poly}(\lambda)$, the function $S : \mathbb{Z}_q^{n \times \ell_1} \times (\mathbb{Z}_q^{n \times \tau})^{\ell_2} \to \mathbb{Z}_p^{\ell_1 \times \ell_2}$ that takes in as input $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_{\ell_1}) \xleftarrow{R} \mathbb{Z}_q^{n \times \ell_1}$, $(\mathbf{S}_1, \ldots, \mathbf{S}_{\ell_2}) \xleftarrow{R} (\mathbb{Z}_q^{n \times \tau})^{\ell_2}$, and returns the pairwise application of the LWRE function

$$\left\{\mathcal{G}_{n,q,p,\chi,\tau}(\mathbf{a}_i, \mathbf{S}_j)\right\}_{i \in [\ell_1], j \in [\ell_2]}$$

satisfies the security requirements for a synthesizer. Therefore, as long as ℓ_1 and ℓ_2 are set such that the cardinality of the sets $\mathbb{Z}_q^{n \times \ell_1}$, $(\mathbb{Z}_q^{n \times \tau})^{\ell_2}$, and $\mathbb{Z}_p^{\ell_1 \times \ell_2}$ have the same cardinality, the function $S : \mathbb{Z}_q^{n \times \ell_1} \times (\mathbb{Z}_q^{n \times \tau})^{\ell_2} \to \mathbb{Z}_p^{\ell_1 \times \ell_2}$ satisfies Definition 4.5. We can naturally set the parameters ℓ_1 and ℓ_2 as

$$\ell_1 = n\tau \left\lceil \frac{\log q}{\log p} \right\rceil, \quad \ell_2 = n \left\lceil \frac{\log q}{\log p} \right\rceil.$$

[6]The LWR synthesizer satisfies the more general definition of a pseudorandom synthesizer where the domain and range of the synthesizer can differ.

Formally, we define our LWRE synthesizer as follows.

Construction 4.8 (LWRE Synthesizer)**.** Let λ be the security parameter, let n, q, p, χ, τ be a set of LWRE parameters, and let $\ell = n\tau\lceil \log q / \log p \rceil$. We define the LWRE synthesizer $S : \mathbb{Z}_q^{n\times\ell} \times \mathbb{Z}_q^{n\times\ell} \to \mathbb{Z}_q^{n\times\ell}$ as follows:

- $S(\mathbf{A}, \mathbf{S})$: On input $\mathbf{A}, \mathbf{S} \in \mathbb{Z}_q^{n\times\ell}$, the synthesizer parses the matrices
  - $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_{\ell_1})$ where $\mathbf{a}_i \in \mathbb{Z}_q^n$ and $\ell_1 = n\tau\lceil \log q / \log p \rceil$,
  - $\mathbf{S} = (\mathbf{S}_1, \ldots, \mathbf{S}_{\ell_2})$ where $\mathbf{S}_j \in \mathbb{Z}_q^{n\times\tau}$ and $\ell_2 = n\lceil \log q / \log p \rceil$.

  Then, the synthesizer computes the $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ function $b_{i,j} \leftarrow \mathcal{G}_{n,q,p,\chi,\tau}(\mathbf{a}_i, \mathbf{S}_j)$ for all $i \in [\ell_1]$ and $j \in [\ell_2]$. It translates the bits of $\{b_{i,j}\}_{j\in[\ell_1], i\in[\ell_2]} \in \mathbb{Z}_p^{\ell_1\times\ell_2}$ as the representation of a matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times\ell}$. It returns $\mathbf{B}$.

We now state the formal security statement for the LWRE synthesizer in Construction 4.8. The proof follows immediately from Definitions 4.1, 4.7, and Theorem 4.2.

Theorem 4.9 (Security)**.** *Let λ be the security parameter and let n, q, p, χ, τ be a set of* LWRE *parameters. Then, assuming that the* $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ *problem is hard, the LWRE synthesizer in Construction 4.8 is a secure pseudorandom synthesizer (Definition 4.5).*

By combining Theorems 4.2, 4.6, and 4.9, we get the following corollary.

Corollary 4.10. *Let λ be a security parameter and $n = \mathsf{poly}(\lambda)$ be a positive integer. Then, there exists a pseudorandom function $F : \mathcal{K} \times \{0,1\}^{\mathsf{poly}(\lambda)} \to \mathcal{Y}$ that can be computed by a circuit in NC^3, and whose security can be reduced from the worst-case hardness of approximating* GapSVP *and* SIVP *to a polynomial approximation factor on n-dimensional lattices.*

We additionally dicuss the parameter choices for our PRF in Section 4.3.

4.3 Parameter Instantiations

In this section, we discuss the various ways of instantiating the parameters for the LWRE synthesizer from Construction 4.8. As discussed in Section 1.2, we consider two parameter settings that provide different level of security against an adversary. For *theoretical* security, we require that an efficient adversary's distinguishing advantage of the LWRE synthesizer to degrade super-polynomially in λ. For 2^λ-security, we require that an efficient adversary's advantage degrades exponentially in λ. By Theorem 4.2, the main factor that we consider in determining the security of the synthesizer is the term $(2Bp/q + 1/2^\lambda)^\tau$.

Theoretical security. For theoretical security, we must let $(2Bp/q + 1/2^\lambda)^\tau = \mathsf{negl}(\lambda)$. In one extreme, we can set $\tau = 1$ and q to be greater than $2Bp$ by a super-polynomial factor in λ, which reproduces the result of [BPR12].[7] As both vector-matrix multiplication and the rounding operation can be implemented in NC^1, the resulting PRF can be implemented in NC^2 for input space $\{0,1\}^\ell$ where $\ell = \mathsf{poly}(\lambda)$.

If we set $\tau = \omega(1)$, then we can set q to be any function that is greater than $2Bp$ by a polynomial factor in λ. In this case, when considering the depth of the evaluation circuit, we must take into

[7] We note that the term $1/2^\lambda$ is needed for Lemma 3.4. If q is set to be prime, then this factor can be ignored.

account the depth of the sampling algorithm D_χ for the error distribution χ. To base security on approximating worst-case lattice problems such as GapSVP or SIVP, we must set χ to be the discrete Gaussian distribution over $\mathbb{Z}$ with Gaussian parameter $\sigma > \sqrt{n}$. In this case, we can either use the rejection sampling algorithm of [GPV08] or pre-compute the samples for each possible seed for the sampler and use a look-up table. In both cases, we can guarantee that a random element in $\mathbb{Z}_p$ provides enough entropy to the Gaussian sampler by setting $p = \omega(\lambda^2)$.

Since the Gaussian function can be computed by an arithmetic circuit with depth $O(\log p)$, the rejection sampling algorithm can be implemented by a circuit of depth $\omega(\log \lambda \cdot \log\log \lambda)$. Therefore, the synthesizer can be evaluated by a circuit in $\mathsf{NC}^{2+\varepsilon}$ for any constant $\varepsilon > 0$ and the final PRF can be evaluated by a circuit in $\mathsf{NC}^{3+\varepsilon}$ for input space $\{0,1\}^\ell$ where $\ell = \mathsf{poly}(\lambda)$. When using a look-up table, the synthesizer can be evaluated by a circuit in $\mathsf{NC}^{1+\varepsilon}$ for any constant $\varepsilon > 0$, and the final PRF can be evaluated in $\mathsf{NC}^{2+\varepsilon}$.

2^λ-security. For 2^λ-security, we must let $(2Bp/q)^\tau = 1/2^{\Omega(\lambda)}$ or equivalently, $\tau \cdot \log q = \tilde{\Omega}(\lambda)$ for q prime. This provides a trade-off between the size of q and the chaining parameter τ, which dictates the depth needed to evaluate the PRF. In one extreme, we can set $\tau = 1$ and require the modulus to be greater than $2Bp$ by an exponential factor in λ. In the other extreme, we can let τ be linear in λ and decrease the modulus to be only a constant factor greater than $2Bp$. For practical implementations, a natural choice of parameters would be to set both τ and $\log q$ to be $\Omega(\sqrt{\lambda})$. For practical implementations, one can derive the secret keys from a single λ-bit seed using a pseudorandom generator.

Concrete instantiations The concrete parameters for our PRF can be instantiated quite flexibly depending on the applications. The modulus p can first be set to determine the output of the PRF. For instance, as the range of the PRF is $\mathbb{Z}_p$, the modulus p can be set to be 2^8 or 2^{16} such that the output of the PRF is byte-aligned. Then the PRF can be run multiple times (in parallel) to produce a 128-bit output.

Once p is set, the modulus q and τ can be set such that $\tau \cdot \log q \approx \lambda \cdot \log p$. For instance, to provide 2^λ-bit security, one can reasonable set q to be a 20-bit prime number and $\tau = 12$. Finally, after q is set, the LWE parameter n and noise distribution χ can be set such that the resulting LWE problem provides λ-bit level of security. Following the analysis in [APS15], we can set n to be around 600 (classical security) or 800 (quantum security), and χ to be either a uniform distribution over $[-24, 24]$ or an analogous Gaussian distribution with similar bits of entropy.

5 The Ring Setting

In this section, we extend the results from Section 4 to the ring setting. We first define the Ring-LWRE problem (RLWRE) problem, which is defined analogously to the standard LWRE problem over the ring R_q (Section 3.1).

Definition 5.1 (Ring-LWRE)**.** Let $\lambda \in \mathbb{N}$ be the security parameter. The *ring learning with rounding and errors* (RLWRE) problem is defined with respect to the parameters

- RLWE parameters (n, q, χ),
- Rounding modulus $p \in \mathbb{N}$ such that $p < q$,
- Chaining parameter $\tau \in \mathbb{N}$,

that are defined as functions of λ. Additionally, let $D_\chi : \{0,1\}^{\lfloor n \log p \rfloor} \to R$ be a sampling algorithm for the error distribution χ. Then, we define the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ real and ideal oracles $\mathcal{O}^{(\mathsf{rlwre})}_{\tau,\mathbf{s}}$ and $\mathcal{O}^{(\mathsf{ideal})}$ as follows:

- $\mathcal{O}^{(\mathsf{rlwre})}_{\tau,\mathbf{s}}$: The real oracle is defined with respect to a chaining parameter $\tau \in \mathbb{N}$, and a secret vector of ring elements $\mathbf{s} = (s_1, \ldots, s_\tau) \in R_q^\tau$. On its invocation, the real oracle samples a ring element $a \xleftarrow{\text{R}} R_q$, and initializes $e_1 = 0 \in R$. Then, for $1 \le i < \tau$, it iteratively computes:

  1. $r_i \leftarrow \lfloor a \cdot s_i + e_i \rceil_p$.
  2. $e_{i+1} \leftarrow D(r_i)$.

  It then sets $b = \lfloor a \cdot s_\tau + e_\tau \rceil_p$, and returns $(a, b) \in R_q \times R_q$.

- $\mathcal{O}^{(\mathsf{ideal})}$: On its invocation, the ideal oracle samples a pair of random ring elements $a, b \xleftarrow{\text{R}} R_q$, and returns $(a, b) \in R_q \times R_q$.

The $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem is to distinguish the oracles $\mathcal{O}^{(\mathsf{rlwre})}_{\tau,\mathbf{s}}$ for $\mathbf{s} \xleftarrow{\text{R}} R_q^\tau$. More precisely, we way that an adversary $\mathcal{A}$ solves the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem if there exists a noticeable function $\mu(\cdot)$ such that for $\mathbf{s} \xleftarrow{\text{R}} R_q^\tau$, we have

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{rlwre})}_{\tau,\mathbf{s}}}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{(\mathsf{ideal})}}(1^\lambda) = 1\right] \right| \ge \mu(\lambda).$$

As in the case of the standard $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem, the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem for $\tau = 1$ is equivalent to the standard Ring Learning with Rounding problem [BPR12]. For $\tau > 1$, we can show that the hardness of $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ is implied by the hardness of $\mathsf{RLWE}_{n,q,p,\chi}$ by extending Theorem 4.2 to the ring setting.

Theorem 5.2. *Let λ be the security parameter and n, q, p, χ, τ be a set of parameters for the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem such that $(2Bp/q+1/2^\lambda)^\tau = \mathsf{negl}(\lambda)$ and p divides q. Then, the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem is as hard as the $\mathsf{RLWE}_{n,q,\chi}$ problem.*

Theorem 5.2 follows by exactly the same argument used to prove the hardness of the standard $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ problem (Section A).

Using the same ideas in Section 4.2, we can construct a pseudorandom synthesizer from the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem. Specifically, we can naturally extend Definition 4.7 and Construction 4.8 to the ring setting as follows.

Definition 5.3 (RLWRE Function)**.** Let λ be the security parameter and let n, q, p, χ, τ be a set of RLWRE parameters. Then, we define the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ function $\mathcal{G}^{(\mathsf{rlwre})}_{n,q,p,\chi,\tau} : R_q \times R_q^\tau \to R_p$ as follows:

- $\mathcal{G}^{(\mathsf{rlwre})}_{n,q,p,\chi,\tau}(a, \mathbf{s})$: On input $a \in R_q$ and $\mathbf{s} = (s_1, \ldots, s_\tau) \in R_q^\tau$, the RLWRE function sets $e_1 = 0 \in R$, and computes for $1 \le i < \tau$:

  1. $r_i \leftarrow \lfloor a \cdot s_i + e_i \rceil_p$.
  2. $e_{i+1} \leftarrow D(r_i)$.

  It sets $b = \lfloor a \cdot s_\tau + e_\tau \rceil_p$ and returns $b \in R_p$.

Construction 5.4 (RLWRE Synthesizer)**.** Let λ be the security parameter, let n, q, p, χ, τ be a set of LWRE parameters, and let $\ell = \tau \lceil \log q / \log p \rceil$. We define the RLWRE synthesizer $S : R_q^\ell \times R_q^\ell \to R_q^\ell$ as follows:

- $S(\mathbf{a}, \mathbf{s})$: On input $\mathbf{a}, \mathbf{s} \in R_q^\ell$, the synthesizer parses the vectors
  - $\mathbf{a} = (a_1, \ldots, a_{\ell_1})$ where $a_i \in R_q$ and $\ell_1 = \tau \lceil \log q / \log p \rceil$.
  - $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_{\ell_2})$ where $\mathbf{s}_j \in R_q^\tau$ and $\ell_2 = \lceil \log q / \log p \rceil$.

  Then, the synthesizer computes the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ function $b_{i,j} \leftarrow \mathcal{G}_{n,q,p,\chi,\tau}^{(\mathsf{rlwre})}(a_i, \mathbf{s}_j)$ for all $i \in [\ell_1]$ and $j \in [\ell_2]$. It translates the bits of $\{b_{i,j}\} \in R_p^{\ell_1 \times \ell_2}$ as the representation of a vector $\mathbf{b} \in R_p^\ell$. It returns the vector $\mathbf{b}$.

We can state the security of Construction 5.4 analogously to Theorem 4.9.

Theorem 5.5. *Let λ be the security parameter and let n, q, p, χ, τ be a set of RLWRE parameters. Then, assuming that the $\mathsf{RLWRE}_{n,q,p,\chi,\tau}$ problem is hard, the LWRE synthesizer in Construction 5.4 is a secure pseudorandom synthesizer (Definition 4.5).*

Corollary 5.6. *Let λ be a security parameter and $n = \mathsf{poly}(\lambda)$ be a positive integer. Then, there exists a pseudorandom function $F : \mathcal{K} \times \{0,1\}^{\mathsf{poly}(\lambda)} \to \mathcal{Y}$ that can be computed by a circuit in NC^3 and whose security can be reduced (via a quantum reduction) from the worst-case hardness of approximating SVP to a polynomial approximation factor on n-dimensional ideal lattices.*

6 Key-Homomorphic PRFs

In this section, we show how to use the chaining method to construct key-homomorphic PRFs directly from the Learning with Errors assumption with a polynomial modulus q. Our construction is the modification of the Banerjee-Peikert (BP) PRF of [BP14], which generalizes the algebraic structure of previous LWE-based PRFs [BPR12, BLMR13].

To be precise, the BP PRF is not a single PRF family, but rather multiple PRF families that are each defined with respect to a full binary tree. The LWE parameters that govern the security of a BP PRF are determined by the structure of the corresponding binary tree. In order to construct a key-homomorphic PRF that relies on the hardness of LWE with a polynomial modulus q, we must use the BP PRF that is defined with respect to the "right-spine" tree. However, as our modification works over any BP PRF family, we present our construction with respect to any general BP PRF. For general BP PRFs, our modification is not enough to bring the size of the modulus q to be polynomial in λ, but it still reduces its size by superpolynomial factors.

We provide our main construction in Section 6.1 and discuss its parameters in Section 6.2. We provide the proof of security in Section B and proof of key-homomorphism in Section B.2

6.1 Construction

The BP PRF construction is defined with respect to full (but not necessarily complete) binary trees. Formally, a full binary tree T is a binary tree for which every non-leaf node has two children. The shape of the full binary tree T that is used to define the PRF determines various trade-offs in the parameters and evaluation depth. As we only consider full binary trees in this work, we will implicitly refer to any tree T as a full binary tree.

Throughout the construction description and analysis, we let $|T|$ denote the number of its leaves. For any tree with $|T| \geq 1$, we let $T.\ell$ and $T.r$ denote the left and the right subtrees of T respectively (which may be empty trees). Finally, for a full binary tree T, we define its expansion factor $e(T)$ recursively as follows:

$$e(T) = \begin{cases} 0 & \text{if } |T| = 1 \\ \max\{e(T.\ell) + 1, e(T.r)\} & \text{otherwise.} \end{cases}$$

This is simply the "left-depth" of the tree, i.e., the maximum length of a root-to-leaf path, counting edges from parents to their left children.

With these notations, we define our PRF construction in a sequence of steps. We first define an input-to-matrix mapping $\mathbf{A}_T : \{0,1\}^{|T|} \to \mathbb{Z}_q^{n \times m}$ as follows.

Definition 6.1. Let n, q, χ be a set of LWE parameters, let $p < q$ be a rounding modulus, and let $m = n\lceil \log q \rceil$. Then, for a full binary tree T, and matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, define the function $\mathbf{A}_T : \{0,1\}^{|T|} \to \mathbb{Z}_q^{n \times m}$ recursively:

$$\mathbf{A}_T(x) = \begin{cases} \mathbf{A}_x & \text{if } |T| = 1 \\ \mathbf{A}_{T.\ell}(x_\ell) \cdot \mathbf{G}^{-1}(\mathbf{A}_{T.r}(x_r)) & \text{otherwise,} \end{cases},$$

where $x = x_\ell \| x_r$ for $|x_\ell| = |T.\ell|$, $|x_r| = |T.r|$.

Then, for a binary tree T, and a PRF key $\mathbf{s} \in \mathbb{Z}_q^n$, the BP PRF $F^{(\mathsf{BP})} : \{0,1\}^{|T|} \to \mathbb{Z}_p^m$ is defined as $F_{\mathbf{s}}^{(\mathsf{BP})}(x) = \lfloor \mathbf{s}^T \mathbf{A}_T(x) \rceil_p$. To define our new PRF, we must first define its "noisy" variant $\mathcal{G}_{\mathbf{s},\mathbf{E},T} : \{0,1\}^{|T|} \to \mathbb{Z}_p^m$ as follows.

Definition 6.2. Let n, q, χ be a set of LWE parameters, let $p < q$ be a rounding modulus, and let $m = n\lceil \log q \rceil$. Then, for a full binary tree T, public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, error matrix $\mathbf{E} = (\mathbf{e}_1, \ldots, \mathbf{e}_{e(T)}) \in \mathbb{Z}^{m \times e(T)}$, and a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, we define the function $\mathcal{G}_{\mathbf{s},\mathbf{E},T} : \{0,1\}^{|T|} \to \mathbb{Z}_p^m$:

$$\mathcal{G}_{\mathbf{s},\mathbf{E},T}(x) = \begin{cases} \mathbf{s}^T \mathbf{A}_T(x) + \mathbf{e}_1^T & \text{if } |T| = 1 \\ \mathcal{G}_{\mathbf{s},\mathbf{E},T.\ell}(x_\ell) \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{T.r}(x_r)\big) + \mathbf{e}_{e(T)} & \text{otherwise,} \end{cases}$$

where $x = x_\ell \| x_r$ for $|x_\ell| = |T.\ell|$, $|x_r| = |T.r|$.

We note that when the error matrix $\mathbf{E}$ is set to be an all-zero matrix $\mathbf{E} = \mathbf{0} \in \mathbb{Z}^{m \times e(T)}$, then the function $\lfloor \mathcal{G}_{\mathbf{s},\mathbf{0},T}(\cdot) \rceil_p$ is precisely the BP PRF.

We define our new PRF to be the iterative chaining of the function $\mathcal{G}_{\mathbf{s},\mathbf{E},T}$. Specifically, our PRF is defined with respect to τ secret vectors $\mathbf{s}_1, \ldots, \mathbf{s}_\tau \in \mathbb{Z}_q^n$ where $\tau \in \mathbb{N}$ is the chaining parameter. On input $x \in \{0,1\}^{|T|}$, the PRF evaluation function computes $\mathbf{r}_1 \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_1,\mathbf{0},T}(x) \rceil_p$ and uses $\mathbf{r}_1 \in \mathbb{Z}_p^m$ as a seed to derive the noise term $\mathbf{E}_2$ for the next iteration $\mathbf{r}_2 \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_2,\mathbf{E}_2,T}(x) \rceil_p$. The evaluation function repeats this procedure for $\tau - 1$ iterations and returns $\mathbf{r}_\tau \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_{\tau-1},\mathbf{E}_{\tau-1},T}(x) \rceil_p$ as the final PRF evaluation.

Construction 6.3. Let n, m, q, and χ be LWE parameters and $p < q$ be an additional rounding modulus. Then, our PRF construction is defined with respect to a full binary tree T, two public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, a secret matrix $\mathbf{S} \in \mathbb{Z}_q^{m \times \tau}$, and a sampler $D_\chi : \{0,1\}^{m\lfloor \log p \rfloor} \to \mathbb{Z}^{m \times e(T)}$ for the noise distribution χ. We define our PRF $F_{\mathbf{S}} : \{0,1\}^{|T|} \to \mathbb{Z}_q^m$ as follows:

- $F_{\mathbf{S}}(x)$: On input $x \in \{0,1\}^{|T|}$, the evaluation algorithm sets $\mathbf{E}_1 = \mathbf{0} \in \mathbb{Z}_q^{m \times e(T)}$. Then, for $i = 1, \ldots, \tau - 1$, it iteratively computes

  1. $\mathbf{r}_i \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_i, \mathbf{E}_i, T}(x) \rceil_p$.
  2. $\mathbf{E}_{i+1} \leftarrow D_\chi(\mathbf{r}_i)$.

  It sets $\mathbf{y} \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_\tau, \mathbf{E}_\tau, T}(x) \rceil_p$, and returns $\mathbf{y} \in \mathbb{Z}_p^m$.

For the PRF to be well-defined, we must make sure that a seed $\mathbf{r}_i \leftarrow \lfloor \mathcal{G}_{\mathbf{s}_i, \mathbf{E}_i, T}(x) \rceil_p \in \mathbb{Z}_p^m$ for $i \in [\tau - 1]$ provides enough entropy to derive the noise terms $\mathbf{E}_{i+1} \leftarrow \chi^{m \times e(T)}$. As in the case of LWRE (Remark 4.3), there are two main ways of ensuring this condition. One method is to set the rounding modulus p to be big enough such that there exists a sampler $D_\chi : \{0,1\}^{m \lfloor \log p \rfloor} \to \mathbb{Z}^{m \times e(T)}$ for the noise distribution $\chi^{m \times e(T)}$. Alternatively, one can expand the seed $\mathbf{r}_i \in \mathbb{Z}_p^m$ using a pseudorandom generator to derive sufficiently many bits to sample from $\chi^{m \times e(T)}$. Since the issue of deriving the noise terms $\mathbf{E}_{i+1}$ from the seeds $\mathbf{r}_i$ is mostly orthogonal to the central ideas of our PRF construction, we assume that the rounding modulus p is set to be big enough such that the noise terms $\mathbf{E}_{i+1}$ can be derived from the bits of $\mathbf{r}_i$.

We now state the main security theorem for the PRF in Construction 6.3.

Theorem 6.4. *Let T be any full binary tree, λ the security parameter, n, m, q, p, τ positive integers and χ a B-bounded distribution such that $m = n\lceil \log q \rceil > 2$, $(2Rmp/q)^{\tau-1} = \mathsf{negl}(\lambda)$ for $R = |T|Bm^{e(T)}$ and p divides q. Then, assuming that the $\mathsf{LWE}_{n,q,\chi}$ problem is hard, the PRF in Construction 6.3 is a 2-almost key-homomorphic PRF (Definition 3.6).*

We provide the proof of Theorem 6.4 in Section B. Except for the components on chaining, the proof inherits many of the arguments that are already used in [BP14]. For the main intuition behind the proof, we refer the readers to Section 2.3.

6.2 Instantiating the Parameters.

As in the original Banerjee-Peikert PRF [BP14], the size of the modulus q and the depth of the evaluation circuit is determined by the structure of the full binary tree T. The size of the modulus q is determined by the expansion factor $e(T)$ or equivalently, the length of the maximum root-to-leaf path to the left children of T. Namely, to satisfy Theorem 6.4, we require q to be large enough such that $(2Rmp/q)^{\tau-1} = \mathsf{negl}(\lambda)$ for $R = |T|Bm^{e(T)}$.

The depth of the evaluation circuit is determined by the *sequentiality* factor $s(T)$ of the tree, which is formally defined by the recurrence

$$s(T) = \begin{cases} 0 & \text{if } |T| = 1 \\ \max\{s(T.\ell), s(T.r) + 1\} & \text{otherwise.} \end{cases}$$

Combinatorially, $s(T)$ denotes the length of the maximum root-to-leaf path to the right children of T. For a tree T, our PRF can be evaluated by depth $\tau \cdot s(T) \log |T|$.

When we restrict the chaining parameter to be $\tau = 1$, then we recover the parameters of the Banerjee-Peikert PRF where the modulus q is required to be $q = Rmp\lambda^{\omega(1)}$. However, setting τ to be a super-constant function $\omega(1)$, we can set $q = 2Rmp \cdot \lambda^c$ for any constant $c > 0$, which reduces the size of the modulus q by a super-polynomial factor. To guarantee that an adversary's advantage in breaking the PRF degrades exponentially in λ, we can set τ to be linear in λ.

To set q to be polynomial in the security parameter, we can instantiate the construction with respect to the "right-spine" binary tree where the left child of any node in the tree is a leaf node. In this tree T, the sequentiality factor becomes linear in the size of the tree $s(T) = |T|$, but the expansion factor becomes a constant $e(T) = 1$. Therefore, when $\tau = \omega(1)$ (or linear in λ for 2^λ-security), the modulus q can be set to be $q = 2mp\lambda^c$ for any constant $c > 0$. The concrete parameters for our PRF can be set similarly to our synthesizer construction (see Section 4.3).

Acknowledgments

We thank the Eurocrypt reviewers for their helpful comments. This work was funded by NSF, DARPA, a grant from ONR, and the Simons Foundation. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[AA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from lwe to lwr. *IACR Cryptology ePrint Archive*, 2016(589), 2016.

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *CRYPTO*, 2010.

[ABP15a] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. An algebraic framework for pseudorandom functions and applications to related-key security. In *CRYPTO*, 2015.

[ABP15b] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. Multilinear and aggregate pseudorandom functions: New constructions and improved security. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 103–120. Springer, 2015.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, 2009.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In *CRYPTO*. 2013.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, 2014.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[BBPS18] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. Order-lwe and the hardness of ring-lwe with entropic secrets. Cryptology ePrint Archive, Report 2018/494, 2018. https://eprint.iacr.org/2018/494.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, 1996.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *FOCS*, 1996.

[BDGJ19] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Rise and shine: Fast and secure updatable encryption. Cryptology ePrint Archive, Report 2019/1457, 2019. https://eprint.iacr.org/2019/1457.

[Bel06] Mihir Bellare. New proofs for nmac and hmac: Security without collision-resistance. In *CRYPTO*, 2006.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC*, 2016.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[BKM17] Dan Boneh, Sam Kim, and Hart Montgomery. Private puncturable PRFs from standard lattice assumptions. In *EUROCRYPT*, 2017.

[BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*. 2013.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *CCS*, 2010.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, 2012.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from lwe. In *TCC*, 2017.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *ITCS*, 2014.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, 2015.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for NC^1 from LWE. In *EUROCRYPT*, 2017.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

[DS15] Nico Döttling and Dominique Schröder. Efficient pseudorandom functions via on-the-fly adaptation. In *CRYPTO*, 2015.

[EPRS17] Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key rotation for authenticated encryption. In *CRYPTO*, 2017.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient abe for branching programs. In *ASIACRYPT*, 2015.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[JKP18] Tibor Jager, Rafael Kurek, and Jiaxin Pan. Simple and more efficient prfs with tight security from lwe and matrix-ddh. In *ASIACRYPT*, 2018.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (r) cca secure updatable encryption with integrity protection. In *EUROCRYPT*, 2019.

[KW17] Sam Kim and David J Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO*, 2017.

[KW19] Sam Kim and David J Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In *CRYPTO*, 2019.

[Lan00] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*, volume 1. Ripol Classic Publishing House, 2000.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, 2013.

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

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In *EUROCRYPT*, 2018.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, 2011.

[Mon18] Hart Montgomery. More efficient lattice prfs from keyed pseudorandom synthesizers. In *INDOCRYPT*, 2018.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *EUROCRYPT*, 1999.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. In *STOC*, 2017.

[QWZ18] Willy Quach, Daniel Wichs, and Giorgos Zirdelis. Watermarking prfs under standard assumptions: Public marking and security with extraction queries. In *TCC*, 2018.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[RSSS17] Miruna Roşca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In *CRYPTO*, 2017.

[RSW18] Miruna Roşca, Damien Stehlé, and Alexandre Wallet. On the ring-lwe and polynomial-lwe problems. In *EUROCRYPT*, 2018.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.

A Proof of Theorem 4.2

We proceed via a hybrid argument. To do so, we first define a hybrid oracle $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ corresponding to our intermediate hybrid experiment. The oracle $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ is defined identically as the real $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ except that $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ samples $e_1 \leftarrow \chi$ rather than setting it to be initially 0. Formally, for $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_\tau) \in \mathbb{Z}_q^{n\times\tau}$, the oracle $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ is defined as follows:

- $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$: On its invocation, the hybrid oracle first samples a random vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ *and* an error vector $e_1 \leftarrow \chi$. Then, for $i = 1, \ldots, \tau - 1$, it iteratively computes:

  1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
  2. $e_{i+1} \leftarrow D_\chi(r_i)$.

  Then, it sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.

With the oracles $\mathcal{O}^{(\mathsf{lwre})}_{\mathbf{S}}$, $\mathcal{O}^{(\mathsf{hyb})}_{\mathbf{S}}$, and $\mathcal{O}^{(\mathsf{ideal})}$, we define our hybrid experiments as follows:

- HYB_0: This is the real $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ experiment where the adversary $\mathcal{A}$ is interacting with the real oracle $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ for $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$ as defined in Definition 4.1. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta \in \{0,1\}$, which is also the output of the experiment.

- HYB_1: This is the intermediate security experiment where the adversary $\mathcal{A}$ is interacting with an intermediate oracle $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ for $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$ as defined above. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta \in \{0,1\}$, which is also the output of the experiment.

- HYB_2: This is the ideal $\mathsf{LWRE}_{n,q,p,\chi,\tau}$ experiment where the adversary $\mathcal{A}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$ as defined in Definition 4.1. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta \in \{0,1\}$, which is also the output of the experiment.

We now bound an adversary's distinguishing advantage of the consecutive hybrid experiment. For an experiment HYB_i and an adversary $\mathcal{A}$, we define $\text{HYB}_i(\mathcal{A})$ to denote the random variable representing the output of experiment HYB_i with respect to the adversary $\mathcal{A}$.

Lemma A.1. *For any (unbounded) adversary $\mathcal{A}$ making at most a Q number of oracle calls, we have*

$$\big|\Pr[\text{HYB}_0(\mathcal{A}) = 1] - \Pr[\text{HYB}_1(\mathcal{A}) = 1]\big| \leq Q\big(2Bp/q + 1/2^\lambda\big)^\tau.$$

Proof. To prove the lemma, we show that for an overwhelming fraction of the vectors $\mathbf{a}$ in $\mathbb{Z}_q^n$, the oracles $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ return the exact same output with overwhelming probability over the matrix $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$. Since each of these two oracles generates the vectors $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ uniformly at random, this shows that an adversary $\mathcal{A}$ that invokes the oracles $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ at most a polynomial number of times will not have enough information to distinguish the two oracles with overwhelming probability.

To analyze the behavior of $\mathcal{O}^{(\mathsf{lwre})}_{\tau,\mathbf{S}}$ and $\mathcal{O}^{(\mathsf{hyb})}_{\tau,\mathbf{S}}$ for a vector $\mathbf{a} \in \mathbb{Z}_q^n$, we define the following parameterized (oracle) algorithm:

- $\mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})$: The oracle is defined with respect to a vector $\mathbf{a} \in \mathbb{Z}_q^n$ and an error term $e_1 \in [-B, B]$. It takes in a matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times\tau}$, and computes for $i = 1, \ldots, \tau - 1$:

1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
2. $e_{i+1} \leftarrow D_\chi(r_i)$.

It then sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.

By definition, for $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $e_1 \leftarrow \chi$, and $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$, the output $(\mathbf{a}, b) \leftarrow \mathcal{O}_{\mathbf{a},0}(\mathbf{S})$ is identically distributed as a single $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{lwre})}$ sample, and the output $(\mathbf{a}, b') \leftarrow \mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})$ is identically distributed as a single $\mathcal{O}_{\tau,\mathbf{S}}^{(\mathsf{ideal})}$ sample. We show that for a random vector $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$, and *any* error value $e_1 \in [-B, B]$, we have

$$\Pr[\mathcal{O}_{\mathbf{a},0}(\mathbf{S}) \neq \mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})] = (2Bp/q)^\tau + 2^{-\lambda},$$

To do this, we consider the transcript of the executions of the two algorithms $\mathcal{O}_{\mathbf{a},0}(\mathbf{S})$ and $\mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})$. Let $r_1^{(0)}, \ldots, r_{\tau-1}^{(0)}, r_\tau^{(0)} = b^{(0)}$ be the random coins that are set during the execution of $\mathcal{O}_{\mathbf{a},0}(\mathbf{S})$, and let $r_1^{(1)}, \ldots r_{\tau-1}^{(1)}, r_\tau^{(1)} = b^{(1)}$ be the random coins set during the execution of $\mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})$. Then, for the of analysis, define the following random variables

$$X_i = \begin{cases} 1 & \text{if } r_i^{(0)} \neq r_i^{(1)} \\ 0 & \text{otherwise.} \end{cases}$$

By definition, we have

$$\Pr[\mathcal{O}_{\mathbf{a},0}(\mathbf{S}) \neq \mathcal{O}_{\mathbf{a},e_1}(\mathbf{S})] \leq \Pr[X_\tau = 1],$$

and our goal is to bound the probability $\Pr[X_\tau = 1]$ over a random matrix $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau}$.

We first note that for any $i \in [\tau - 1]$, the condition $r_i^{(0)} = r_i^{(1)}$ implies that $r_{i+1}^{(0)} = r_{i+1}^{(1)}$ since the sampler D_χ is deterministic. This means that for $i \in [\tau - 1]$, we have

$$\Pr[X_{i+1} = 1 | X_i = 0] = 0,$$

and therefore,

$$\begin{aligned} \Pr[X_{i+1} = 1] &= \Pr[X_{i+1} = 1 | X_i = 1] \cdot \Pr[X_i = 1] + \Pr[X_{i+1} = 1 | X_i = 0] \cdot \Pr[X_i = 0] \\ &= \Pr[X_{i+1} = 1 | X_i = 1] \cdot \Pr[X_i = 1]. \end{aligned}$$

Now, the probability that $X_\tau = 1$ can be represented as the product of the conditional probabilities

$$\Pr[X_\tau = 1] = \prod_{i\in[\tau-1]} \Pr[X_{i+1} = 1 | X_i = 1]. \tag{A.1}$$

For $i \in [\tau - 1]$, let us consider the probability that $r_{i+1}^{(0)} \neq r_{i+1}^{(1)}$ given that $r_i^{(0)} \neq r_i^{(1)}$. As the error terms $e_{i+1}^{(0)} \leftarrow D_\chi(r_i^{(0)})$ and $e_{i+1}^{(1)} \leftarrow D_\chi(r_i^{(1)})$ are B-bounded, the probability that

$$r_{i+1}^{(0)} = \lfloor \langle \mathbf{a}, \mathbf{s}_{i+1} \rangle + e_{i+1}^{(0)} \rceil_p \neq \lfloor \langle \mathbf{a}, \mathbf{s}_{i+1} \rangle + e_{i+1}^{(1)} \rceil_p = r_{i+1}^{(1)},$$

is at most the probability that the inner product $\langle \mathbf{a}, \mathbf{s}_i \rangle$ is in the "boundary" set

$$\langle \mathbf{a}, \mathbf{s}_i \rangle \in [-B, B] + \frac{q}{p} \cdot \left(\mathbb{Z} + \frac{1}{2} \right) = \left\{ \frac{q}{p} \cdot \mathbb{Z} + \left\lfloor \frac{q}{2p} \right\rceil + [-B, B] \right\},$$

which means that

$$\Pr[X_{i+1} = 1 | X_i = 1] \leq \Pr\big[\, \langle \mathbf{a}, \mathbf{s}_i \rangle \in [-B, B] + q/p \cdot (\mathbb{Z} + 1/2)\big]. \tag{A.2}$$

Finally, using Lemma 3.4, we can bound the probability

$$\Pr\big[\, \langle \mathbf{a}, \mathbf{s}_i \rangle \in [-B, B] + q/p \cdot (\mathbb{Z} + 1/2)\big] \leq (2Bp/q) + 2^{-\lambda}. \tag{A.3}$$

Combining the relations (A.1), (A.2), and (A.2), we have

$$\Pr[X_\tau = 1] \leq \left(\frac{2Bp}{q} + \frac{1}{2^\lambda}\right)^\tau,$$

and the lemma now follows by a union bound over the number of queries that $\mathcal{A}$ makes.

□

Lemma A.2. *For any efficient adversary $\mathcal{A}$, we have*

$$\big|\Pr[\textsc{hyb}_1(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1]\big| \leq \tau \cdot \mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}).$$

Proof. To prove the lemma, we define a sequence of intermediate hybrid experiments $\textsc{hyb}_{1,j}$ for $j \in [\tau]$:

- $\textsc{hyb}_{1,j}$: In this security experiment, the adversary $\mathcal{A}$ is given access to the oracle $\mathcal{O}^{(j)}_{\tau,\mathbf{S}}$ that works as follows:
  - $\mathcal{O}^{(j)}_{\tau,\mathbf{S}}$: On its invocation, the oracle samples an error term $e_j \leftarrow \chi$. Then, for $i \in [j, \tau - 1]$, it iteratively computes:
    1. $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_i \rangle + e_i \rceil_p$.
    2. $e_{i+1} \leftarrow D_\chi(r_i)$.

    Then, it sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.

  At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta \in \{0, 1\}$, which is the output of the experiment.

By definition, the hybrid experiment $\textsc{hyb}_{1,1}$ corresponds to the experiment $\textsc{hyb}_1$. We show that each consecutive hybrid experiments $\textsc{hyb}_{1,i}$ and $\textsc{hyb}_{1,i+1}$ for $i = 1, \ldots, \tau$ are computationally indistinguishable. Then, we show that the experiments $\textsc{hyb}_{i,\tau}$ and $\textsc{hyb}_2$ are computationally indistinguishable.

Lemma A.3. *For any efficient adversary $\mathcal{A}$, we have*

$$\big|\Pr[\textsc{hyb}_{1,i}(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_{1,i+1}(\mathcal{A}) = 1]\big| \leq \mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}),$$

for all $i \in [\tau - 1]$.

Proof. Let $\mathcal{A}$ be a distinguisher for the experiments $\textsc{hyb}_{1,j}$ and $\textsc{hyb}_{1,j+1}$. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the $\mathsf{LWE}_{n,q,\chi}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

1. At the start of the experiment, algorithm $\mathcal{B}$ samples secret vectors $\mathbf{s}_{j+1}, \ldots, \mathbf{s}_\tau \xleftarrow{\text{R}} \mathbb{Z}_q^n$.

2. Whenever $\mathcal{A}$ invokes the oracle (one of $\mathcal{O}_{\tau,\mathbf{S}}^{(j)}$ or $\mathcal{O}_{\tau,\mathbf{S}}^{(j+1)}$), algorithm $\mathcal{B}$ first invokes its own oracle (one of $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$) to receive $(\mathbf{a}, \hat{b}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Then, it rounds $r_i \leftarrow \lfloor \hat{b} \rceil_p$, samples $e_{i+1} \leftarrow D_\chi(r_i)$, and for $i = j+1, \ldots, \tau - 1$, iteratively computes:
   (a) $r_i \leftarrow \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_i \rceil_p$.
   (b) $e_{i+1} \leftarrow D_\chi(r_i)$.

   It sets $b = \lfloor \langle \mathbf{a}, \mathbf{s}_\tau \rangle + e_\tau \rceil_p$, and returns $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ to $\mathcal{A}$.
3. At the end of the experiment, when $\mathcal{A}$ returns its guess $\beta \in \{0, 1\}$, algorithm $\mathcal{B}$ returns β.

We now show that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates either $\mathcal{O}_{\tau,\mathbf{S}}^{(j)}$ or $\mathcal{O}_{\tau,\mathbf{S}}^{(j+1)}$.

- Suppose that $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ for $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$. Then, for each of $\mathcal{A}$'s queries, algorithm $\mathcal{B}$ receives an LWE sample $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ for $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Setting $\mathbf{s}_j = \mathbf{s}$ and $e_j = e$, the response of $\mathcal{B}$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in $\mathcal{O}_{\tau,\mathbf{S}}^{(j)}$ by definition.

- Suppose that $\mathcal{B}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$. Then, for each of $\mathcal{A}$'s queries, algorithm $\mathcal{B}$ receives a uniform sample $(\mathbf{a}, \hat{b}) \xleftarrow{\text{R}} \mathbb{Z}_q^n \times \mathbb{Z}_q$. As $\hat{b}$ is uniform in $\mathbb{Z}_q$ and q is divisible by p, the randomness $r_j = \lfloor \hat{b} \rceil_p$ is also uniformly generated. Therefore, by definition of the sampler $D_\chi(r_j)$, the error term $e_{j+1} \leftarrow D_\chi(r_j)$ is distributed exactly as in $e_{j+1} \leftarrow \chi$, and $\mathcal{B}$ perfectly simulates the distribution of $\mathcal{O}_{\tau,\mathbf{S}}^{(j+1)}$.

We have shown that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates the distributions of either $\textsc{hyb}_{1,j}$ or $\textsc{hyb}_{1,j+1}$. Therefore, with the same distinguishing advantage of $\mathcal{A}$, algorithm $\mathcal{B}$ solves the $\mathsf{LWE}_{n,q,\chi}$ problem. □

Lemma A.4. *For any efficient adversary $\mathcal{A}$, we have*

$$\left| \Pr[\textsc{hyb}_{1,\tau}(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1] \right| = \mathsf{Adv}_{\mathsf{LWE}}(n, q, \chi, \mathcal{A}).$$

Proof. The lemma follows via a similar argument to the proof of Lemma A.3. Let $\mathcal{A}$ be a distinguisher for the experiments $\textsc{hyb}_{1,\tau}$ and $\textsc{hyb}_2$. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the $\mathsf{LWE}_{n,q,\chi}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

1. Whenever $\mathcal{A}$ invokes the oracle (one of $\mathcal{O}_{\tau,\mathbf{S}}^{(\tau)}$ or $\mathcal{O}^{(\mathsf{ideal})}$), algorithm $\mathcal{B}$ first invokes its own oracle (one of $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$) to receive $(\mathbf{a}, \hat{b}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. It provides $(\mathbf{a}, \lfloor \hat{b} \rceil_p)$ to $\mathcal{A}$.

2. At the end of the experiment, when $\mathcal{A}$ returns its guess $\beta \in \{0, 1\}$, algorithm $\mathcal{B}$ returns β.

We now show that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates either $\mathcal{O}_{\mathbf{S}}^{(\tau)}$ or $\mathcal{O}^{(\mathsf{ideal})}$.

- Suppose that $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ for $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$. Then, for each of $\mathcal{A}$'s queries, algorithm $\mathcal{B}$ receives an LWE sample $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ for $\mathbf{a} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Setting $\mathbf{s}_\tau = \mathbf{s}$ and $e_\tau = e$, the response of $\mathcal{B}$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in $\mathcal{O}_{\tau,\mathbf{S}}^{(\tau)}$ by definition.

- Suppose that $\mathcal{B}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$. Then, for each of $\mathcal{A}$'s queries, algorithm $\mathcal{B}$ receives a uniform sample $(\mathbf{a}, \hat{b}) \xleftarrow{\text{R}} \mathbb{Z}_q^n \times \mathbb{Z}_q$. As $\hat{b}$ is uniform in $\mathbb{Z}_q$ and q is divisible by p, the sample $(\mathbf{a}, \lfloor \hat{b} \rceil)$ is uniform in $\mathbb{Z}_q^n \times \mathbb{Z}_p$. Therefore, by definition, $\mathcal{B}$'s response for each of $\mathcal{A}$'s oracle queries is distributed exactly as in $\mathcal{O}^{(\mathsf{ideal})}$.

We have shown that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates the distributions of either $\text{HYB}_{1,\tau}$ or HYB_2. Therefore, with the same distinguishing advantage of $\mathcal{A}$, algorithm $\mathcal{B}$ solves the $\mathsf{LWE}_{n,q,\chi}$ problem. It follows that assuming the hardness of the $\mathsf{LWE}_{n,q,\chi}$ problem, the distinguishing advantage of $\mathcal{A}$ is negligible.

□

This conclude the proof of Lemma A.2. □

B Proof of Theorem 6.4

In this section, we provide the proof of Theorem 6.4. We start by providing additional definitions that will simplify our analysis of security and key-homomorphism in Section B.1. Then, we provide the proof of key-homomorphism in Section B.2 and the proof of security in Section B.3.

B.1 Additional Definitions

We first define a combinatorial operation called *pruning*.

Definition B.1 (Pruning)**.** For a full binary tree T of at least one node, we define its *pruning* $T' = \mathsf{Prune}(T)$ inductively as follows:

- If $|T.\ell| \leq 1$, then set $T' = T.r$.
- Otherwise, set $T'.\ell = \mathsf{Prune}(T.\ell)$ and $T'.r = T.r$.

We let $T^{(i)}$ denote the i^{th} successive pruning of T, i.e., $T^{(0)} = T$ and $T^{(i)} = \mathsf{Prune}(T^{(i-1)})$.

In words, the pruning operation Prune takes in a binary tree, removes its leftmost leaf v, and replaces the subtree rooted at v's parent (if it exists) with the sub-tree rooted at v's sibling. By definition, for any $|T| \geq 1$, we have $|T^{(i)}| = |T| - i$. Furthermore, the pruning of a tree cannot increase a tree's expansion factor, i.e. $e(T') \leq e(T)$.

Next, we recall two functions $\mathbf{S}_T$ and $\mathcal{E}_T$ that characterize the unraveling of the recursive relations in Definitions 6.1 and 6.2.

Definition B.2. For a full binary tree T of at least one node and two matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, define the function $\mathbf{S}_T : \{0,1\}^{|T|-1} \to \mathbb{Z}^{m \times m}$ recursively as follows:

$$\mathbf{S}_T(x) = \begin{cases} \mathbf{I} & \text{if } |T| = 1 \\ \mathbf{S}_{T.\ell}(x_\ell) \cdot \mathbf{G}^{-1}(\mathbf{A}_{T.r}(x_r)) & \text{otherwise,} \end{cases} .$$

where $x = x_\ell \| x_r$ for $|x_\ell| = |T.\ell| - 1$, $|x_r| = |T.r|$.

Definition B.3. For a full binary tree T of at least one node and two matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, define the (randomized) function $\mathcal{E}_T : \{0,1\}^{|T|} \to \mathbb{Z}^m$ inductively as follows:

- For $|T| = 0$, set $\mathcal{E}_T(\varepsilon) = \mathbf{0}$.
- For $|T| \geq 1$, parse $x = x_1 \| x' \in \{0,1\} \times \{0,1\}^{|T|-1}$, sample $\mathbf{e}_{x_1} \leftarrow \chi^m$, and set

$$\mathcal{E}_T(x) = \mathbf{e}_{x_1} \cdot \mathbf{S}_T(x') + \mathcal{E}_{T'}(x'),$$

  where T' is the pruning of T.

It follows by definition that for $x = x_1 \| x' \in \{0,1\} \times \{0,1\}^{|T|-1}$, we have

$$\begin{aligned} \mathbf{A}_T(x) &= \mathbf{A}_{x_1} \cdot \mathbf{S}_T(x'), \\ \mathbf{G} \cdot \mathbf{S}_T(x') &= \mathbf{A}_{T'}(x'). \end{aligned}$$

Furthermore, for any secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, and error matrix $\mathbf{E} \leftarrow \chi^{m \times \tau}$, the following distributions are identical by definition

$$\mathcal{G}_{\mathbf{s},\mathbf{E},T}(x) \equiv \mathbf{s}^\intercal \mathbf{A}_T(x) + \mathcal{E}_T(x).$$

Finally, before proceeding with the proofs, we bound the norm of the function $\mathcal{E}_T(x)$ in the following lemma.

Lemma B.4. *Let n, q, χ be LWE parameters where χ is a B-bounded distribution. Then, for any full binary tree T, matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, and input $x \in \{0,1\}^{|T|}$, we have*

$$\Pr\left[|\mathcal{E}_T(x)| \leq |T| \cdot B \cdot m^{e(T)}\right] = 1,$$

where the probability is taken over the random coins used by the function $\mathcal{E}_T(\cdot)$.

Proof. The lemma follows by induction on $|T|$.

- When $|T| = 0$, we have $\mathcal{E}_T(x) = 0$ by definition.
- When $|T| \geq 1$, then for $x = x_1 \| x' \in \{0,1\}^{|T|}$, $T' = \mathsf{Prune}(T)$, and $\mathbf{e}_{x_1} \leftarrow \chi^m$, the function $\mathcal{E}_T(x)$ is defined as

$$\mathcal{E}_T(x) = \mathbf{e}_{x_1}^\intercal \cdot \mathbf{S}_T(x') + \mathcal{E}_{T'}(x').$$

  By induction, we have $\|\mathcal{E}_{T'}(x')\| \leq |T'| \cdot B \cdot m^{e(T')} \leq (|T| - 1) \cdot B \cdot m^{e(T)}$. Furthermore, by unrolling the recursion in Definition B.2, the output $\mathbf{S}_T(x')$ is the product of at most $e(T)$ binary matrices $\mathbf{G}^{-1}(\cdot)$. Therefore, we have $\|\mathbf{S}_T(x')\| \leq m^{e(T)}$ and since χ is a B-bounded distribution, we have $\|\mathbf{e}_{x_1}\| \leq B$. The triangle inequality gives

$$\left\|\mathbf{e}_{x_1}^\intercal \cdot \mathbf{S}_T(x') + \mathcal{E}_{T'}(x')\right\| \leq B \cdot m^{e(T)} + (T - 1) \cdot B \cdot m^{e(T)}.$$

$\square$

B.2 Proof of Key-Homomorphism

Fix a full binary tree T, public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$, PRF input $x \in \{0,1\}^{|T|}$, and secret matrices $\mathbf{S}, \tilde{\mathbf{S}} \in \mathbb{Z}_q^{n \times \tau}$. Then by (almost) linearity of rounding, there exists a vector $\boldsymbol{\eta}_1 \in \{0,1\}^m$ such that

$$F_{\mathbf{S}}(x) + F_{\tilde{\mathbf{S}}}(x) = \lfloor \mathcal{G}_{\mathbf{s}_\tau, \mathbf{E}_\tau, T}(x) \rceil_p + \lfloor \mathcal{G}_{\tilde{\mathbf{s}}_\tau, \tilde{\mathbf{E}}_\tau, T}(x) \rceil_p = \lfloor \mathcal{G}_{\mathbf{s}_\tau, \mathbf{E}_\tau, T}(x) + \mathcal{G}_{\tilde{\mathbf{s}}_\tau, \tilde{\mathbf{E}}_\tau, T}(x) \rceil_p + \boldsymbol{\eta}_1 \tag{B.1}$$

where the matrices $\mathbf{E}_\tau$ and $\tilde{\mathbf{E}}_\tau$ are the output of the sampler $\mathbf{E}_\tau \leftarrow D_\chi(\mathbf{r}_{\tau-1})$ and $\tilde{\mathbf{E}}_\tau \leftarrow D_\chi(\tilde{\mathbf{r}}_{\tau-1})$. Then by Lemma B.4, the values $\mathcal{G}_{\mathbf{s},\mathbf{E},T}(x)$ and $\mathcal{G}_{\tilde{\mathbf{s}},\tilde{\mathbf{E}},T}(x)$ above can each be written as

$$\mathcal{G}_{\mathbf{s}, \mathbf{E}_\tau, T}(x) = \mathbf{s}^\intercal \mathbf{A}_T(x) + \mathbf{e}_T^\intercal$$

$$\mathcal{G}_{\tilde{\mathbf{s}}, \tilde{\mathbf{E}}_\tau, T}(x) = \tilde{\mathbf{s}}^\intercal \mathbf{A}_T(x) + \tilde{\mathbf{e}}_T^\intercal,$$

for some error vectors $\mathbf{e}_T, \widetilde{\mathbf{e}}_T \in [-R, R]^m$ where $R = |T| B m^{e(T)}$. Therefore, their sum can be represented as

$$\begin{aligned} \lfloor \mathcal{G}_{\mathbf{s}, \mathbf{E}_\tau, T}(x) + \mathcal{G}_{\tilde{\mathbf{s}}, \tilde{\mathbf{E}}_\tau, T}(x) \rceil_p &= \lfloor (\mathbf{s}^\intercal \mathbf{A}_T(x) + \mathbf{e}_T^\intercal) + (\tilde{\mathbf{s}}^\intercal \mathbf{A}_T(x) + \tilde{\mathbf{e}}_T^\intercal) \rceil_p \\ &= \lfloor (\mathbf{s} + \tilde{\mathbf{s}})^\intercal \mathbf{A}_T(x) + (\mathbf{e}_T + \widetilde{\mathbf{e}}_T)^\intercal \rceil_p, \end{aligned}$$

Now, by definition, $F_{\mathbf{S}+\tilde{\mathbf{S}}}(x)$ can also be represented with respect to the matrix $\mathbf{A}_T(x)$ and some error vector $\hat{\mathbf{e}} \in [-R, R]^m$ as

$$\begin{aligned} F_{\mathbf{S}+\tilde{\mathbf{S}}}(x) &= \lfloor \mathcal{G}_{\mathbf{s}+\tilde{\mathbf{s}}, \hat{\mathbf{E}}_\tau, T}(x) \rceil_p \\ &= \lfloor (\mathbf{s} + \tilde{\mathbf{s}})^\intercal \mathbf{A}_T(x) + \hat{\mathbf{e}}_T^\intercal \rceil_p. \end{aligned}$$

Therefore, we have the relation

$$\begin{aligned} \lfloor \mathcal{G}_{\mathbf{s}, \mathbf{E}_\tau, T}(x) + \mathcal{G}_{\tilde{\mathbf{s}}, \tilde{\mathbf{E}}_\tau, T}(x) \rceil_p &= \lfloor (\mathbf{s}^\intercal \mathbf{A}_T(x) + \mathbf{e}_T^\intercal) + (\tilde{\mathbf{s}}^\intercal \mathbf{A}_T(x) + \tilde{\mathbf{e}}_T^\intercal) \rceil_p \\ &= \lfloor F_{\mathbf{S}+\tilde{\mathbf{S}}}(x) + (\mathbf{e}_T + \widetilde{\mathbf{e}}_T - \hat{\mathbf{e}}_T)^\intercal \rceil_p. \end{aligned}$$

Since $\|\mathbf{e}_T + \widetilde{\mathbf{e}}_T - \hat{\mathbf{e}}_T\| \le 3R$ and $(2Rmp/q)^{\tau-1} = \mathsf{negl}(\lambda)$, which implies that $2R < q/mp$, we have

$$\lfloor \mathcal{G}_{\mathbf{s}, \mathbf{E}_\tau, T}(x) + \mathcal{G}_{\tilde{\mathbf{s}}, \tilde{\mathbf{E}}_\tau, T}(x) \rceil_p = F_{\mathbf{S}+\tilde{\mathbf{S}}}(x) + \boldsymbol{\eta}_2, \tag{B.2}$$

for some vector $\boldsymbol{\eta}_2 \in \{0,1\}^m$. Now, combining (B.1) and (B.2), the 2-almost key-homomorphism of Construction 6.3 follows.

B.3 Proof of Security

Hybrid PRF. To prove security, we first define a hybrid (randomized) PRF function $\tilde{F}_{\mathbf{S}}(\cdot)$ for our intermediate hybrid experiment. The function $\tilde{F}_{\mathbf{S}}(\cdot)$ is defined identically as the real PRF $F_{\mathbf{S}}(\cdot)$ except that it samples the initial error term $\mathbf{E}_1$ from the noise distribution $\mathbf{E}_1 \leftarrow \chi^{m \times \ell}$ instead of setting it to be the all-zero matrix $\mathbf{E}_1 = \mathbf{0}$. Additionally, before returning the final output $\mathcal{G}_{\mathbf{s}_\tau, \mathbf{E}_\tau, T}(x)$, the function $\tilde{F}_{\mathbf{S}}(\cdot)$ checks an extra abort condition and returns $\perp$ if the condition is satisfied. Namely, at the end of the execution, it verifies whether any of the iterations of $\mathcal{G}_{\mathbf{s}_1,\mathbf{E}_1,T}(x),\ldots,\mathcal{G}_{\mathbf{s}_\tau,\mathbf{E}_\tau,T}(x)$ resulted in a "borderline" vector in the set

$$\mathsf{Borderline}_R = \left\{\mathbf{y}\in\mathbb{Z}_q^m : \exists\eta\in[m] \text{ s.t. } y_\eta\in[-R,R]+\frac{q}{p}\cdot\left(\mathbb{Z}+\frac{1}{2}\right)\right\}.$$

Intuitively, the set $\mathsf{Borderline}_R$ captures the set of vectors $\mathbf{y}\in\mathbb{Z}_q^m$ that can potentially be rounded to a different value when noise terms $\mathsf{noise}\in[-R,R]^m$ are added prior to rounding $\mathbf{y}$

$$\lfloor\mathbf{y}\rceil_p \neq \lfloor\mathbf{y}+\mathsf{noise}\rceil_p.$$

Formally, we define the hybrid (randomized) PRF $\tilde{F}_{\mathbf{S}}:\{0,1\}^{|T|}\to\mathbb{Z}_q^m$ with respect to a full binary tree T, two public matrices $\mathbf{A}_0,\mathbf{A}_1\in\mathbb{Z}_q^{n\times m}$, and a secret matrix $\mathbf{S}\in\mathbb{Z}_q^{m\times\tau}$ as follows:

$\tilde{F}_{\mathbf{S}}(x)$: On input $x\in\{0,1\}^{|T|}$, the hybrid evaluation algorithm samples $\mathbf{e}_1,\ldots,\mathbf{e}_{e(T)}\leftarrow\chi^m$ and sets $\mathbf{E}_1=(\mathbf{e}_1,\ldots,\mathbf{e}_{e(T)})$. Then, for $1\le i<\tau$, it iteratively computes

1. $\mathbf{y}_i\leftarrow\mathcal{G}_{\mathbf{s}_i,\mathbf{E}_i,T}(x)$.
2. $\mathbf{r}_i\leftarrow\lfloor\mathbf{y}_i\rceil_p$.
3. $\mathbf{E}_{i+1}\leftarrow D_\chi(\mathbf{r}_i)$.

It sets $\mathbf{y}_\tau\leftarrow\mathcal{G}_{\mathbf{S}_\tau,\mathbf{E}_\tau,T}(x)$. Before returning $\mathbf{y}_\tau$, the evaluation algorithm checks whether there exists an index $i\in[\tau]$ for which $\mathbf{y}_i\notin\mathsf{Borderline}_R$ for $R=|T|Bm^{e(T)}$. If this is not the case, then it returns $\perp$. Otherwise, it returns $\mathbf{y}_\tau\in\mathbb{Z}_q^m$.

Hybrid experiments. We now define our hybrid experiments as follows:

- HYB_0: This is the real PRF security experiment of Definition 3.5. Namely, the adversary $\mathcal{A}$ is given oracle access to the real PRF function $F_{\mathbf{S}}(\cdot)$ for $\mathbf{S}\xleftarrow{\text{R}}\mathbb{Z}_q^{n\times\tau}$. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta\in\{0,1\}$, which is the output of the experiment.

- HYB_1: This is the intermediate security experiment. The adversary $\mathcal{A}$ is given oracle access to the hybrid (randomized) PRF function $\tilde{F}_{\mathbf{S}}(\cdot)$ for $\mathbf{S}\xleftarrow{\text{R}}\mathbb{Z}_q^{n\times\tau}$. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta\in\{0,1\}$, which is the output of the experiment.

- HYB_2: This is the ideal PRF security experiment of Definition 3.5, but the challenger additionally checks the borderline condition. Specifically, for each query $x\in\{0,1\}^{|T|}$ that the adversary $\mathcal{A}$ makes, the challenger samples uniformly random vectors $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{\tau-1}\xleftarrow{\text{R}}\mathbb{Z}_q^m$. Then, it checks whether all of these vectors are contained in the borderline set $\mathsf{Borderline}_R$ for $R=|T|Bm^{e(T)}$. If this is the case, then it outputs $\perp$. Otherwise, it returns $\mathbf{y}_\tau\xleftarrow{\text{R}}\mathbb{Z}_q^m$ to $\mathcal{A}$.

- HYB_3: This is the ideal PRF security experiment of Definition 3.5. Namely, the adversary $\mathcal{A}$ is given oracle access to a random function $f(\cdot)$ for $f\xleftarrow{\text{R}}\mathsf{Funs}\big[\{0,1\}^{|T|},\mathbb{Z}_q^m\big]$. At the end of the experiment, $\mathcal{A}$ outputs a bit $\beta\in\{0,1\}$, which is the output of the experiment.

We show that the consecutive hybrid experiments HYB_i and HYB_{i+1} are either computationally or statistically indistinguishable for $i\in\{0,1,2\}$. As the proof of indistinguishability of HYB_0 and HYB_1 depends on the proof of indistinguishability of HYB_1 and HYB_2, we show that HYB_1 and HYB_2

are computationally indistinguishable before showing that HYB_0 and HYB_1 are computationally indistinguishable.

For the lemma statements below, we use the same notation as in the proof of Theorem 4.2. Namely, for an experiment HYB_i and an adversary $\mathcal{A}$, we define $\text{HYB}_i(\mathcal{A})$ to denote the random variable representing the output of experiment HYB_i with respect to the adversary $\mathcal{A}$.

Lemma B.5. *Suppose that the $\mathsf{LWE}_{n,q,\chi}$ problem is hard. Then, for any efficient adversary $\mathcal{A}$,*

$$\left| \Pr[\text{HYB}_1(\mathcal{A}) = 1] - \Pr[\text{HYB}_2(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda).$$

Proof. Although the proof of the lemma is conceptually simple, it does require some heavy notations. To facilitate the presentation, we first fix some additional notation. For a bit string x of length at least i, let $x_{(i)} = x_1 x_2 \cdots x_i$ denote the string of its first i bits, and let $x^{(i)}$ denote the remainder of the string. Let $\mathcal{P} \subset \mathbb{Z}^{n\ell}$ denote an arbitrary set of representatives of the quotient group $\mathbb{Z}_q^m / \mathbf{G}^\intercal \cdot \mathbb{Z}_q^n$, and define a family of auxiliary function $\mathbf{V}_T^{(i)} : \{0,1\}^{|T|} \to \mathbb{Z}^m$ as follows.

Definition B.6. For a full binary tree T, matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$, and $0 \le i \le |T|$, define the (randomized) function $\mathbf{V}_T^{(i)} : \{0,1\}^{|T|} \to \mathbb{Z}^m$ inductively as follows.

- For $i = 0$, set $\mathbf{V}_T^{(0)}(x) = 0$.
- For $i \ge 1$, the evaluation algorithm samples $\mathbf{v}_{x_{(i)}} \xleftarrow{\text{R}} \mathcal{P}$. It sets

$$\mathbf{V}_T^{(i)}(x) = \mathbf{v}_{x_{(i)}}^\intercal \cdot \mathbf{S}_{T^{(i-1)}}(x^{(i)}) + \mathbf{V}_T^{(i-1)}(x).$$

For the proof of the lemma, we also make an additional simplifying assumption on the adversary $\mathcal{A}$. Namely, we assume that the adversary $\mathcal{A}$ makes evaluation queries on a *distinct* set of inputs in the domain $\{0,1\}^{|T|}$. This is without loss of generality as the challenger can always keep track of $\mathcal{A}$'s previous queries and respond to $\mathcal{A}$'s duplicate queries by looking up its previous responses.

Now, we define a sequence of intermediate hybrid experiments $\text{HYB}_{1,i^*,\eta^*}$ for $i^* \in [\tau]$ and $\eta^* \in [|T|]$ as follows:

- $\text{HYB}_{1,i^*,\eta^*}$: The challenger maintains a look-up table $\mathcal{T} : \{0,1\}^{|T|-\eta^*} \to \mathbb{Z}_q^n$, which maps the prefixes of $x \in \{0,1\}^{|T|}$, which $\mathcal{A}$ submits as its evaluation queries to vectors in $\mathbb{Z}_q^n$. Specifically, for each of these queries $x \in \{0,1\}^{|T|}$, the challenger proceeds as follows:
  - The challenger first checks if $\mathcal{A}$ previously made a query $\hat{x} \in \{0,1\}^{|T|}$ for which $\hat{x}_{(\eta^*-1)} = x_{(\eta^*-1)}$. If this is the case, then it looks up the vector $\mathbf{s}_{x_{(\eta^*-1)}}$ from the table $\mathcal{T}$ and otherwise, it samples a fresh random vector $\mathbf{s}_{x_{(\eta^*-1)}} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and adds it to $\mathcal{T}$.

    Then, the challenger samples an error vector $\mathbf{E}_{i^*} \leftarrow \chi^{n\times\eta^*}$ and computes $\mathbf{y}_{i^*} \leftarrow \mathcal{G}_{\mathbf{s}_{i^*},\mathbf{E}_{i^*},T^{(\eta^*-1)}}(x^{(\eta^*-1)}) + \mathbf{V}_T^{(\eta^*-1)}(x)$. If $i^* = \tau$, then it returns $\mathbf{y}_\tau$. Otherwise, it sets $\mathbf{E}_{i^*+1} \leftarrow D(r_{i^*})$, samples $\mathbf{S} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times\tau-i^*}$, and iteratively computes for $i^* < i \le \tau$:
    1. $\mathbf{y}_i \leftarrow \mathcal{G}_{\mathbf{s}_{i^*},\mathbf{E}_{i^*},T}(x)$.
    2. $r_i \leftarrow \lfloor \mathbf{y}_i \rceil_p$.
    3. $\mathbf{E}_{i^*} \leftarrow D(r_{i^*})$.

Finally, it sets $\mathbf{y}_\tau \leftarrow \mathcal{G}_{\mathbf{s}_\tau,\mathbf{E}_\tau,T}(x)$. Before returning $\mathbf{y}_\tau$, the challenger samples random vector $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{i^*-1} \xleftarrow{\text{R}} \mathbb{Z}_q^n$. If all of the vectors $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{i^*-1},\mathbf{y}_{i^*},\ldots,\mathbf{y}_{\tau-1}$ are contained in the set $\mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$, then the challenger returns $\perp$. Otherwise, it returns $\mathbf{y}_\tau$ to $\mathcal{A}$.

It follows by definition that $\text{HYB}_1 = \text{HYB}_{1,1,1}$. We now show that each of the consecutive experiments are computationally indistinguishable.

Claim B.7. *Suppose that p divides q and the $\mathsf{LWE}_{n,q,\chi}$ problem is hard. Then, for any efficient adversary $\mathcal{A}$ and any $i^* \in [\tau]$, we have*

$$\big|\Pr[\text{HYB}_{1,i^*,|T|}(\mathcal{A}) = 1] - \Pr[\text{HYB}_{1,i^*+1,1}(\mathcal{A}) = 1]\big| = \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be a distinguisher for the experiments $\text{HYB}_{1,i^*,|T|}$ and $\text{HYB}_{1,i^*+1,1}$ that makes at most $Q = \mathsf{poly}(\lambda)$ queries. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the $\mathsf{LWE}_{n,q,\chi}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

- At the start of the experiment, algorithm $\mathcal{B}$ receives an $\mathsf{LWE}_{n,q,\chi}$ challenge $(\mathbf{A},\mathbf{B}) \in \mathbb{Z}_q^{n\times 2m} \times \mathbb{Z}_q^{Q\times 2m}$. It parses $\mathbf{A} = (\mathbf{A}_0,\mathbf{A}_1)$ and provides $\mathbf{A}_0,\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ to $\mathcal{A}$.
- The adversary $\mathcal{A}$ makes Q adaptive evaluation queries. For each of $\mathcal{A}$'s j^{th} query $x_j \in \{0,1\}^{|T|}$, algorithm $\mathcal{B}$ parses $\mathbf{B}^\mathsf{T} = (\mathbf{b}_1,\ldots,\mathbf{b}_Q)$ for $\mathbf{b}_j \in \mathbb{Z}_q^{2m}$, $\mathbf{b}_j^\mathsf{T} = (\mathbf{b}_{j,0}^\mathsf{T},\mathbf{b}_{j,1}^\mathsf{T})$, and computes
$$\mathbf{y}_{i^*}^\mathsf{T} = \mathbf{b}_{j,x_{|T|}}^\mathsf{T} + \mathbf{V}_T^{(|T|-1)}(x) \in \mathbb{Z}_q^m.$$
It then sets $\mathbf{r}_{i^*} \leftarrow \lfloor \mathbf{y}_{i^*} \rceil_p$, $\mathbf{E}_{i^*+1} \leftarrow D_\chi(\mathbf{r}_{i^*})$, and computes for $i^*+1 \le i < \tau$:
  1. $\mathbf{y}_i \leftarrow \mathcal{G}_{\mathbf{s}_i,\mathbf{E}_i,T}(x)$.
  2. $\mathbf{r}_i \leftarrow \lfloor \mathbf{y}_i \rceil_p$.
  3. $\mathbf{E}_{i+1} \leftarrow D_\chi(\mathbf{r}_i)$.

  Finally, it sets $\mathbf{y}_\tau \leftarrow \mathcal{G}_{\mathbf{s}_\tau,\mathbf{E}_\tau,T}(x)$. Before returning $\mathbf{y}_\tau$, algorithm $\mathcal{B}$ samples random vectors $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{i^*-1} \xleftarrow{\text{R}} \mathbb{Z}_q^m$. If all of the vectors $\widetilde{\mathbf{y}}_1,\ldots,\widetilde{\mathbf{y}}_{i^*-1},\mathbf{y}_{i^*},\ldots,\mathbf{y}_{\tau-1}$ are contained in the set $\mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$, then $\mathcal{B}$ returns $\perp$. Otherwise, it returns $\mathbf{y}_\tau$ to $\mathcal{A}$.
- At the end of the experiment, the adversary $\mathcal{A}$ outputs a bit $\beta \in \{0,1\}$. Algorithm $\mathcal{B}$ outputs the same bit β.

We now show that depending on whether the algorithm $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_\mathbf{s}^{(\mathsf{lwe})}$ or the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates $\mathcal{A}$'s view of either $\text{HYB}_{1,i^*,|T|}$ or $\text{HYB}_{1,i^*+1,1}$.

- If $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_\mathbf{s}^{(\mathsf{lwe})}$, then each vector $\mathbf{b}_{j,b}^\mathsf{T}$ for $j \in [Q]$ and $b \in \{0,1\}$ is a valid LWE sample $\mathbf{s}_j^\mathsf{T} \cdot \mathbf{A}_b + \mathbf{e}_{j,b}^\mathsf{T}$ for $\mathbf{s}_j \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e}_{j,b} \leftarrow \chi^m$. Therefore, by definition, $\mathcal{B}$ perfectly simulates $\text{HYB}_{1,i^*,|T|}$.
- If $\mathcal{B}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$, then each vector $\mathbf{b}_{j,b}$ for $j \in [Q]$ and $b \in \{0,1\}$ is a random vector in $\mathbb{Z}_q^m$ and therefore, for each of $\mathcal{A}$'s queries, the vector $\mathbf{y}_{i^*}^\mathsf{T} = \mathbf{b}_{j,x_{|T|}}^\mathsf{T} + \mathbf{V}_T^{(|T|-1)}(x)$ is also a uniformly random vector in $\mathbb{Z}_q^m$. Since p divides

q, this means that each vector $\mathbf{r}_{i^*} = \lfloor \mathbf{y}_{i^*} \rceil_p$ is also uniform in $\mathbb{Z}_p^m$ and hence $\mathbf{E}_{i^*+1}$ is distributed exactly as in $\chi^{m \times \tau}$. Hence, algorithm $\mathcal{B}$ perfectly simulates $\text{HYB}_{1,i^*+1,1}$.

We have shown that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwre})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates the distributions of either $\text{HYB}_{1,i^*,|T|}$ or $\text{HYB}_{1,i^*+1,1}$. Therefore, with the same distinguishing advantage of $\mathcal{A}$, algorithm $\mathcal{B}$ solves the $\mathsf{LWE}_{n,q,\chi}$ problem. It follows that assuming the hardness of the $\mathsf{LWE}_{n,q,\chi}$ problem, the distinguishing advantage of $\mathcal{A}$ is negligible. ☐

Claim B.8. *Suppose that p divides q and the $\mathsf{LWE}_{n,q,\chi}$ problem is hard. Then, for any efficient adversary $\mathcal{A}$ and any $i^* \in [\tau]$, $\eta^* \in [|T|]$, we have*

$$\left| \Pr[\text{HYB}_{1,i^*,\eta^*}(\mathcal{A}) = 1] - \Pr[\text{HYB}_{1,i^*,\eta^*+1}(\mathcal{A}) = 1] \right| = \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be a distinguisher for the experiments $\text{HYB}_{1,i^*,\eta^*}$ and $\text{HYB}_{1,i^*+1,1}$ that makes at most $Q = \mathsf{poly}(\lambda)$ queries. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the $\mathsf{LWE}_{n,q,\chi}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

- At the start of the experiment, algorithm $\mathcal{B}$ receives an $\mathsf{LWE}_{n,q,\chi}$ challenge $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{n \times 2m} \times \mathbb{Z}_q^{Q \times 2m}$. It parses $\mathbf{A} = (\mathbf{A}_0, \mathbf{A}_1)$ and provides $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ to $\mathcal{A}$.
- The adversary $\mathcal{A}$ makes Q adaptive evaluation queries. For each of $\mathcal{A}$'s j^{th} query $x_j \in \{0,1\}^{|T|}$, algorithm $\mathcal{B}$ parses $\mathbf{B}^\intercal = (\mathbf{b}_1, \ldots, \mathbf{b}_Q)$ for $\mathbf{b}_j \in \mathbb{Z}_q^{2m}$, $\mathbf{b}_j^\intercal = (\mathbf{b}_{j,0}^\intercal, \mathbf{b}_{j,1}^\intercal)$, and computes
$$\mathbf{y}_{i^*}^\intercal = \mathbf{b}_{j,x_{\eta^*}}^\intercal \cdot \mathbf{S}_{T^{(\eta^*)}}\left(x^{(\eta^*+1)}\right) + \mathbf{E}_{T^{(\eta^*+1)}}\left(x^{(\eta^*+1)}\right) + \mathbf{V}_T^{(\eta^*)}(x) \in \mathbb{Z}_q^m.$$
It then sets $\mathbf{r}_{i^*} \leftarrow \lfloor \mathbf{y}_{i^*} \rceil_p$, $\mathbf{E}_{i^*+1} \leftarrow D_\chi(\mathbf{r}_{i^*})$, and computes for $i^*+1 \le i < \tau$:
  1. $\mathbf{y}_i \leftarrow \mathcal{G}_{\mathbf{s}_i, \mathbf{E}_i, T}(x)$.
  2. $\mathbf{r}_i \leftarrow \lfloor \mathbf{y}_i \rceil_p$.
  3. $\mathbf{E}_{i+1} \leftarrow D_\chi(\mathbf{r}_i)$

  Finally, it sets $\mathbf{y}_\tau \leftarrow \mathcal{G}_{\mathbf{s}_\tau}, \mathbf{E}_\tau, T(x)$. Before returning $\mathbf{y}_\tau$, algorithm $\mathcal{B}$ samples random vectors $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{i^*-1} \xleftarrow{\text{R}} \mathbb{Z}_q^m$. If all of the vectors $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{i^*-1}, \mathbf{y}_{i^*}, \ldots, \mathbf{y}_{\tau-1}$ are contained in the set $\mathsf{Borderline}_R$ for $R = |T| B m^{e(T)}$, then $\mathcal{B}$ returns $\perp$. Otherwise, it returns $\mathbf{y}_\tau$ to $\mathcal{A}$.

We now show that depending on whether the algorithm $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or the ideal $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates $\mathcal{A}$'s view of either $\text{HYB}_{1,i^*,|T|}$ or $\text{HYB}_{1,i^*+1,1}$.

- If $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$, then each vector $\mathbf{b}_{j,b}^\intercal$ for $j \in [Q]$ and $b \in \{0,1\}$ is a valid LWE sample $\mathbf{s}_j^\intercal \cdot \mathbf{A}_b + \mathbf{e}_{j,b}^\intercal$ for $\mathbf{s}_j \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e}_{j,b} \leftarrow \chi^m$. Therefore, for each of $\mathcal{A}$'s j^{th} queries, we have
$$\begin{aligned}
\mathbf{y}_{i^*}^\intercal &= (\mathbf{s}_j^\intercal \cdot \mathbf{A}_b + \mathbf{e}_{j,b}^\intercal) \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \mathbf{E}_{T^{(i^*+1)}}\left(x^{(i^*+1)}\right) + \mathbf{V}_T^{(i^*)}(x) \\
&= \mathbf{s}_j^\intercal \cdot \mathbf{A}_b \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \left(\mathbf{e}_{j,b}^\intercal \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \mathbf{E}_{T^{(i^*+1)}}\left(x^{(i^*+1)}\right)\right) + \mathbf{V}_T^{(i^*)}(x) \\
&= \mathbf{s}_j^\intercal \cdot \mathbf{S}_{T^{(i^*-1)}}\left(x^{(i^*)}\right) + \mathbf{E}_{T^{(i^*)}}\left(x^{(i^*)}\right) + \mathbf{V}_T^{(i^*)}(x)
\end{aligned}$$
exactly as in $\text{HYB}_{1,i^*,\eta^*}$.

– If $\mathcal{B}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$, then each vector $\mathbf{b}_{j,b}$ for $j \in [Q]$ and $b \in \{0,1\}$ is a random vector in $\mathbb{Z}_q^m$ and therefore, for each of $\mathcal{A}$'s j^{th} queries, we have

$$\begin{aligned}
\mathbf{y}_{i^*}^\intercal &= \mathbf{b}_{j,b}^\intercal \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \mathbf{E}_{T^{(i^*+1)}}\left(x^{(i^*+1)}\right) + \mathbf{V}_T^{(i^*)}(x) \\
&= (\mathbf{s}_{j+1}^\intercal \cdot \mathbf{G} + \mathbf{v}_{x_{(j)}}) \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \mathbf{E}_{T^{(i^*+1)}}\left(x^{(i^*+1)}\right) + \mathbf{V}_T^{(i^*)}(x) \\
&= \left(\mathbf{s}_j^\intercal \cdot \mathbf{G} \cdot \mathbf{S}_{T^{(i^*)}}\left(x^{(i^*+1)}\right) + \mathbf{E}_{T^{(i^*+1)}}\left(x^{(i^*+1)}\right)\right) + \mathbf{v}_{x_{(j)}} + \mathbf{V}_T^{(i^*)}(x) \\
&= \mathcal{G}_{\mathbf{s}_j, \mathbf{E}_j, T^{(\eta^*)}}(x) + \mathbf{V}_T^{(i^*+1)}(x)
\end{aligned}$$

exactly as in $\textsc{hyb}_{1,i^*,\eta^*+1}$.

We have shown that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwre})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates the distributions of either $\textsc{hyb}_{1,i^*,\eta^*}$ or $\textsc{hyb}_{1,i^*,\eta^*+1}$. Therefore, with the same distinguishing advantage of $\mathcal{A}$, algorithm $\mathcal{B}$ solves the $\mathsf{LWE}_{n,q,\chi}$ problem. It follows that assuming the hardness of the $\mathsf{LWE}_{n,q,\chi}$ problem, the distinguishing advantage of $\mathcal{A}$ is negligible. ☐

Claim B.9. *Suppose that the $\mathsf{LWE}_{n,q,\chi}$ problem is hard. Then, for any efficient adversary $\mathcal{A}$, we have*

$$\left|\Pr[\textsc{hyb}_{1,\tau,|T|}(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_2(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. Let $\mathcal{A}$ be a distinguisher for the experiments $\textsc{hyb}_{1,\tau,|T|}$ and $\textsc{hyb}_2$ that makes at most $Q = \mathsf{poly}(\lambda)$ queries. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the $\mathsf{LWE}_{n,q,\chi}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

– At the start of the experiment, algorithm $\mathcal{B}$ receives an LWE challenge $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{n\times 2m} \times \mathbb{Z}_q^{Q\times m}$. It parses $\mathbf{A} = (\mathbf{A}_0, \mathbf{A}_1)$ and provides $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n\times m}$ to $\mathcal{A}$.
– The adversary $\mathcal{A}$ makes Q adaptive evaluation queries. For each of $\mathcal{A}$'s j^{th} query $x_j \in \{0,1\}^{|T|}$, algorithm $\mathcal{B}$ parses $\mathbf{B}^\intercal = (\mathbf{b}_1, \ldots, \mathbf{b}_Q)$ for $\mathbf{b}_j \in \mathbb{Z}_q^{2m}$, $\mathbf{b}_j^\intercal = (\mathbf{b}_{j,0}^\intercal, \mathbf{b}_{j,1}^\intercal)$, and computes
$$\mathbf{y}_\tau^\intercal = \mathbf{b}_{j,x_{|T|}}^\intercal + \mathbf{V}_T^{(|T|-1)}(x) \in Z_q^m.$$
Before returning $\mathbf{y}_\tau$, algorithm $\mathcal{B}$ samples random vectors $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{\tau-1} \xleftarrow{\text{R}} \mathbb{Z}_q^m$. If all of these vectors are contained in the set $\mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$, then $\mathcal{B}$ returns $\perp$. Otherwise, it returns $\mathbf{y}_\tau$ to $\mathcal{A}$.
– At the end of the experiment, the adversary $\mathcal{A}$ outputs a bit $\beta \in \{0,1\}$. Algorithm $\mathcal{B}$ outputs the same bit β.

We show that depending on whether the algorithm $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$ or the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates $\mathcal{A}$'s view of either $\textsc{hyb}_{1,\tau,|T|}$ or $\textsc{hyb}_2$.

– If $\mathcal{B}$ is interacting with the real LWE oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwe})}$, then each vector $\mathbf{b}_{j,b}^\intercal$ for $j \in [Q]$ and $b \in \{0,1\}$ is a valid LWE sample $\mathbf{s}_j^\intercal \cdot \mathbf{A}_b + \mathbf{e}_{j,b}^\intercal$ for $\mathbf{s}_j \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e}_{j,b} \leftarrow \chi^m$. Therefore, by definition, $\mathcal{B}$ perfectly simulates $\textsc{hyb}_{1,\tau,|T|}$.

– If $\mathcal{B}$ is interacting with the ideal oracle $\mathcal{O}^{(\mathsf{ideal})}$, then each vector $\mathbf{b}_{j,b}$ for $j \in [Q]$ and $b \in \{0,1\}$ is a random vector in $\mathbb{Z}_q^m$, and therefore, for each of $\mathcal{A}$'s queries, the vector $\mathbf{y}_\tau^\intercal = \mathbf{b}_{j,x,|T|}^\intercal + \mathbf{V}_T^{(|T|-1)}(x)$ is also a uniformly random vector in $\mathbb{Z}_q^m$. Hence, algorithm $\mathcal{B}$ perfectly simulates HYB_2.

We have shown that depending on whether $\mathcal{B}$ is interacting with the oracle $\mathcal{O}_{\mathbf{s}}^{(\mathsf{lwre})}$ or $\mathcal{O}^{(\mathsf{ideal})}$, it perfectly simulates the distributions of either $\text{HYB}_{1,\tau,|T|}$ or HYB_2. Therefore, with the same distinguishing advantage of $\mathcal{A}$, algorithm $\mathcal{B}$ solves the $\mathsf{LWE}_{n,q,\chi}$ problem. It follows that assuming the hardness of the $\mathsf{LWE}_{n,q,\chi}$ problem, the distinguishing advantage of $\mathcal{A}$ is negligible. □

□

Lemma B.10. *Suppose that $(2Rmp/q)^{\tau-1} = \mathsf{negl}(\lambda)$ and the $\mathsf{LWE}_{n,q,\chi}$ problem is hard. Then, for any efficient adversary $\mathcal{A}$,*

$$\big|\Pr[\text{HYB}_0(\mathcal{A}) = 1] - \Pr[\text{HYB}_1(\mathcal{A}) = 1]\big| = \mathsf{negl}(\lambda).$$

Proof. The difference between the evaluation algorithms of $F_{\mathbf{S}}(\cdot)$ in HYB_0 and $\tilde{F}_{\mathbf{S}}(\cdot)$ in HYB_1 are as follows:

- On input $x \in \{0,1\}^{|T|}$, the evaluation algorithm in HYB_0 sets the initial error term to be an all-zero matrix $\mathbf{E}_1 = \mathbf{0}$ while the evaluation algorithm in HYB_1 samples the initial error term from the error distribution $\tilde{\mathbf{E}}_1 \xleftarrow{\text{R}} \chi^{n\times m}$.
- The evaluation algorithm of $\tilde{F}_{\mathbf{S}}(\cdot)$ in HYB_1, additionally checks for the borderline condition. Specifically, on input $x \in \{0,1\}^{|T|}$, let $\mathbf{y}_1, \ldots, \mathbf{y}_\tau$ be the intermediate computation vectors $\mathbf{y}_i \leftarrow \mathcal{G}_{\mathbf{s}_i,\mathbf{E}_i,T}(x)$ for $i^* \in [\tau]$. Then, in the evaluation algorithm additionally checks whether there exists an index $i^* \in [\tau]$ for which $\mathbf{y}_{i^*} \notin \mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$. If this is not the case, then it returns $\perp$.

Now, for the analysis, consider a single fixed input $x \in \{0,1\}^{|T|}$ that an adversary $\mathcal{A}$ submits as a query and consider the two transcripts of the executions $F_{\mathbf{S}}(x)$ and $\tilde{F}_{\mathbf{S}}(x)$. Specifically, let $\mathbf{r}_1, \ldots, \mathbf{r}_\tau$ be the set of random vectors that are set during the execution of $F_{\mathbf{S}}(x)$ in and let $\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_\tau$ be the set of random vectors that are set during the execution of $\tilde{F}_{\mathbf{S}}(x)$. We first note that if there exists an index $i^* \in [\tau]$ for which $\lfloor \mathcal{G}_{\mathbf{s}_{\hat{i}},\mathbf{E}_{i^*},T}(x) \rceil_p = \lfloor \mathcal{G}_{\mathbf{s}_{\hat{i}},\tilde{\mathbf{E}}_{i^*},T}(x) \rceil_p$, then by definition, $\mathbf{r}_i = \tilde{\mathbf{r}}_i$ for all $i^* < i \le \tau$. Furthermore, by construction, for any $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{E} \leftarrow \chi^{n\times m}$, any full binary tree T, and $x \in \{0,1\}^{|T|}$, we have

$$\mathcal{G}_{\mathbf{s},\mathbf{E},T}(x) = \mathbf{s}^\intercal \mathbf{A}_T(x) + \mathcal{E}_T(x),$$

where $|\mathcal{E}_T(x)| \le |T|Bm^{e(T)}$ (Lemma B.4). This means that by the definition of rounding, as long as there exists a single index $i \in [\tau]$ for which the intermediate vector $\mathbf{y}_i = \mathcal{G}_{\mathbf{s}_i,\mathbf{E}_i,T}(x)$ is not contained in the set $\mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$, the final output of $F_{\mathbf{S}}(x)$ and $\tilde{F}_{\mathbf{S}}(x)$ coincide. In other words, if any efficient adversary $\mathcal{A}$ can force the function $\tilde{F}_{\mathbf{S}}(\cdot)$ to output $\perp$ only with probability $\mathsf{negl}(\lambda)$, the experiments HYB_0 and HYB_1 are computationally indistinguishable.

$$\big|\Pr[\text{HYB}_0(\mathcal{A}) = 1] - \Pr[\text{HYB}_1(\mathcal{A}) = 1]\big| \le \Pr\left[\mathcal{A}^{\tilde{F}_{\mathbf{S}}(\cdot)}(1^\lambda) \to x \wedge \tilde{F}_{\mathbf{S}}(x) = \perp\right].$$

To show that

$$\Pr\left[\mathcal{A}^{\tilde{F}_{\mathbf{S}}(\cdot)}(1^\lambda) \to x \wedge \tilde{F}_{\mathbf{S}}(x) = \perp\right] = \mathsf{negl}(\lambda),$$

we first note that by Lemma B.5, the hybrid experiments $\textsc{hyb}_1$ and $\textsc{hyb}_2$ are computationally indistinguishable. Therefore,

$$\Pr\left[\mathcal{A}^{\tilde{F}_{\mathbf{S}}(\cdot)}(1^\lambda) \to x \wedge \tilde{F}_{\mathbf{S}}(x) = \perp\right] \leq \Pr\left[\mathcal{A}^{\mathcal{O}_{\textsc{hyb}_2}(\cdot)}(1^\lambda) \to x \wedge \mathcal{O}_{\textsc{hyb}_2}(x) = \perp\right] + \mathsf{negl}(\lambda), \tag{B.3}$$

where $\mathcal{O}_{\textsc{hyb}_2}(\cdot)$ denotes the response of the challenger in $\textsc{hyb}_2$. Furthermore, in $\textsc{hyb}_2$, the challenger outputs $\perp$ only when $\widetilde{\mathbf{y}}_i \in \mathsf{Borderline}_R$ for all $i \in [\tau - 1]$, $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{\tau-1} \xleftarrow{\text{R}} \mathbb{Z}_q^m$. Therefore, the challenger outputs $\perp$ with probability at most $(2Rmp/q)^{\tau-1}$, which is negligible by assumption

$$\Pr\left[\mathcal{A}^{\mathcal{O}_{\textsc{hyb}_2}(\cdot)}(1^\lambda) \to x \wedge \mathcal{O}_{\textsc{hyb}_2}(x) = \perp\right] = \mathsf{negl}(\lambda). \tag{B.4}$$

The lemma follows by combining (B.3) and (B.4). □

Lemma B.11. *Suppose that* $(2Rmp/q)^{\tau-1} = \mathsf{negl}(\lambda)$. *Then, for any (unbounded) adversary* $\mathcal{A}$ *making at most* $Q = \mathsf{poly}(\lambda)$ *evaluation queries,*

$$\left|\Pr[\textsc{hyb}_2(\mathcal{A}) = 1] - \Pr[\textsc{hyb}_3(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the experiments $\textsc{hyb}_2$ and $\textsc{hyb}_3$ is in the way the challenger returns $\perp$ to $\mathcal{A}$'s queries. In $\textsc{hyb}_2$, the challenger samples random vectors $\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{\tau-1} \xleftarrow{\text{R}} \mathbb{Z}_q^m$ and returns $\perp$ if all of these vectors are contained in $\mathsf{Borderline}_R$ for $R = |T|Bm^{e(T)}$. In $\textsc{hyb}_3$, the challenger never returns $\perp$.

Since the components of each vector $\widetilde{\mathbf{y}}_i$ for $i \in [\tau - 1]$ are sampled independently, the probability that a single vector $\widetilde{\mathbf{y}}_i \in \mathsf{Borderline}_R$ is at most $2Rmp/q$, and the probability all of these vectors are contained in $\mathsf{Borderline}_R$ is at most $(2Rmp/q)^{\tau-1}$, which is negligible. Since $\mathcal{A}$ can make at most a polynomial number of evaluation queries, it follows that the experiments $\textsc{hyb}_2$ and $\textsc{hyb}_3$ are statistically indistinguishable by the union bound. □