More Efficient Lattice PRFs from Keyed Pseudorandom Synthesizers*

Hart Montgomery
hmontgomery@us.fujitsu.com
Fujitsu Laboratories of America

November 5, 2018

Abstract

We develop new constructions of lattice-based PRFs using keyed pseudorandom synthesizers. We generalize all of the known 'basic' parallel lattice-based PRFs–those of [BPR12], [BLMR13], and [BP14]–to build highly parallel lattice-based PRFs with smaller modulus (and thus better reductions from worst-case lattice problems) while still maintaining computational efficiency asymptotically equal to the fastest known lattice-based PRFs at only the cost of larger key sizes.

In particular, we build several parallel (in NC^2) lattice-based PRFs with modulus independent of the number of PRF input bits based on both standard LWE and ring LWE. Our modulus for these PRFs is just $O\left(m^{f(m)}\right)$ for lattice dimension m and any function $f(m) \in \omega(1)$. The only known parallel construction of a lattice-based PRF with such a small modulus is a construction from Banerjee's thesis [Ban15], and some of our parallel PRFs with equivalently small modulus have smaller key sizes and are very slightly faster (when using FFT multiplication). These PRFs also asymptotically match the computational efficiency of the most efficient PRFs built from *any* LWE- or ring LWE-based assumptions known[1], respectively, and concretely require less computation per output than any known parallel lattice-based PRFs (again when using FFT multiplication).

We additionally use our techniques to build other efficient PRFs with very low circuit complexity (but higher modulus) which improve known results on highly parallel lattice PRFs. For instance, for input length λ, we show that there exists a ring LWE-based PRF in NC^1 with modulus proportional to m^{λ^c} for any $c \in (0,1)$. Constructions from lattices with this circuit depth were only previously known from larger moduli.

1 Introduction

Pseudorandom functions, first defined by Goldreich, Goldwasser, and Micali [GGM84], are one of the most fundamental building blocks in cryptography. They are used for a wide variety of cryptographic applications, including encryption, message integrity, signatures, key derivation, user authentication, and much more. PRFs are important in computational complexity as well since they can be used to build lower bounds in learning theory.

In a nutshell, a PRF is a function that is indistinguisahble from a truly random function[2]. The most efficient PRFs are built from block ciphers like AES and security is based on ad-hoc *interactive* assumptions. It is a longstanding open problem to construct PRFs that are efficient

*An edited version of this paper appears in Indocrypt 2018.

[1] At least, at the time of initial publication. We discuss concurrent work in the introduction.

[2] We give a precise definition in section 2.

as these block ciphers from *offline* assumptions like factoring or the decisional Diffie-Hellman problem. The history of PRFs based on standard, offline assumptions is long and filled with many interesting constructions [NR97]. For a full treatment of PRFs and their applications, we highly recommend reading [BR17]. In this work, however, we specifically focus on lattice-based PRFs.

While there are many desirable properties of good PRFs, three that immediately come to mind are speed, parallelization, and cryptographic hardness. Speed speaks for itself: all other things equal, faster PRFs are better. Parallelization is also another desired quality: it means that PRFs can practically be computed more quickly and has interesting implications for complexity theory [BFKL94]. Of course, PRFs that are harder to break are also more desirable. Throughout this paper we will examine all of these PRF qualities.

Learning with Errors. In this work, we base our PRFs on the hardness of the *learning with errors* (LWE) problem [Reg05], which is the most commonly used lattice problem in cryptography[3]. Informally, the LWE problem is, for a uniformly random fixed key $\mathbf{s} \in \mathbb{Z}_q^n$, random samples $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$, and discrete Gaussian noise terms δ_i, to distinguish from random the distribution consisting of samples of the form $(\mathbf{a}_i, \mathbf{a}_i^\intercal \cdot \mathbf{s} + \delta_i \mod q)$.

Regev [Reg05] showed that solving the LWE problem is as hard as finding approximate solutions to certain worst-case lattice problems. The quality of the approximate solution (and thus the hardness of the problem solved) was proportional to the ratio of the modulus q to the width of the Gaussian noise terms. Most LWE-based cryptosystems today rely on the hardness of an LWE instance with a small, polynomial Gaussian noise distribution, so the hardness of the scheme is typically directly tied to the modulus q. Thus, decreasing the modulus of LWE-based cryptosystems is an important goal across many areas of lattice cryptography[4].

1.1 Lattice-Based PRFs

It has been known how to build completely sequential (and thus high depth) PRFs from LWE by using generic constructions like [GGM84] since the original LWE result [Reg05] was published. For instance, it is possible to build a very simple lattice-based PRF using the [GGM84] construction by treating LWE as a PRG. This simple construction also has the added benefit of a polynomially sized modulus q. However, these PRFs from generic constructions are maximally sequential and very inefficient, since LWE noise (i.e. Gaussians) has to be sampled at every step in the generic construction.

The study of PRFs based on lattice problems truly began in 2011, when Banerjee, Peikert, and Rosen [BPR12] invented the learning with rounding (LWR) problem, reduced to it from LWE, and showed that it could be used to build efficient and highly parallel PRFs. The authors built three new PRFs using the new rounding technique: one using the GGM construction, one using pseudorandom synthesizers, and one direct construction. The ring-based direct construction had the nice property that it could be implemented in NC^1, even if it was slightly less efficient than the generic constructions.

In a follow-up work, Boneh, Lewi, Montgomery, and Raghunathan [BLMR13] invented the first key homomorphic PRF in the standard model (from any assumption) using lattices. While their PRF was not extremely efficient, key homomorphic PRFs have a wide variety of applications, and their techniques (in particular, the use of LWE samples with low noise) turned out to be useful in other applications.

[3]Please see section 2 for a comprehensive definition of and discussion on the LWE problem.

[4]For a full treatment of lattice and LWE complexity, we strongly recommend [MG12]

In a follow-up work, Banerjee and Peikert [BP14] developed a general family of key homomorphic PRFs that dramatically improved upon the PRFs in [BLMR13] and even were (for certain choices of parameters) competitive with the non-key homomorphic PRFs of [BPR12] in terms of performance. The authors used a clever tree structure and rigorous analysis to carefully schedule 'bit decomposition' that allowed for good performance while still managing to retain key homomorphism.

In his Ph.D. thesis, Banerjee [Ban15] further improved the pseudorandom synthesizer construction technique from [BPR12], which allowed for tighter asymptotic constructions than previously known[5]. Around the same time, Döttling and Schröder [DS15] showed how to use their general technique of on-the-fly adaptation to also build LWE-based PRFs with relatively small moduli from low-depth circuits.

Concurrent Work. Very recently, and in a work concurrent with (and independent from) ours, Jager, Kurek, and Pan [JKP18] introduce all-prefix universal hash functions and show how to use these in conjunction with the augmented cascade construction [BMR10] to build efficient lattice-based PRFs with slighty superpolynomial modulus. Their LWE-based PRF can be thought of as a much more efficient version of [DS15]. We do not fully analyze this construction here, but it is likely more efficient than ours (although it does not have quite as small of a modulus as some of our constructions).

Application-Focused Lattice PRFs. Lattice PRFs have also been used for a number application specific PRFs, including puncturable PRFs [GGM84], constrained PRFs [BW13] [DKW16] (including key homomorphic constrained PRFs [BFP+15] [BV15]), PRFs secure against related key attacks [LMR14], and PRFs that hide constraints or functions [CC17] [BKM17] [BTVW17] [KW17] [PS17]. It is not known how to achieve many of these results from standard, non-lattice assumptions. Moreover, many of these works utilize very strong versions of the LWE assumption. It is our hope that the techniques introduced in this paper can be used to improve the efficiency and assumptions of some of these works.

1.2 Pseudorandom Synthesizers and Lattices

Pseudorandom synthesizers were first invented by Naor and Reingold in their famous work [NR95] as a way to construct PRFs with low circuit depth. The first synthesizer PRF constructions from lattices were introduced in [BPR12].

It can be cumbersome to define synthesizer PRFs in a way that is immediately understandable, so we present an 8-bit version of the synthesizer PRF from [BPR12]. Let the matrices $\mathbf{S}_{i,b} \in \mathbb{Z}_q^{m \times m}$ for $i \in [1, ..., 8]$ and $b \in \{0, 1\}$ be sampled uniformly at random. The original lattice-based synthesizer construction of [BPR12] had the following form on an 8-bit input $\mathbf{x} = x_1 ... x_8$:

$$\left\lfloor \left\lfloor \lfloor \mathbf{S}_{1,x_1} \cdot \mathbf{S}_{2,x_2} \rceil_{p_2} \cdot \lfloor \mathbf{S}_{3,x_3} \cdot \mathbf{S}_{4,x_4} \rceil_{p_2} \right\rfloor_{p_1} \cdot \left\lfloor \lfloor \mathbf{S}_{5,x_5} \cdot \mathbf{S}_{6,x_6} \rceil_{p_2} \cdot \lfloor \mathbf{S}_{7,x_7} \cdot \mathbf{S}_{8,x_8} \rceil_{p_2} \right\rfloor_{p_1} \right\rfloor_{p_0}$$

Note that this construction has the unfortunate requirement that $q >> p_2 >> p_1 >> p_0$. In his thesis [Ban15], Banerjee showed how to eliminate this 'tower of moduli' requirement from this synthesizer construction by using rectangular matrices. To illustrate this, suppose we set our modulus q and our rounding parameter p such that $q = p^2$. Let the matrices $\mathbf{S}_{i,b}$ for $i \in [1, ..., 4]$

[5] To our knowledge this result has not been formally published in conference proceedings.

and $b \in \{0,1\}$ now be defined such that $\mathbf{S}_{i,b} \in \mathbb{Z}_q^{m \times 2m}$. We can take the product of the transpose of one of these matrices with another and round in the following way:

$$\left\lfloor \left[\begin{array}{c} \mathbf{S}_{i,b_i}{}^{\intercal} \end{array} \right] \cdot \left[\begin{array}{cc} & \mathbf{S}_{i+1,b_{i+1}} \end{array} \right] \right\rceil_p = \mathbf{T} \in \mathbb{Z}_p^{2m \times 2m}$$

for some matrix $\mathbf{T}$ that will be indistinguishable from random by the hardness of LWR. In addition, note that $\mathbf{T}$ has enough entropy to produce a new, uniformly random matrix $\mathbf{S}' \in \mathbb{Z}_q^{m \times 2m}$. In fact, we can just set

$$\mathbf{S}' = \mathbf{T} \cdot \left[\begin{array}{c} \mathbf{I}_m \\ q\mathbf{I}_m \end{array} \right]$$

to trivially extract this randomness. Suppose we now consider a 4-bit input $\mathbf{x} = x_1 ... x_4$: if we put this all together, we can present a four-bit version of the PRF based on the improved synthesizer from [Ban15] in the following way:

$$\left\lfloor \left(\left\lfloor \left[\begin{array}{c} \mathbf{S}_{1,x_1}{}^{\intercal} \end{array} \right] \cdot \left[\begin{array}{cc} & \mathbf{S}_{2,x_2} \end{array} \right] \right\rceil_p \cdot \left[\begin{array}{c} \mathbf{I}_m \\ q\mathbf{I}_m \end{array} \right] \right) \cdot \left(\left\lfloor \left[\begin{array}{c} \mathbf{S}_{3,x_3}{}^{\intercal} \end{array} \right] \cdot \left[\begin{array}{cc} & \mathbf{S}_{4,x_4} \end{array} \right] \right\rceil_p \cdot \left[\begin{array}{c} \mathbf{I}_m \\ q\mathbf{I}_m \end{array} \right] \right)^{\intercal} \right\rceil_p \cdot \left[\begin{array}{c} \mathbf{I}_m \\ q\mathbf{I}_m \end{array} \right]$$

This new synthesizer construction from [Ban15] was the first lattice-based PRF construction where the modulus q was independent of the input length λ of the PRF.

1.3 Our Contributions

In this paper, we introduce a new, general technique that we use to build new lattice-based PRFs by applying a pseudorandom synthesizer structure [NR95] to the three main generic PRF constructions of [BPR12], [BLMR13], and [BP14]. While our constructions are not key homomorphic, they are either as efficient or more efficient and have as small or smaller modulus (and thus better reductions to worst-case lattice problems) than existing lattice-based PRFs. For lattice-based PRFs with any degree of parallelism, only the synthesizer-based PRF from Banerjee's thesis [Ban15] matches the most efficient of our constructions asymptotically, and our constructions are (slightly) more efficient in practice assuming we use fast Fourier transform multiplication[6].

In order to illustrate our construction technique, we will start by considering the [BLMR13] PRF F_{BLMR}. Recall that F_{BLMR} uses two public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_2^{m \times m}$ where the entries of these matrices are sampled uniformly at random from $\{0,1\}$ such that $\mathbf{A}_0$ and $\mathbf{A}_1$ are full-rank. The dimension m is derived from the security parameter, and the key for the PRF is a single vector $\mathbf{k} \in \mathbb{Z}_q^m$ and its input domain is $\{0,1\}^\lambda$. We also need an integer modulus q and a rounding parameter p. The PRF at the point $\mathbf{x} = x_1 \cdots x_\lambda \in \{0,1\}^\lambda$ is defined as

$$F_{BLMR}(\mathbf{k}, x) = \left\lceil \prod_{i=1}^{\lambda} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rfloor_p \tag{1.1}$$

[6] See [Fat06] and especially [KSN+04] for a discussion of integer multiplication algorithms.

where $\lceil \cdot \rfloor_p$ denotes the standard rounding operation[7]. F_{BLMR} is both key homomorphic and massively parallelizable (in NC^2) which is quite desirable. However the construction is quite inefficient, and the modulus q required for the PRF to be secure is enormous ($\log q$ scales linearly with the input length λ), meaning that the worst-case lattice problems that we can reduce to the PRF require quite strong assumptions. The work of [BP14] aims to alleviate some of these problems by cleverly inserting some 'bit decomposition' operations into the evaluation of the PRF, getting an overall tree structure that results in more efficient PRFs with better modulus.

Many years ago, the cascade construction [BCK96] and the augmented cascade construction [BMR10] were used to build more efficient PRFs by adding 'key material' at every layer of the PRF. This generally increased the key size of the resulting PRFs, but increased efficiency and (sometimes) allowed for weaker assumptions. To this point, no such ideas have been applied to PRFs based on lattice assumptions. While we cannot directly utilize these constructions for lattices, we can apply their core idea–add 'key material' at every layer of the construction–to build more efficient PRFs.

To this end, suppose we view F_{BLMR} in tree form and add an additional secret key at every layer of the tree (up to $\log \lambda$ total). In order to do this efficiently, we use a pseudorandom synthesizer construction [NR95]. Pseudorandom synthesizers, which we define in section 3, are efficient ways to construct parallelizable PRFs.

In our work, we construct a keyed synthesizer S_ℓ which has a square matrix $\mathbf{S} \in \mathbb{Z}_q^{m \times m}$ with entries sampled uniformly at random over $\mathbb{Z}_q$ as a key. Our synthesizer is parameterized by a parallelization factor ℓ and uses ℓ 'lists' of binary random matrices $\mathbf{A}_{i,x_i} \in \mathbb{Z}_2^{m \times m}$, where $i \in [1, \ell]$ is the list indicator and $x_i \in [1, k_i]$ is the index of a particular matrix $\mathbf{A}_{i,x_i}$ in the list i. Each output block of our synthesizer S_ℓ looks like the following:

$$S_\ell\left(x_1, ...x_\ell\right) \stackrel{\mathsf{def}}{=} \left\lceil \left[\prod_{i=1}^{\ell} \mathbf{A}_{i,x_i} \right] \mathbf{S} \right\rfloor_p$$

S_ℓ looks very much like F_{BLMR} with input length ℓ, although there are some key differences that we need in order for the synthesizer construction to efficiently work. We rigorously define and prove the security of this synthesiser S_ℓ later in the paper, and show how the pseudorandom synthesizer construction of [NR95] can be used to turn various versions of this synthesizer into PRFs. The proof of security borrows elements from the proofs of [BLMR13] and especially [BP14].

Now suppose that we set our rounding parameter $p = 2$. This will turn out to be a practical parameter choice. We next select binary matrices $\mathbf{A}_{i,b} \in \mathbb{Z}_2^{m \times m}$ uniformly at random for $i \in [1, ..., 8]$ and $b \in \{0, 1\}$ and keys $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3 \in \mathbb{Z}_q^{m \times m}$ uniformly at random. Our synthesizer S_2 can be used to build what we call the PRF F^2 which, on 8-bit input $\mathbf{x} = x_1...x_8$, gives us the following construction:

$$\left\lfloor \left\lfloor \lfloor \mathbf{A}_{1,x_1} \mathbf{A}_{2,x_2} \mathbf{S}_1 \rceil_2 \cdot \lfloor \mathbf{A}_{3,x_3} \mathbf{A}_{4,x_4} \mathbf{S}_1 \rceil_2 \mathbf{S}_2 \right\rceil_2 \cdot \left\lfloor \lfloor \mathbf{A}_{5,x_5} \mathbf{A}_{6,x_6} \mathbf{S}_1 \rceil_2 \cdot \lfloor \mathbf{A}_{7,x_7} \mathbf{A}_{8,x_8} \mathbf{S}_1 \rceil_2 \mathbf{S}_2 \right\rceil_2 \mathbf{S}_3 \right\rceil_2 \tag{1.2}$$

Note that each rounded subset product (i.e. $\lfloor \mathbf{A}_{1,x_1} \mathbf{A}_{2,x_2} \mathbf{S}_1 \rceil_2$) evaluates to a new random-looking matrix over $\mathbb{Z}_2^{m \times m}$ (assuming the nonuniform LWE assumption from [BLMR13]), so after one stage of (parallel) evaluation, the above 8-bit PRF looks like the following:

$$\left\lfloor \left\lfloor \tilde{\mathbf{A}}_{x_1,x_2} \tilde{\mathbf{A}}_{x_3,x_4} \mathbf{S}_2 \right\rceil_2 \cdot \left\lfloor \tilde{\mathbf{A}}_{x_5,x_6} \tilde{\mathbf{A}}_{x_7,x_8} \mathbf{S}_2 \right\rceil_2 \mathbf{S}_3 \right\rceil_2$$

[7] $\lceil \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lceil x \rfloor_p = i$, where $i \cdot \lfloor q/p \rfloor$ is the largest multiple of $\lfloor q/p \rfloor$ that does not exceed x

for random-looking matrices $\tilde{\mathbf{A}}_{x_i,x_{i+1}} \in \mathbb{Z}_2^{m\times m}$ that depend on the bits x_i and x_{i+1}. We can also generalize and pick higher values of ℓ, some of which will have interesting ramifications. Below we show a (abbreviated) 16-bit construction of a PRF F^4 using S_4:

$$\left\lfloor \lfloor \mathbf{A}_{1,x_1}\mathbf{A}_{2,x_2}\mathbf{A}_{3,x_3}\mathbf{A}_{4,x_4}\mathbf{S}_1 \rceil_2 \cdot \lfloor \mathbf{A}_{5,x_5}\mathbf{A}_{6,x_6}\mathbf{A}_{7,x_7}\mathbf{A}_{8,x_8}\mathbf{S}_1 \rceil_2 \cdot \lfloor \cdot \rceil_2 \cdot \lfloor \cdot \rceil_2 \cdot \mathbf{S}_2 \right\rceil_2$$

The inputs to each layer of our PRFs are new, random-looking binary matrices, which allows our synthesizer to compose nicely. Right away, it should be obvious that our synthesizer offers some advantages over the basic synthesizer construction of [BPR12]. Most obviously, our rounding parameter and modulus can be independent of the PRF length. However, some of the comparisons are a bit more nuanced. While we can build PRFs from (almost) any choice of ℓ–including PRFs that include synthesizers with different choices of ℓ–we examine one particular choice in section 5 which we briefly discuss here.

PRF F^2 from Synthesizer S_2. Our PRF F^2, which we showed for 8 bits in equation 1.2, is one of the simplest PRFs we can build, but also one of the most efficient. F^2 has modulus $O\left(m^{\omega(1)}\right)$[8] which is currently the (asymptotically) smallest known modulus for any lattice PRF that uses rounding. We note that this modulus is independent of the input length λ of the PRF. F^2 is as asymptotically efficient in terms of output per work as the naive pseudorandom synthesizer of [BPR12], which is currently the most efficient known PRF in this regard from standard lattices. Additionally, F^2 can be computed (practically, even) in circuit class NC^2, meaning that it is highly parallelizable. The only drawback of F^2 is the relatively large key size.

F^2 is the second (after that in [Ban15]) known lattice PRF with modulus $m^{\omega(1)}$ independent of the number of input bits of the PRF with *any* sublinear circuit depth and also happens to also be one of the most efficient known lattice PRF (from standard lattices) in terms of output per work.

Synthesizing [BPR12]. We can also apply pseudorandom synthesizers to other lattice PRF constructions. The logical place to continue is, of course, the original lattice PRF construction: [BPR12]. There are substantial differences between [BLMR13] and [BPR12], the largest of which is that F_{BLMR} uses a LWE sample subset-product structure while the direct PRF F_{BPR} from [BPR12] uses a key subset-product structure. However, it turns out we can still build interesting PRFs by synthesizing F_{BPR} with a few minor tricks.

In this vein, we next construct a keyed synthesizer K_ℓ inspired by (and almost identical to) F_{BPR} which has a square matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times m}$ with entries sampled uniformly at random over $\mathbb{Z}_q$ as a key. K_ℓ is parameterized by a parallelization factor ℓ and uses ℓ 'lists' of random matrices $\mathbf{S}_{i,x_i} \in \mathbb{Z}_p^{m\times m}$ for some superpolynomially large p, where $i \in [1,\ell]$ is the list indicator and $x_i \in [1,k_i]$ is the index of a particular matrix $\mathbf{S}_{i,x_i}$ in the list i. Each output block of K_ℓ looks like the following:

$$K_\ell\left(x_1,...x_\ell\right) \stackrel{\text{def}}{=} \left\lceil \mathbf{A}\left[\prod_{i=1}^{\ell}\mathbf{S}_{i,x_i}\right]\right\rfloor_p \tag{1.3}$$

We chose to make the keys uniformly random in order to make the synthesizer compose properly when used to construct PRFs[9]. Of course, proving synthesizer security requires that LWE problem is hard with superpolynomially large uniform key and noise, but this is a very straightforward (and probably known in folklore) result. We prove this as a part of our analysis. In

[8] We use this as shorthand for $O\left(m^{f(m)}\right)$ for any function $f(m) \in \omega(1)$. This is technically incorrect, but a nice convenience and is common in LWR literature.

[9] There are other choices available for the key distribution here–perhaps even more efficient ones.

the meantime, we again demonstrate how such a synthesizer would look by spelling out an 8-bit version of K_ℓ.

Suppose we select 'public samples' $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3 \in \mathbb{Z}_q^{m \times m}$ uniformly at random and 'secret' matrices $\mathbf{S}_{i,b} \in \mathbb{Z}_p^{m \times m}$ uniformly at random for $i \in [1, ..., 8]$ and $b \in \{0, 1\}$. K_2 can be used to build a PRF that we refer to as P^2, which, on 8-bit input $\mathbf{x} = x_1 ... x_8$, gives us the following construction (where all computations are performed $\mod q$):

$$\left\lfloor \mathbf{A}_1 \left\lfloor \mathbf{A}_2 \left\lfloor \mathbf{A}_3 \mathbf{S}_{1,x_1} \mathbf{S}_{2,x_2} \right\rceil_p \cdot \left\lfloor \mathbf{A}_3 \mathbf{S}_{3,x_3} \mathbf{S}_{4,x_4} \right\rceil_p \right\rfloor_p \cdot \left\lfloor \mathbf{A}_2 \left\lfloor \mathbf{A}_3 \mathbf{S}_{5,x_5} \mathbf{S}_{6,x_6} \right\rceil_p \cdot \left\lfloor \mathbf{A}_3 \mathbf{S}_{7,x_7} \mathbf{S}_{8,x_8} \right\rceil_p \right\rfloor_p \right\rceil_p \tag{1.4}$$

As is evident from the equations, one can view K_ℓ as a sort of key-sample flip-flop with S_ℓ. However, this relationship is not exact, since the hardness results of LWE with different distributions of keys and samples are not equivalent.

Moving to Rings. While PRFs derived from K_ℓ (which we will call P^ℓ) are a little bit more complicated and slightly less efficient than those built from S_ℓ (although this doesn't show up asymptotically under $\tilde{O}(\cdot)$ notation), they have one huge advantage over constructions from S_ℓ: they admit ring instantiations. As in [BPR12], we can almost immediately derive a ring form of K_ℓ, which we call $K_{R,\ell}$, and a corresponding PRF $P^{R,\ell}$. These ring PRFs allow us to match the efficiency of all previously known ring LWE-based PRFs while mainaining a slightly superpolynomial modulus at the cost of only more key size.

PRFs P^2 and $P^{R,2}$ from Synthesizers K_2 and $K_{R,2}$. Our PRF P^2, which we showed for 8 bits in equation 1.4, turns out to have parameters almost exactly asymptotically equivalent to F^2, including modulus $O\left(m^{\omega(1)}\right)$. The only difference is that P^2 has large secret keys, while F^2 has large public parameters.

We see substantial improvements when we move to rings. The PRF $P^{R,2}$ is the second (after that in [Ban15]) known PRF based on the hardness of ring LWE with modulus $m^{\omega(1)}$ (where m is now the degree of the polynomial of the ring R) with *any* sublinear circuit depth. In addition, $P^{R,2}$ matches the most efficient known ring LWE-based PRFs (like those of [BP14]) in terms of asymptotic computational efficiency. Again, the only drawback is larger key sizes.

Synthesizing [BP14]. We can also build what we call tree-based synthesizers, which are based on and look almost identical to the PRFs from [BP14]. We call these synthesizers T_ℓ and $T_{R,\ell}$ for the standard and ring-based versions, respectively. While the tree constructions are a bit too complicated to explain here, we note that we can get parameters asymptotically equivalent to what we have achieved earlier for simple PRFs based on T_2 and $T_{R,2}$ (which we call B^2 and $B^{R,2}$, respectively).

The main advantage of these tree-based synthesizers is that we can potentially build many more interesting PRFs than we otherwise could with the simpler synthesizers. We have yet to fully explore the potential of these synthesizers, but we think that there might be many interesting applications.

Moving to Higher ℓ. In our PRF constructions, ℓ essentially acts as a parallelization parameter– it can be thought of as a 'locality' parameter for the synthesizer. While we do not seem to gain anything in the integer lattice setting from setting ℓ to be anything higher than a constant (other

than the case where $\ell = \lambda$ and we gain key homomorphic properties for certain PRFs), we can achieve some theoretically interesting results from a higher ℓ in the ring setting.

In [BPR12], the authors showed how to construct a PRF in NC^1 using ring LWE. We generalize this PRF with our synthesizer $K_{R,\ell}$ and show two interesting choices of PRF to examine with higher ℓ. We first consider the PRF $P^{R,\lambda^{\frac{1}{\sqrt{\log \lambda}}}}$ which is built using the synthesizer $K_{R,\lambda^{\frac{1}{\sqrt{\log \lambda}}}}$. This PRF has modulus $m^{\omega(1)\left(\lambda^{\frac{1}{\sqrt{\log \lambda}}}\right)}$, which is clearly large but is still smaller than m^{λ^c} for any constant c. The synthesizer construction tree of this PRF also has depth $\sqrt{\log n}$, so we can build this PRF in overall circuit depth of $O\left((\log n)^{\frac{3}{2}}\right)$, giving us a PRF in the unorthodox class $NC^{1.5}$.

We finally consider the PRF $P^{R,\lambda^{\lambda^c}}$ for some constant $c \in (0,1)$ which is based upon the synthesizer $K_{R,\lambda^{\lambda^c}}$. This PRF has modulus $m^{\omega(1)\lambda^c}$ and synthesizer construction with constant depth, meaning that it can be built in NC^1. This PRF is interesting because it is the PRF with the smallest modulus that we can build in NC^1 using our techniques. This lets us build lattice PRFs in NC^1 with any subexponential modulus (assuming λ is polynomial in m), but we still do not know how to break this subexponential barrier (or if it is even possible). The existence of PRFs in NC^1 has many interesting implications in complexity theory [BFKL94], so building them from maximally hard assumptions is an important problem.

Comparison with Previous Work. In table 1 on the adjacent page, we compare our new PRFs with those of the relevant previous works. We borrow the table format from [BP14].

The General View. While our synthesizers look very similar to existing PRF constructions, the PRF constructions themselves can be viewed as generalizations of those in [BPR12], [BLMR13], and [BP14]. In fact, setting $\ell = \lambda$ for the synthesizers S_ℓ, K_ℓ, and T_ℓ result in PRFs that are almost identical to those in in [BLMR13], [BPR12], and [BP14], respectively. However, we do lose the key homomorphic properties of [BLMR13] and [BP14] when we set $\ell < \lambda$.

Concrete Instantiations and Parameters. If we want to instantiate an actual PRF, we need to look beyond the asymptotics. It is relatively straightforward to see that, except for the highly sequential GGM-based and [BP14] sequential constructions, the constructions of PRFs from synthesizers with small values of ℓ and the construction from Banerjee's thesis [Ban15] are the most efficient overall PRF constructions (although the key sizes are larger) for large input lengths: these have at most a small constant (either 2 or 4) times the number of multiplications as the more direct constructions with a substantially smaller modulus. So, we choose to analyze concretely the synthesizer constructions here.

If we fix a particular 'lattice security dimension' m and a particular subexponential parameter (derived from how we set the modulus in any learning with rounding reduction) r_p, we can examine how some of the schemes work practically. We show these concrete metrics in table 2. While we have not implemented these PRFs or closely examined the speed for various multiplication algorithms, it seems like the PRF $P_{R,2}$ is a strong candidate to be the fastest parallel PRF from lattices known today.

Theoretical Implications. A lofty goal in lattice-based cryptography is to build a PRF based on the hardness of LWE with polynomial modulus q. While we obviously do not achieve that in this work, we seemingly make progress towards this goal. In particular, previous lattice-based PRFs typically relied on long subset-products of matrices multiplied by a secret key. In this work, we show that only a 2-subset product of matrices multiplied by a key is generally

Table 1: **Comparison with Previous Work:** The parameters are with respect to PRFs with input length λ and reductions to worst-case lattice problems in dimension m. We let τ denote the exponent of matrix multiplication. Brackets $[\cdot]$ denote parameters of a ring-LWE construction that are better than those of integer lattices (when ring LWE schemes are possible). We ignore constants, lower-order terms, and logarithmic factors. *Parallel Circuit Complexity* refers to the parallel circuit complexity class of the PRF, when applicable. *Parallel Matrix Comp.* refers to the parallel matrix complexity of a PRF in terms of matrix multiplication operations, which is a much better measure of practical parallelizability than circuit depth. PP refers to public parameters, again when applicable. ★ refers to polynomially-sized parameters that are too big to fit nicely in the table (and are relatively unimportant anyway). ✠ refers to parameters that are dependent on a (unspecified) universal hash function.

Reference	**Modulus**	**Parallel Circuit Complexity**	**Parallel Matrix or Ring Comp.**
[BPR12] GGM	$m^{\omega(1)}$	–	λ
[BPR12] synth	$m^{\log \lambda}$	NC^2	$\log \lambda$
[BPR12] direct	m^{λ}	NC^2 $\left[NC^1\right]$	$\log \lambda$
[BLMR13]	m^{λ}	NC^2	$\log \lambda$
[BP14] sequen	$m^{\omega(1)}$	–	λ
[BP14] balanced	$m^{\log \lambda}$	NC^2	$\log \lambda$
[Ban15] synth	$m^{\omega(1)}$	NC^2	$\log \lambda$
[DS15]	$m^{\log \lambda}$	$NC^{1+o(1)}$	$\log \log \lambda$
This Work: F^2	$m^{\omega(1)}$	NC^2	$\log \lambda$
This Work: P^2	$m^{\omega(1)}$	NC^2	$\log \lambda$
This Work: B^2	$m^{\omega(1)}$	NC^2	$\log \lambda$
This Work: $P^{\lambda^{\frac{1}{\sqrt{\log \lambda}}}}$	$m^{\omega(1)\lambda^{\left(\lambda^{\frac{1}{\sqrt{\log \lambda}}}\right)}}$	NC^2 $\left[NC^{1.5}\right]$	$\log \lambda$
This Work: P^{λ^c}	$m^{\omega(1)\lambda^c}$	NC^2 $\left[NC^1\right]$	$\log \lambda$

Reference	**Key Size**	**PP Size**	**Time/Out**	**Out**
[BPR12] GGM	m	m^2 $[m]$	λm $[\lambda]$	m
[BPR12] synth	λm^2 $[\lambda m]$	0 [0]	$\lambda m^{\tau-2}$ $[\lambda]$	m^2 $[m]$
[BPR12] direct	$\lambda^3 m^2$ $\left[\lambda^2 m\right]$	0 [0]	$\lambda^3 m$ $\left[\lambda^2\right]$	λm $[\lambda m]$
[BLMR13]	$\lambda^2 m$	$\lambda^3 m^2$	$\lambda^3 m$	λm
[BP14] sequen	m	m^2 $[m]$	$\lambda m^{\tau-1}$ $[\lambda]$	m
[BP14] balanced	m	m^2 $[m]$	$\lambda m^{\tau-1}$ $[\lambda]$	m
[Ban15] synth	λm^2 $[\lambda m]$	0 [0]	$\lambda m^{\tau-2}$ $[\lambda]$	m^2 $[m]$
[DS15]	m^2 $[m]$	✠	✠	m
This Work: F^2	m^2	λm^2	$\lambda m^{\tau-2}$	m^2
This Work: P^2	λm^2 $[\lambda m]$	m^2 $[m]$	$\lambda m^{\tau-2}$ $[\lambda]$	m^2 $[m]$
This Work: B^2	m^2 $[m]$	λm^2 $[\lambda m]$	$\lambda m^{\tau-2}$ $[\lambda]$	m^2 $[m]$
This Work: $P^{\lambda^{\frac{1}{\sqrt{\log \lambda}}}}$	★	★	★	★
This Work: P^{λ^c}	★	★	★	★

Some Comments: We note that the large keys of some of the PRFs (including ours) can be computed using a lattice-based PRG in practical cases, making this less of an issue in practice. In addition, we note that it is trivial to modify the (standard LWE-based) PRFs from [BP14] to have exactly the same time per output and output sizes as our PRFs F^2, P^2, and B^2 by expanding the secret to be a full-rank matrix rather than a vector. The authors of [BP14] mention this as an optimization in their work.

Table 2: **Practical Comparison with Previous Work:** In this table we consider the practical implementation of the most efficient parallel lattice-based PRFs. We compare our PRFs to that from [Ban15] in terms of concrete efficiency. The parameters are with respect to PRFs with input length λ and reductions from worst-case lattice problems in dimension m. Brackets $[\cdot]$ denote parameters of a ring-LWE construction that are better than those of integer lattices (when ring LWE schemes are possible). For simplicity, we only state higher-order terms (i.e., we ignore polynomial terms when superpolynomial terms exist, and we ignore constants when polynomial terms exist). The term r_p–short for 'rounding parameter'–refers to the (superpolynomially large) value induced by the LWE $\rightarrow$ LWR reduction that we have denoted in table 1 as $m^{\omega(1)}$.

Reference	**Modulus**	**Matrix/Ring Dimension**	**Key Size (Matrices/Ring Elements)**
[Ban15] synth	r_p^2	m	$8\lambda \times \mathbb{Z}_q^{m \times m}[R_q]$
This Work: F^2	r_p	$m \log q$	$4\lambda \times Z_2^{m \log q \times m \log q}, \log \lambda \times \mathbb{Z}_q^{m \log q \times m \log q}$
This Work: P^2	r_p^3	m	$4\lambda \times \mathbb{Z}_p^{m \times m}[R_p], \log \lambda \times \mathbb{Z}_q^{m \times m}[R_q]$

Reference	**Matrix/Ring Multiplies**	**Matrix/Ring Product Computation**
[Ban15] synth	$4(\lambda - 1)$	$\mathcal{U}\left(Z_q^{m \times m}\right) \times \mathcal{U}\left(Z_q^{m \times m}\right)[\mathcal{U}(R_q) \times \mathcal{U}(R_q)]$
This Work: F^2	$2(\lambda - 1)$	$\mathcal{U}\left(\mathbb{Z}_2^{m \log q \times m \log q}\right) \times \mathcal{U}\left(\mathbb{Z}_q^{m \log q \times m \log q}\right)$
This Work: P^2	$2(\lambda - 1)$	$\mathcal{U}\left(\mathbb{Z}_{r_p}^{m \times m}\right) \times \mathcal{U}\left(\mathbb{Z}_q^{m \times m}\right)[\mathcal{U}(R_{rp}) \times \mathcal{U}(R_q)]$

Some Comments: The 'matrix/ring product computation' shows the distributions of what matrices or ringswe are multiplying in the PRFs above. All of the operations are computed modulo q, but, for our PRFs listed here, some of the matrices or rings are nonuniform (and thus can be multiplied more quickly). The fastest PRF is most likely either P^2 for regular LWE or the ring version of P^2 for ring LWE (this is definitely the case if we use fast Fourier transform (FFT) multiplication over field elements, but less clear for other multiplication algorithms–the construction from [Ban15] or F^2 may be faster for asymptotically slow modular multiplication algorithms).

sufficient to build a PRF. This substantially generalizes the requirements seemingly needed to build a lattice-based PRF with polynomial modulus. We hope that this result can be used as a stepping stone towards such PRFs.

1.4 Paper Outline

The rest of the paper proceeds as follows: we begin by defining some basic cryptographic notation and facts about PRFs and lattice problems in section 2. A reader knowledgeable in lattices and PRFs can safely skip this section. In section 3, we define pseudorandom synthesizers and state results from [NR95] on constructions of PRFs from pseudorandom synthesizers. Our definitions are phrased a little differently than those in [NR95], but the meaning is identical.

In section 4 we formally define our first pseudorandom synthesizer S_ℓ which is based on F_{BLMR} and give an overview of the proof of security. We then give formal analysis of the PRF F^2 which we build from S_2 in section 5. In section 6, we offer a formal proof of security for our synthesizer S_ℓ, also including an explanation of the intuition behind the proof.

We define our synthesizers K_ℓ and $K_{R,\ell}$ which are based on the PRFs from [BPR12], in section 7. We build PRFs from these synthesizers in 8 and follow this up with the security discussion in section 9. We analyze PRFs built from higher values of ℓ with these synthesizers in section 10, which enable us to achieve more parallelism at the expense of a higher modulus.

We finally discuss our synthesizer constructions based on [BP14], which we call T_ℓ and $T_{R,\ell}$, in section 11. We build PRFs from these synthesizers in section 13 and then discuss the security of these synthesizers in section 12.

Finally, in section 14 we conclude and state what we consider are interesting and important open problems in the area.

2 Preliminaries

We start by discussing some basic background material for the paper. A reader who is familiar with the basic cryptographic concepts in each subsection can safely skip the respective subsections.

2.1 Notation

For a random variable X we denote by $x \leftarrow X$ the process of sampling a value x according to the distribution of X. Similarly, for a finite set S we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S. We sometimes also use $\mathcal{U}(S)$ to denote the uniform distribution over a set S. We typically use bold lowercase letters (i.e. $\mathbf{a}$) to denote vectors and bold uppercase letters (i.e. $\mathbf{A}$) to denote matrices.

For two bit-strings x and y (or vectors $\mathbf{x}$ and $\mathbf{y}$) we denote by $x \| y$ their concatenation. A non-negative function $f : \mathbb{N} \to \mathbb{R}$ is negligible if it vanishes faster than any inverse polynomial. We denote by $\mathrm{Rk}_i(\mathbb{Z}_p^{a \times b})$ the set of all $a \times b$ matrices over $\mathbb{Z}_p$ of rank i.

Binary Trees. We use $\mathcal{T}$ to denote a binary tree. For any given binary tree $\mathcal{T}$, we let $\mathcal{T}_l$ denote the left subtree of the root note and $\mathcal{T}_r$ denote the right subtree of the root node. Note that these may be empty. We let $|\mathcal{T}|$ denote the number of leaves of $\mathcal{T}$.

In addition, as in [BP14], we define the *expansion* of a tree $e(\mathcal{T})$ recursively in the following way:

$$e(\mathcal{T}) = \left\{ \begin{array}{cc} 0 & \text{if } |\mathcal{T}| = 1 \\ max\{e(\mathcal{T}_l) + 1, e(\mathcal{T}_r)\} & \text{otherwise} \end{array} \right\} \tag{2.1}$$

Note that this is just the 'left depth' of the tree $\mathcal{T}$.

Rounding. We use $\lfloor \cdot \rfloor$ to denote rounding a real number to the largest integer which does not exceed it. For integers q and p where $q \geq p \geq 2$, we define the function $\lceil \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lceil x \rfloor_p = i$, where $i \cdot \lfloor q/p \rfloor$ is the largest multiple of $\lfloor q/p \rfloor$ that does not exceed x. For a vector $\mathbf{v} \in \mathbb{Z}_q^m$, we define $\lceil \mathbf{v} \rfloor_p$ as the vector in $\mathbb{Z}_p^m$ obtained by rounding each coordinate of the vector individually. A probability distribution χ over $\mathbb{R}$ is said to be *B-bounded* if it holds that $\Pr_{x \leftarrow \chi}[|x| > B]$ is negligible in the security parameter.

Rings. Throughout this paper, we let R denote the cyclotomic polynomial ring $R \stackrel{\mathsf{def}}{=} \mathbb{Z}[x] / (x^n + 1)$ where n is a power of 2. Many properties of our cryptographic constructions will depend on the structure of these cyclotomic rings, so it is important that we only use this kind of ring. For any integer q, we define the quotient ring $R_q \stackrel{\mathsf{def}}{=} R/qR$. We note that an element of R can be represented as a polynomial in x of degree $n-1$ with integer coefficients.

While there are many more interesting rings, we work with these special cyclotomics due to performance reasons. Our results will generally hold for many other choices of rings, but the efficiency of our scehemes will suffer.

2.2 Pseudorandomness

We next review the definition of pseudorandom generators and pseudorandom functions [GGM84]. We start by discussing pseudorandom generators.

Pseudorandom Function Informally, a pseudorandom function is an efficiently computable function such that no efficient adversary can distinguish the function from a truly random function given only black-box access.

More precisely, a PRF is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ where $\mathcal{K}$ is called the key space, $\mathcal{X}$ is called the domain, and $\mathcal{Y}$ is called the range. In this paper, we sometimes allow the PRF to take additional public parameters pp and use $F_{pp} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ to denote such a PRF. Security for a PRF is defined using two experiments between a challenger and an adversary $\mathcal{A}$. For $b \in \{0,1\}$ the challenger in Exp_b works as follows.

1. When $b = 0$ the challenger chooses a random key $k \in K$ and sets $f(\cdot) \stackrel{\mathsf{def}}{=} F(k, \cdot)$.
2. When $b = 1$ the challenger chooses a random function $f : \mathcal{X} \to \mathcal{Y}$.
3. The adversary (adaptively) sends input queries $x_1, \ldots, x_q$ in $\mathcal{X}$ to the challenger and the challenger responds with $f(x_1), \ldots, f(x_q)$. Eventually the adversary outputs a bit $b' \in \{0,1\}$.

For $b \in \{0,1\}$ let W_b be the probability that $\mathcal{A}$ outputs 1 in Exp_b.

Definition 2.1. *A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is secure if for all efficient adversaries $\mathcal{A}$ the quantity*

$$\mathsf{PRF}_{adv}[\mathcal{A}, F] \stackrel{\mathsf{def}}{=} |W_0 - W_1|$$

is negligible.

As usual, we can make the terms "efficient" and "negligible" precise using asymptotic notation by equating efficient with probabilistic polynomial time and equating negligible with functions smaller than all inverse polynomials. Here, we use non-asymptotic language to simplify the notation.

2.3 Lattices

We now review background material on lattices. Let q, n, and m be positive integers, and let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix. We let $\Lambda_q^{\perp}(\mathbf{A})$ denote the lattice spanned by all $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q$. For a vector $\mathbf{u} \in \mathbb{Z}_q^n$, we generalize this and let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the set of all vectors such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{u}$. Note that this is a coset of $\Lambda_q^{\perp}(\mathbf{A})$.

Discrete Gaussians. We borrow the elegant presentation style of [GPV08]. For any $s > 0$ define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter $\mathbf{s}$:

$$\forall x \in \mathbb{R}^n, \quad \rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi ||\mathbf{x}-\mathbf{c}||^2/s^2}$$

We sometimes omit the subscripts s and $\mathbf{c}$ in the case that they are 1 and 0, respectively.

For any $\mathbf{c} \in \mathbb{R}^n$, real $s > 0$, and n-dimensional lattice Λ, we define the discrete Gaussian distribution over Λ as:

$$\forall \mathbf{x} \in \Lambda, \quad \mathcal{D}_{\Lambda,\mathbf{s},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\mathbf{s},\mathbf{c}}(\mathbf{x})}{\rho_{\mathbf{s},\mathbf{c}}(\Lambda)}$$

Smoothing Parameter. In [MR04], Micciancio and Regev defined the smoothing parameter: for any n-dimensional lattice Λ and positive real ε, the *smoothing paramter* $\eta_\varepsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.

The Gadget Matrix. In [MP12], Micciancio and Peikert invented the gadget vector $\mathbf{g}$ and the gadget matrix $\mathbf{G}$. Let $q = 2^c$ be some integer, and let $c \in \mathbb{Z} = \lceil \log_2 q \rceil$. Recall from [MP12] that the vector $\mathbf{g} \in \mathbb{Z}_q^c$ is defined to be a primitive vector such that $\mathbf{g}_i = 2^{i-1}$, or $\mathbf{g}^\intercal = \left[1|2|4|....|2^{c-1}\right]$. Additionally recall the matrix $\mathbf{S} \in \mathbb{Z}_q^{c \times c}$ was defined such that entries of the form $\mathbf{S}_{i,i} = 2$, $\mathbf{S}_{i+1,i} = -1$, and all other entries are zero. In picture form, we have

$$\begin{bmatrix} 1 & 2 & 4 & .... & 2^{c-1} \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 & ... & 0 \\ -1 & 2 & 0 & ... & 0 \\ 0 & -1 & 2 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & .... & 0 \end{bmatrix} \mod q$$

If q is not a power of 2, we can modify the last column to be the binary bit decomposition of q. We also will need the standard tensor forms of $\mathbf{g}$ and $\mathbf{S}$ as well. We define the matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nc}$ to be $\mathbf{G} = g^\intercal \otimes \mathbf{I}_n$ and the matrix $\mathbf{T} \in \mathbb{Z}_q^{nc \times nc}$ to be $\mathbf{T} = \mathbf{S} \otimes \mathbf{I}_n$. Pictorially, we have something like the following (dimensions not to scale):

$$\begin{bmatrix} -\mathbf{g}^\intercal- & 0 & 0 & ... & 0 \\ 0 & -\mathbf{g}^\intercal- & 0 & ... & 0 \\ 0 & 0 & -\mathbf{g}^\intercal- & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & -\mathbf{g}^\intercal- \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S} & 0 & 0 & ... & 0 \\ 0 & \mathbf{S} & 0 & ... & 0 \\ 0 & 0 & \mathbf{S} & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \mathbf{S} \end{bmatrix} = \mathbf{0}^{nc \times c} \mod q$$

We define the deterministic 'binary decomposition' function $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}_2^{\lceil \log q \rceil}$ in the following way: let the ith entry of the output of $\mathbf{g}^{-1}(a)$ for some a be the ith bit of a. In other words, if we let the vector $\mathbf{x} \in \mathbb{Z}_2^{\lceil \log q \rceil} = \mathbf{g}^{-1}(a)$, then $a = \sum_{i=0}^{\lceil \log q \rceil - 1} \mathbf{x}_i 2^i$.

In addition, for all vectors and matrices over $\mathbb{Z}_q$, we define the function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \to \mathbb{Z}_2^{n\lceil \log q \rceil \times m}$ to be the function $\mathbf{g}^{-1}$ applied entry by entry.

2.4 Learning With Errors

Learning with errors (LWE) assumption. The LWE problem was introduced by Regev [Reg05] who showed that solving the LWE problem *on average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition 2.2. ***Learning with Errors Problem (LWE)****: Consider integers n and q, some distribution ψ over $\mathbb{Z}_q$, and distributions $\mathcal{K}$ and $\mathcal{T}$, both over $\mathbb{Z}_q^n$.*

A $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem instance consists of access to an unspecified challenge oracle $\mathcal{O}^{LWE}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{LWE}$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{LWE}$, whose behaviors are respectively as follows:

$\mathcal{O}_{\mathbf{s}}^{LWE}$**:** *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{a}_i^{\intercal} \cdot \mathbf{s} + \delta_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$, $\delta_i \in \mathbb{Z}_q$ consists of a fresh sample from ψ, and $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$.*

$\mathcal{O}_{\$}^{LWE}$**:** *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{r}_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$ and $\mathbf{r}_i$ is a uniform random sample from $\mathbb{Z}_q$.*

The $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem allows repeated queries to the challenge oracle $\mathcal{O}^{LWE}$. We say that an algorithm $\mathcal{A}$ decides the $(q, n, \psi, \mathcal{K}, \mathcal{T})$-LWE problem if

$$\mathbf{Adv}^{LWE}[\mathcal{A}] \stackrel{\mathsf{def}}{=} \left| \Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{LWE}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{LWE}} = 1] \right|$$

is non-negligible for a $\mathbf{s}$ selected appropriately at random from $\mathcal{K}$.

Let $\mathcal{U}_q^n$ be the uniform distribution over vectors in $\mathbb{Z}_q^n$. As we alluded to before, Regev [Reg05] shows that for a certain B-bounded noise distribution χ (for some $B = poly(n)$, for n polynomial in some security parameter, and a sufficiently large q, the $\left(q, n, \chi, \mathcal{U}_q^n, \mathcal{U}_q^n\right)$-LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction (see also [Pei09, BLP+13]). Similarly, the noise distribution $\boldsymbol{\chi}$ can be a simple low-norm distribution [MP13] if the number of samples received by an adversary is small (which is unfortunately not the case in known PRF reductions).

Ring LWE. In [LPR10], the authors developed a ring analogue of the learning with errors problem. They showed that solving the Ring LWE problem *on average* for certain choices of rings is as hard as (quantumly) solving several standard lattice problems *in the worst case* over *ideal lattices*. This paper has been generalized by [PRSD17] to include a wider variety of rings, but in this work we stick with a very specific class of rings as we mentioned earlier in this section. Below we define the ring LWE problem formally.

Definition 2.3. ***Ring Learning with Errors Problem (RLWE)****: Consider integers n and q, a polynomial ring R of maximum degree n, some distribution $\boldsymbol{\psi}$ over R, and distributions $\mathcal{K}$ and $\mathcal{T}$, both over R_q.*

A $(R, q, n, \boldsymbol{\psi}, \mathcal{K}, \mathcal{T})$-RLWE problem instance consists of access to an unspecified challenge oracle $\mathcal{O}^{RLWE}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{RLWE}$ carrying some constant random secret key $\mathbf{s} \in R_q$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{RLWE}$, whose behaviors are respectively as follows:

$\mathcal{O}_{\mathbf{s}}^{RLWE}$**:** *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{a}_i^{\intercal} \cdot \mathbf{s} + \boldsymbol{\delta}_i\right) \in R_q \times R_q$, where $\mathbf{s} \in R_q$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$, $\boldsymbol{\delta}_i \in R$ consists of a fresh sample from $\boldsymbol{\psi}$, and $\mathbf{a}_i \in R_q$ is sampled at random from $\mathcal{T}$.*

$\mathcal{O}_{\$}^{RLWE}$**:** *Outputs samples of the form* $\left(\mathbf{a}_i, \mathbf{r}_i\right) \in R_q \times R_q$*, where* $\mathbf{a}_i \in R_q$ *is sampled at random from* $\mathcal{T}$ *and* $\mathbf{r}_i$ *is a uniform random sample from* R_q*.*

The $(R, q, n, \boldsymbol{\psi}, \mathcal{K}, \mathcal{T})$*-RLWE problem allows repeated queries to the challenge oracle* $\mathcal{O}^{RLWE}$*. We say that an algorithm* $\mathcal{A}$ *decides the* $(R, q, n, \boldsymbol{\psi}, \mathcal{K}, \mathcal{T})$*-RLWE problem if*

$$\mathbf{Adv}^{RLWE}\left[\mathcal{A}\right] \stackrel{\mathsf{def}}{=} \left| \Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{RLWE}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{RLWE}} = 1] \right|$$

is non-negligible for a $\mathbf{s}$ *selected appropriately at random from* $\mathcal{K}$*.*

LWE with Key Sampled from the Noise Distribution. In [ACPS09], these hardness results were extended to show that the LWE secret $\mathbf{s}$ can be sampled from a low norm distribution (in particular, from the noise distribution $\boldsymbol{\chi}$) and the resulting problem is as hard as the basic LWE problem. We will need this fact for our reductions, so we state this as a theorem here.

Theorem 2.4. *Let* n *and* q *be integers, and let* ψ *be some noise distribution over* $\mathbb{Z}$*. Let the distribution* $\boldsymbol{\psi} \in \mathbb{Z}^n$ *consist of the concatenation of* n *samples from* ψ*.*

Any adversary that can solve the $\left(q, n, \psi, \mathcal{U}_q^n, \boldsymbol{\psi}\right)$*-LWE problem can be used to solve the* $\left(q, n, \psi, \mathcal{U}_q^n, \mathcal{U}_q^n\right)$*-LWE problem.*

We note that the $(q, n, \psi, \mathcal{U}_q^n, \mathcal{U}_q^n)$-LWE problem is the standard version of the LWE problem. This theorem immediately extends to rings as well. We state this corollary next.

Theorem 2.5. *Let* n *and* q *be integers, let* R *be a polynomial ring* R *of maximum degree* n*, and let* $\boldsymbol{\psi}$ *be some noise distribution over* R*.*

Any adversary that can solve the $\left(R, q, n, \boldsymbol{\psi}, \mathcal{U}\left(R\right), \boldsymbol{\psi}\right)$*-RLWE problem can be used to solve the* $\left(R, q, n, \boldsymbol{\psi}, \mathcal{U}\left(R\right), \mathcal{U}\left(R\right)\right)$*-RLWE problem.*

We again note that the $\left(R, q, n, \psi, \mathcal{U}\left(R_q\right), \mathcal{U}\left(R_q\right)\right)$-RLWE problem is the standard version of the RLWE problem.

Non-uniform LWE. In [BLMR13], the authors introduced a variant of the learning with errors (LWE) problem in which the the rows of $\mathbf{A}$ (i.e., the LWE samples) are sampled from a *non-uniform* distribution $\boldsymbol{\eta}$ over $\mathbb{Z}_q^n$. They called this variant of LWE *Non-uniform Learning with Errors*, or NLWE for short, and showed that for suitable parameters it is as hard as the basic LWE problem. While the authors of [BLMR13] show that LWE instances with samples drawn from a wide range of nonuniform distributions are hard, we will use the fact that LWE is hard with uniform binary samples for appropriate parameter choices. We state this in a theorem below.

Theorem 2.6. *Let* $q = q(n)$ *be an integer such that* $\frac{2^{\lceil \log q \rceil} - q}{q}$ *is negligible (i.e.,* q *is close to a power of* 2*). Let* $m = n\lceil \log_2 q \rceil$ *be an integer. Let* $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ *denote the uniform distribution of binary vectors in dimension* m*, let* $\mathcal{U}_q^m$ *denote the uniform distribution of all vectors over* $\mathbb{Z}_q^m$*, and let* ψ *be a* B*-bounded noise distribution over* $\mathbb{Z}$*.*

Any adversary that can solve the $(q, m, \psi, \boldsymbol{\eta}_{\mathsf{Bin}(m)}, \mathcal{U}_q^m)$-LWE *problem with advantage* ε *can be used to solve the* $(q, n, \psi, \mathcal{U}_q^n, \mathcal{U}_q^n)$-LWE *problem with advantage* ε*.*

Unfortunately this result does not apply to rings as well–ring LWE with low-norm samples is trivially insecure.

LWE with Parallel Instances. We note that, by a simple hybrid argument, we can show that LWE is hard if the key is a matrix of size $n \times m$ for some polynomially sized integer m (provided the columns of the key matrix are sampled independently at random from the keyspace $\mathcal{K}$). We note that we lose a factor of m in the distinguishing advantage (which is typically not an issue since m is a polynomial).

In addition, for any polynomially-bounded integer Q, it is the case that Q instances of LWE (with keys as vectors or matrices) are also indistinguishable from Q random instances, and this also can be shown using an easy hybrid argument. As expected, we lose a factor of Q in an adversary's distinguishing advantage. These facts have appeared in a number of lattice-based papers, typically in a folklore-styled presentation, including [BLMR13]. We sometimes use $\mathrm{LWE}^{x,y}$ to denote the problem of distinguishing y parallel LWE instances containing keys with x vectors and note that an adversary that can distinguish such an instance from random with advantage ε can distinguish a regular LWE instance from random with advantage $\frac{\varepsilon}{mQ}$.

Learning with Rounding. Banerjee, Peikert, and Rosen[BPR12] consider a related problem, denoted the "learning with *rounding*" (LWR) problem (recall the notation $\lceil \cdot \rfloor_p$ from earlier). LWR can be viewed as a deterministic version of LWE. Our results do not explicitly use the hardness of this problem (we reduce from LWE and RLWE instead) but use many of the core ideas of the problem in their proofs.

Definition 2.7. ***Learning with Rounding Problem (LWR)****: Consider integers n, p, and q such that $q \geq p$, and distributions $\mathcal{K}$ and $\mathcal{T}$, both over $\mathbb{Z}_q^n$.*

A $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem instance consists of access to an unspecified challenge oracle $\mathcal{O}^{LWR}$, being, either, a noisy pseudorandom sampler $\mathcal{O}_{\mathbf{s}}^{LWR}$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$ sampled from the distribution $\mathcal{K}$, or, a truly random sampler $\mathcal{O}_{\$}^{LWR}$, whose behaviors are respectively as follows:

- $\mathcal{O}_{\mathbf{s}}^{LWR}$**:** *Outputs samples of the form $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^{\mathsf{T}} \cdot \mathbf{s} \rfloor_p\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is a persistent value invariant across invocations sampled by querying the distribution $\mathcal{K}$ and $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$.*
- $\mathcal{O}_{\$}^{LWR}$**:** *Outputs samples of the form $\left(\mathbf{a}_i, \mathbf{r}_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$, where $\mathbf{a}_i \in \mathbb{Z}_q^n$ is sampled at random from $\mathcal{T}$ and $\mathbf{r}_i$ is a uniform random sample from $\mathbb{Z}_p$.*

The $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem allows repeated queries to the challenge oracle $\mathcal{O}^{LWR}$. We say that an algorithm $\mathcal{A}$ decides the $(q, p, n, \mathcal{K}, \mathcal{T})$-LWR problem if

$$\mathbf{Adv}^{LWR}[\mathcal{A}] \stackrel{\mathsf{def}}{=} \left| \Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^{LWR}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\$}^{LWR}} = 1] \right|$$

is non-negligible for a $\mathbf{s}$ selected appropriately at random from $\mathcal{K}$.

We note that it is also possible to define a ring analogue of LWR, but we omit this in this work.

3 Pseudorandom Synthesizers

In this section we introduce pseudorandom synthesizers. Pseudorandom synthesizers were invented by Naor and Reingold in their seminal work [NR95]. Since previous general-purpose PRF constructions were entirely sequential [GGM84] (i.e. had circuit depth at least linear in the number of input bits), which was both theoretically and practically inefficient, Naor and Reingold developed a new technique for building highly parallel PRF constructions which they called pseudorandom synthesizers.

We spend a little bit more time on this than usual because we present the material in a different way than [NR95] or Omer Reingold's thesis [Rei]. Rather than using synthesizer ensembles, we opt for the more modern game-based definitions where keys are chosen randomly (rather than functions are selected randomly from an ensemble). This means that, in addition to a traditional synthesizier, we need to define a keyed synthesizer as well. The definitional changes require us to change the way the definitions of synthesizers are presented, but we note that the content remains exactly the same.

We additionally generalize some of the definitions to cover alternative constructions that are mentioned in [NR95] (and shown to be secure), but not covered by the main definition. We start by defining a basic pseudorandom synthesizer.

Pseudorandom Synthesizer Basics. A pseudorandom synthesizer is, in rough terms, a two-input function $S(\cdot,\cdot)$ parameterized by two integers m and n such that on random inputs $(x_1, ..., x_m) \in \mathcal{X}$ and $(w_1, ..., w_n) \in \mathcal{W}$, the matrix $\mathbf{M}$ of all mn values of $S(x_i, w_j) = \mathbf{M}_{ij}$ is indistinguishable from random.

Let m and n be integers. In precise terms, a pseudorandom synthesizer is an efficiently computable function $S : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ parameterized by m and n, where $\mathcal{X}$ and $\mathcal{W}$ are the two input domains and $\mathcal{Y}$ is the range. In this paper, we sometimes allow the synthesizer to take additional public parameters pp and use $S_{pp} : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ to denote such a synthesizer.

Security for a synthesizer is defined using two experiments between a challenger and an adversary $\mathcal{A}$. For $b \in \{0, 1\}$ the challenger in Exp_b works as follows.

1. When $b = 0$ the challenger sets $f(\cdot,\cdot) \stackrel{\text{def}}{=} S(\cdot,\cdot)$.
2. When $b = 1$ the challenger chooses a random function $f : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$.
3. The challenger samples input values $(x_1, ..., x_m) \leftarrow \mathcal{X}$ and $(w_1, ..., w_n) \leftarrow \mathcal{W}$ and sends the values $f(x_i, w_j)$ for all $i \in [1, m]$ and $j \in [1, n]$ to the adversary. Eventually the adversary outputs a bit $b' \in \{0, 1\}$.

For $b \in \{0, 1\}$ let W_b be the probability that $\mathcal{A}$ outputs 1 in Exp_b.

Definition 3.1. *A synthesizer $S : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ is secure if for all efficient adversaries $\mathcal{A}$ the quantity*

$$\mathsf{SYNTH}_{adv}[\mathcal{A}, F] \stackrel{\text{def}}{=} |W_0 - W_1|$$

is negligible.

The above definition is what most papers that present pseudorandom synthesizers use. However, as we earlier alluded, the work of [NR95] allows for substantially more generality. First, we note that we can also use *keyed* synthesizers. A keyed pseudorandom synthesizer (KPS) is, in rough terms, a *keyed* two-input function $S(k, \cdot, \cdot)$ parameterized by two integers m and n such that on sets of random inputs $(x_1, ..., x_m) \in \mathcal{X}$ and $(w_1, ..., w_n) \in \mathcal{W}$, the matrix $\mathbf{M}$ of all mn values of $S(k, x_i, w_j) = \mathbf{M}_{ij}$ is indistinguishable from random. We note that on successive queries to the KPS, the same key is used but new input values x_i and y_i are chosen.

Additionally, we note that it is not necessary that our synthesizer S be a function with two inputs and one output. S could have three (or more) inputs, as long as we can still prove security. Once again, Naor and Reingold show a proof of security for this case as well in [NR95]. Thus, we overload S so that it can take more than two inputs. We next present a modified definition of a pseudorandom synthesizer that takes all of these extra considerations into account.

Keyed Pseudorandom Synthesizer Definition. Let ℓ be an integer, and let $n_1, ..., n_\ell$ be integers as well. In precise terms, a pseudorandom synthesizer is an efficiently computable function $S : \mathcal{K} \times \mathcal{X}_1 \times ... \times \mathcal{X}_\ell \to \mathcal{Y}$ parameterized by ℓ, n_1, ... , n_ℓ where $\mathcal{K}$ is the keyspace, $\mathcal{X}_1$, ... , $\mathcal{X}_\ell$ are the ℓ input domains and $\mathcal{Y}$ is the range. In this paper, we sometimes allow the synthesizer to take additional public parameters pp and use $S_{pp} : \mathcal{K} \times \mathcal{X}_1 \times ... \times \mathcal{X}_\ell \to \mathcal{Y}$ to denote such a keyed synthesizer.

Security for a synthesizer is defined using two experiments between a challenger and an adversary $\mathcal{A}$. For $b \in \{0, 1\}$ the challenger in Exp_b works as follows.

1. When $b = 0$ the challenger selects a random key $k \leftarrow \mathcal{K}$ and sets $f(\cdot, \cdot) \stackrel{\text{def}}{=} S(k, \cdot, ..., \cdot)$.
2. When $b = 1$ the challenger chooses a random function $f : \mathcal{X}_1 \times ... \times \mathcal{X}_\ell \to \mathcal{Y}$.
3. The challenger samples input values $(x_{1,1}, ..., x_{1,n}) \leftarrow \mathcal{X}_1$, $(x_{2,1}, ..., x_{2,n}) \leftarrow \mathcal{X}_2$, ... , $(x_{\ell,1}, ..., x_{\ell,n}) \leftarrow \mathcal{X}_\ell$ and sends the values $f(x_{1,i_1}, x_{2,i_2}..., x_{\ell,i_\ell})$ for all $i_1 \in [1, n_1]$, ... , $i_\ell \in [1, n_\ell]$ to the adversary. If the number of possible values of f is superpolynomial, the adversary is allowed to adaptively query f on inputs of the form $(i_1, ..., i_\ell)$ of its choice. The challenger repeats this process an arbitrary polynomial number of times. Eventually the adversary outputs a bit $b' \in \{0, 1\}$.

For $b \in \{0, 1\}$ let W_b be the probability that $\mathcal{A}$ outputs 1 in Exp_b.

Definition 3.2. *A synthesizer $S : \mathcal{K} \times \mathcal{X}_1 \times ... \times \mathcal{X}_\ell \to \mathcal{Y}$ is secure if for all efficient adversaries $\mathcal{A}$ the quantity*

$$\mathsf{SYNTH}_{adv}[\mathcal{A}, F] \stackrel{\text{def}}{=} |W_0 - W_1|$$

is negligible.

3.1 Building PRFs from Synthesizers

In this section we explain how to build pseudorandom functions from synthesizers using the main theorem from [NR95]. We use different terminology but the content of the theorem statement remains the same.

ℓ-Admissible Synthesizers. In order to build synthesizers that combine ℓ inputs into one, we need to make sure that the overall bit length of our input is appropriate for our synthesizer length ℓ. To see how this might go wrong, suppose we are trying to construct a 4-bit PRF from a 3-way synthesizer. If we combine inputs 1, 2, and 3, we will get another input $1'$. But we will only have input 4 to combine with it. If our synthesizer only works on three inputs (and not two–some synthesizers might work on both two or three inputs), we will be stuck and unable to finish our PRF! The authors of [NR95] do not explicitly mention such an idea, but it is implicit in their work.

Definition 3.3. *We say that a number λ is ℓ-admissible if the following procedure outputs one:*

1. *While $\lambda \geq \ell$:*
   (a) *Write $\lambda = \ell k + r$ where $r \in [0, \ell]$*
   (b) *Set $\lambda = \frac{\lambda - r}{\ell} + r$.*
2. *Output λ*

We note that any λ is 2-admissible. For larger values of ℓ, the situation is slightly more complicated. In practice, this admissibility fact won't be too much of an issue–we can just pick λ to be a multiple of ℓ, for instance–but we need it for our synthesizer definition to be complete.

Definition 3.4. ***Squeeze Function*** *SQ^{ℓ}_{sk}: Let X be some set and sk some secret key. Let k and ℓ be an integers, and let $k \mod \ell = r$. For every function $S^{\ell}_{sk} : \mathcal{X}^{\ell} \to \mathcal{X}$, and every sequence of inputs $L = \{x_1, ..., x_k\}$ where $x_i \in X$ we define the squeeze $SQ_{sk}(L)$ to be the sequence $L' = \left\{x'_1, ..., x'_{\lfloor \frac{k}{\ell} \rfloor}, x'_{\lfloor \frac{k}{\ell} \rfloor + 1}, ..., x'_{\lfloor \frac{k}{\ell} \rfloor + r}\right\}$ where $x'_i = S_{sk}\left(x_{\ell i-(\ell-1)}, ...x_{\ell i-1}, x_{\ell i}\right)$ for $i \leq \lfloor \frac{k}{\ell} \rfloor$, and if $k \neq 0 \mod \ell$, then for each $i \geq \lfloor \frac{k}{\ell} \rfloor$ we set $x'_i = x_{(\ell-1)\lfloor \frac{k}{\ell} \rfloor + i}$.*

Definition 3.5. *Let ℓ be an integer, and let λ be an ℓ-admissible integer. Let $S^{\ell}_{sk} : \mathcal{X}^{\ell} \to \mathcal{X}$ be a family of keyed pseudrandom synthesizers with key generation algorithm $KeyGen$. We define a pseudorandom function F in the following way:*
Key Generation:

1. *For $j \in [1, ..., \lceil \log_{\ell} \lambda \rceil]$, sample $sk_i \leftarrow KeyGen$.*
2. *For $i \in [1, ..., \lambda]$ and $b \in [0, 1]$, sample $x_{i,b} \leftarrow \mathcal{X}$.*

Evaluation: *For some bit string $\mathbf{i} \in \mathbb{Z}_2^{\Lambda} = \{i_1 i_2 ... i_{\lambda}\}$ we have*

$$F_{pp}(\mathbf{i}) = SQ^{\ell}_{sk_1}\left(SQ^{\ell}_{sk_2}\left(...SQ^{\ell}_{sk_{\lceil \log_{\ell} \lambda \rceil}}\{x_{1,i_1}, x_{2,i_2}, ..., x_{\lambda,i_{\lambda}}\}...\right)\right)$$

We next state the main theorem from [NR95], which proves that any adversary that can distinguish the PRF construction in definition 3.5 from random can be used to distinguish the output of the synthesizer S^{ℓ}_{sk} from random. We paraphrase the theorem slightly to accomodate our definitions, which, as we have mentioned numerous times, are slightly different from those in [NR95].

Theorem 3.6. *Let ℓ and λ be integers. Let S^{ℓ}_{sk} be a pseudorandom synthesizer as defined in definition 3.2, and let F_{pp} be the function defined in definition 3.5. Any adversary that can distinguish F_{pp} from a truly random function with advantage ε can be used to distinguish S^{ℓ}_{sk} from random with advantage $\frac{\varepsilon}{\log_{\ell} \lambda}$.*

Proof. This theorem is almost exactly Theorem 5.1 of [NR95] and the proof can be found there. ∎

4 Sample Subset-Product Pseudorandom Synthesizer

In this section we define a new pseudorandom synthesizer based on the LWE assumption and prove that it is secure. Our synthesizer S_{ℓ} very closely resembles the PRF from [BLMR13]. We choose to present this synthesizer first because it is the simplest and most intuitive construction and has the easiest composition into PRFs.

4.1 Synthesizer Definition

Definition 4.1. *Let m, q, p, and ℓ be integers such that $p \leq q$ and $\frac{2^{\lceil \log q \rceil} - q}{q}$ is negligible. Let $k_1, ..., k_{\ell} \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_i]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_2^{m \times m}$ be a uniformly distributed binary matrix. Let $\mathbf{S} \in \mathbb{Z}_q^{m \times m}$ be a matrix sampled uniformly at random.*

We define the synthesizer $S_\ell : \mathbb{Z}_q^{m\times m} \times \left[\left(\mathbb{Z}_2^{m\times m}\right)^{k_1} \times ... \times \left(\mathbb{Z}_2^{m\times m}\right)^{k_\ell}\right] \to \mathbb{Z}_p^{k_1 m\times ...\times k_\ell m}$ *in the following way: for each* $m \times m$ *block of output of* S_ℓ*, define*

$$S_\ell\left(x_1, ...x_\ell\right) \stackrel{\text{def}}{=} \left\lfloor \left[\prod_{i=1}^{\ell} \mathbf{A}_{i,x_i}\right] \mathbf{S} \right\rceil_p \tag{4.1}$$

where $x_i \in [1, k_i]$.

We now offer some comments on our synthezier S_{LWE}. First, note that as long as p is even, we can 'chain' this synthesizer. In other words, if this is the case, we can modify the output of each (i, j)-block of the synthesizer $S_{\text{LWE}(i,j)}$ to be a random matrix over $\mathbb{Z}_2^{m\times m}$ by just computing the output modulo two. Later in the paper (when we select parameters and analyze the overall PRF's performance) we will comment more on this.

We defer the security proof of this construction to section 6. In section 6, we explain the intuition behind the proof and then work through it formally. Unfortunately, it is different enough from known security proofs that we thought it needed to be explained fully. While the proof of [BP14] could be applied obliquely to get the same result, the implied parameters are not quite as good as we can achieve with a direct proof. Thus, we elect to present a straightforward proof here.

5 Constructions of PRFs from S_ℓ

In this section we show how our synthesizer S_ℓ can be used to build PRFs. We also show some optimizations that we can achieve by slightly modifying the overall synthesizer construction (in a way that doesn't affect security).

We start by stating an overall theorem about the security of PRFs constructed from our pseudorandom synthesizers using the synthesizer construction of [NR95]. This theorem follows almost immediately from applying the synthesizer construction security theorem of [NR95] as we stated in theorem 3.6 to our theorem 6.8proving our synthesizers S_ℓ secure.

Theorem 5.1. *Let* m*,* n*,* q*,* p*,* λ*, and* ℓ *be integers. In words,* m *will be our lattice dimension,* n *will be the dimension of the LWE problem we reduce to,* q *is our modulus,* p *is our rounding parameter,* λ *is our PRF length, and* ℓ *is our synthesizer parameter. Let* $\psi \in \mathbb{Z}$ *be a* B*-bounded noise distribution. We additionally require that* $q \geq 2m^{\ell+\omega(1)}Bp$ *and* $\frac{2^k-q}{q}$ *is negligible for some integer* k*. Let* $p = 2$*.*

Let the PRF F^ℓ *defined by applying the synthesizer construction defined in definition 3.5 to the synthesizer* S_ℓ *defined in definition 4.1. Let* Q *be the number of queries an adversary makes to* F^ℓ*.*

Any adversary that can distinguish F^ℓ *from random as defined in definition 2.1with advantage* ε *can be used to solve the* $\left(q, n, \psi, \mathcal{U}_q^n, U_q^n\right)$-LWE *problem (the standard version of LWE) with advantage* $\frac{\varepsilon}{2mQ\log\lambda}$*.*

As long as we follow the parameter choices implied by theorem 5.1, we can build PRFs from S_ℓ for any choice of ℓ. In fact, we could mix and match different values of ℓ in a single PRF, but we do not see any logical reason to do so (perhaps odd hardware constraints could make such a thing useful).

In the rest of the section, we examine the PRF F^2, which we build from S_2.

5.1 The Synthesizer S_2

We start by analyzing the simplest, and yet one of the most efficient, synthesizers: S_2. From theorem 5.1, we know that F^2–the PRF built using S_2–is secure as long as $q \geq 2m^{2+\omega(1)}Bp = 2m^{\omega(1)}Bp$ and $\frac{2^k-q}{q}$ is negligible for some integer k. We let $p = 2$.

Let's evaluate our PRF construction F^2. We start by noting that, due to our choice of parameters, $q \geq 4m^{2+\omega(1)}Bp = 2m^{\omega(1)}Bp$, and thus

$$\log q = O\left(\omega\left(1\right)\left(\log m\right)\right) = O\left(\omega\left(1\right)\left(\log\left(n\log q\right)\right)\right) =$$

$$O\left(\omega\left(1\right)\left(\log\left(n\right)\log\log\left(q\right)\right)\right) = O\left(\omega\left(1\right)\log n\right)$$

Efficiency Calculations. At level i of the synthesizer tree, we do $2 \cdot 2^{\log(\lambda)-i}$ matrix multiplications and $2^{\log(\lambda)-i}$ matrix rounding operations. If we sum over all levels of the tree, we perform $4\lambda - 1$ matrix multiplications and $2\lambda - 1$ matrix rounding operations. Since $p = 2$, the output of our synthesizer is a random-looking binary matrix that can be immediately used at the next level, and we do not have to spend any computational time reformatting this output.

Our matrices are of dimension size $m = n\log q$. If we set τ to be the exponent corresponding to optimal matrix multiplication, this means our PRF can be computed in time $\tilde{O}\left(\lambda n^{\tau}\right)$. In addition, note that we output m^2 bits, so our operations per output bit is $\tilde{O}\left(\lambda n^{\tau-2}\right)$.

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of matrix operations if this is what we are only allowed (i.e. multiply, add, and round) and absolute complexity. Our PRF F^2 clearly has $O\left(\log\lambda\right)$ matrix operation complexity, since the longest potential path from root to leaf on our synthesizer tree has $2\log\left(\lambda\right)$ multiplies and $\log\left(\lambda\right)$ rounding operations. Since matrix multiplication is in NC^1 [RW04] and our synthesizer tree has depth $\log\lambda$, this means F^2 is in NC^2.

Key and Public Parameter Size. The one area that our construction F^2 does not do well on is key size. For our key, we need $\log\lambda$ uniform matrices in $\mathbb{Z}_q^{m\times m}$ as well as 2λ matrices in $\mathbb{Z}_2^{m\times m}$. We note that these additional binary matrices can be made public without any loss of security (since they are what would traditionally be the public matrices of an LWE instance). This gives us secret key sizes of $\log\left(\lambda\right)\left(m^2\log q\right)$ and public parameter sizes of λm^2.

6 Proof of Security

In this section, we formally prove security of our synthesizer S_ℓ.

6.1 Security Proof Intuition

We start by discussing the principles behind the proof security of our synthesizer construction. Our proof is somewhat of a Frankenstein of the proofs of both [BLMR13] and [BP14]. The proof does not follow exactly from either of these PRF papers: we need to use an arbitrary polynomial number of matrices at every muiltiplication step. Additionally, the proof in [BLMR13] has the added restriction that the $\mathbf{A}$ matrices are full rank (the proof in [BP14] avoids this) which we would like to avoid because it would substantially complicate composing S in synthesizer constructions.

However, our proof is quite closely related to those two proofs, and a reader familiar with those two papers can predict how our proof works without too much imagination.

Proof Intuition for $\ell = 2$. In order to explain the proof at a high level, we will explicity spell out the case for $\ell = 2$. The intuition for this case directly applies to the cases for larger ℓ, and the notation can get very confusing since we have to keep track of an enormous number of variables, so we will explain this case in a fair amount of detail. For a reader familiar with [BLMR13], we note that we are essentially proving the generalized version of their PRF where, instead of having two matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ that we use at every level of the construction, we have k_i different random matrices $\mathbf{A}_{i,1}, ..., \mathbf{A}_{i,k_i}$ we use for each level i.

The proof involves two main steps: proving that an unrounded, noisy version of our synthesizer is secure, and then proving that rounding the unrounded, noisy synthesizer almost always results in the same value as computing the actual synthesizer. The proof of [BP14] almost exactly follows this format, and the proof of [BLMR13] accomplishes this by interleaving hybrids. We discuss each of these two steps below.

The Noisy Synthesizer. To begin, recall that our synthesizer for $\ell = 2$ has the form $\lceil \mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} \rfloor_p$. Ignoring the rounding, we simply have $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S}$. Our synthesizer S_2 is obviously insecure if we ignore the rounding. However, suppose we add noise at each layer the synthesizer. In other words, we compute

$$\mathbf{A}_{1,i}\left(\mathbf{A}_{2,j}\mathbf{S} + \mathbf{\Delta}_j\right) + \mathbf{\Delta}_{i,j} = \mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} + \left[\mathbf{A}_{1,i}\mathbf{\Delta}_j + \mathbf{\Delta}_{i,j}\right] \tag{6.1}$$

where the $\mathbf{\Delta}_j$ and $\mathbf{\Delta}_{i,j}$ are independently sampled noise matrices where each entry is sampled from some B-bounded noise distribution ψ. In order to show that this noisy synthesizer is random, we can use the format of the left-hand side of equation 6.1 and hybridize over the implicitly nested LWE with binary samples instances. This is shown in the table below.

Distribution	**Output for each (i,j) input**	**Reduction Technique**
$0'$: Unrounded Real Scheme with Noise	$\mathbf{A}_{1,i}\left(\mathbf{A}_{2,j}\mathbf{S} + \mathbf{\Delta}_j\right) + \mathbf{\Delta}_{i,j}$	–
$1'$:	$\mathbf{A}_{1,i}\left(\mathbf{S}_j\right) + \mathbf{\Delta}_{i,j}$	LWE with binary samples
$2'$: Random	$\mathbf{R}_{ij} \leftarrow \mathbb{Z}_q^{m \times m}$	LWE with binary samples

We note that the values $\mathbf{S}_j$ and $\mathbf{R}_{ij}$ are sampled independently and uniformly at random for each j and i, j pair, respectively. Additionally, we can sample the $\mathbf{\Delta}$ terms (and, if we are not given them in advance, the $\mathbf{A}$ terms as well) lazily. This will become important to remember as we move to higher ℓ.

Rounding the Synthesizer. In the preceding paragraph, we explained how the 'noisy synthesizer' version of S_2 can be shown to be secure. In the remainder of the proof, we show that, as long as the ratio $\frac{q}{p}$ is large enough, an efficient adversary can only find an input value (i, j) such that

$$\lceil \mathbf{A}_{1,i}\left(\mathbf{A}_{2,j}\mathbf{S} + \mathbf{\Delta}_j\right) + \mathbf{\Delta}_{i,j} \rfloor_p \neq \lceil \mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} \rfloor_p$$

with negligible probability. Since we have already shown the noisy synthesizer to be indistinguishable from random, this will complete the proof.

Our proof technique for this portion of the proof closely mirrors that of [BP14]. In [BLMR13], the authors use a slightly different technique (and overall series of hybrids). We start by restating equation 6.1.

$$\mathbf{A}_{1,i}\left(\mathbf{A}_{2,j}\mathbf{S} + \mathbf{\Delta}_j\right) + \mathbf{\Delta}_{i,j} = \mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} + \left[\mathbf{A}_{1,i}\mathbf{\Delta}_j + \mathbf{\Delta}_{i,j}\right] \tag{6.2}$$

Note that $\mathbf{A}_{1,i}$ is a binary matrix and $\mathbf{\Delta}_j$ has B-bounded entries, so the product $\mathbf{A}_{i,1}\mathbf{\Delta}_j$ has entries bounded by mB. Since $\mathbf{\Delta}_{i,j}$ has entries bounded by B, the entire sum $\mathbf{A}_{1,i}\mathbf{\Delta}_j + \mathbf{\Delta}_{i,j}$ has entries bounded by $(m+1)\,B$.

Now let's consider what would have to happen for a rounding error to occur: a rounding error happens when $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} + [\mathbf{A}_{1,i}\boldsymbol{\Delta}_j + \boldsymbol{\Delta}_{i,j}]$ and $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S}$ are closest to different integer multiples of $\frac{q}{p}$. Note that this implies that $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S}$ must be within $(m+1)\,B$ of an integer multiple of $\frac{q}{p}$.

Consider the following game: suppose a dealer chooses polynomially many random integers in $[0, q-1]$ and gives these to a player. We say that a player wins the game if any of these random integers is within $(m+1)\,B$ of an integer multiple of $\frac{q}{p}$. If we choose the ratio $\frac{q}{p}$ to be superpolynomially large in our security parameter, the probability that any single individual integer is close to a an integer multiple of $\frac{q}{p}$ is negligible and thus, by a union bound, the probability that any one of the set of integers is close to an integer multiple of $\frac{q}{p}$ is also negligible. Thus, the player loses with all but negligible probability.

Unfortunately, it is difficult to directly prove that an adversary cannot find a value of $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S}$ for some choices of i and j such that the overall value is close to an integer multiple of $\frac{q}{p}$. This is particularly troublesome to prove if k_1 and k_2 are superpolynomially large (and we are, of course, lazily sampling) and the adversary gets to select which choices of i and j for which they see outputs (which happens implicitly in the hybrid arguments of our PRF proofs). If we had such a proof, then we could perhaps simplify our overall proof substantially.

However, we have already shown that $\mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} + [\mathbf{A}_{1,i}\boldsymbol{\Delta}_j + \boldsymbol{\Delta}_{i,j}]$ is indistinguishable from random. Since no efficient adversary can distinguish this from random, we know that, intuitively, an efficient adversary can only find an output of this noisy synthesizer within $(m+1)\,B$ of an integer multiple of $\frac{q}{p}$ with negligible probability, as this would distinguish the noisy synthesizer from random. Since we know that this noisy synthesizer is always within $(m+1)$B of the real, unrounded synthesizer, we can see that any adversary that can cause a rounding error to happen can distinguish the noisy synthesizer from random, which completes the proof.

This rounding error issue is the reason why we need two sets of hybrid arguments instead of just one: we need the proof that the noisy synthesizer is random in order to show that this rounding error does not occur. This is the only step in the reduction where a superpolynomial modulus is required. If we had a more efficient LWR to LWE reduction, we might be able to avoid such an inefficient proof. The table below summarizes the second set of hybrids.

Distribution	**Output for each** (i,j) **input**	**Reduction Technique**
0: Real Scheme	$\lceil \mathbf{A}_{1,i}\mathbf{A}_{2,j}\mathbf{S} \rfloor_p$	–
1	$\lceil \mathbf{A}_{1,i}\left(\mathbf{B}_{2,j}\mathbf{S} + \boldsymbol{\Delta}_j\right) + \boldsymbol{\Delta}_{i,j} \rfloor_p$	Error doesn't change output Relies on $0' \to 2'$
2	$\lceil \mathbf{A}_{1,i}\left(\mathbf{S}_j\right) + \boldsymbol{\Delta}_{i,j} \rfloor_p$	Implied by $0' \to 1'$
3: Random	$\lceil \mathbf{R}_{ij} \leftarrow \mathbb{Z}_q^{m \times m} \rfloor_p$	Implied by $1' \to 2'$

Moving to Higher ℓ We have shown how security works for our synthesizer S_2. For larger values of ℓ, the proof technique is exactly the same. The proofs of S_ℓ include $\ell - 2$ additional hybrid levels, but the overall structure is exactly the same. The main differences are semantic: we introduce quite a bit of notation, which can make the proof difficult to follow. This is why we decided to explain the $\ell = 2$ case for intuition.

6.2 Background Definitions and Lemmas

Before we start with the formal proof, we need to introduce some new notation. We will also show some helpful lemmas that will greatly simplify our proof.

Error Aggregating Function $\mathcal{E}$. In order to make the proof readable, we will lump all of the error terms together into a single term, almost exactly like in the proof of [BP14]. This will

allow us to state our theorems more concisely.

Definition 6.1. *Let m, q, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_\ell]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_2^{m \times m}$ be a uniformly distributed matrix.*

We next define a series of $\mathbf{\Delta}$ terms. For $x_1 \in [0, k_1]$, $x_2 \in [0, k_2]$, ... , $x_\ell \in [0, k_\ell]$ we let each term of one of the following forms

$$\mathbf{\Delta}_{x_\ell}, \mathbf{\Delta}_{x_{\ell-1},x_\ell}, ...\mathbf{\Delta}_{x_1,x_2,...,x_\ell}$$

be sampled such that each entry is sampled independently from some B-bounded distribution ψ. We note that there are potentially exponentially many $\mathbf{\Delta}$ terms. However, when computing the function $\mathcal{E}$, we can sample these terms lazily.

Let $d \in [1, \ell]$. We define the function $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) : [1, ..., k_1] \times ... \times [1, ..., k_\ell] \to \mathbb{Z}^{m \times m}$ in the following way:

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=} \sum_{j=0}^{d-1} \left(\left[\prod_{i=1}^{j} \mathbf{A}_{i,x_i} \right] \mathbf{\Delta} x_{j+1}, ...x_\ell \right)$$

The equation for $\mathcal{E}$ looks extremely complicated. In reality, it is just the generalization of the telescoping noise term that we showed in the $\ell = 2$ synthesizer proof sketch. For instance, if we expand out the summation, it is easy to see that (for some $d >> 3$)

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=} \left(\left[\prod_{i=1}^{d-1} \mathbf{A}_{i,x_i} \right] \mathbf{\Delta} x_d, ...x_\ell \right) + \left(\left[\prod_{i=1}^{d-2} \mathbf{A}_{i,x_i} \right] \mathbf{\Delta} x_{d-1}, x_d, ...x_\ell \right) + ...$$

$$(\mathbf{A}_{2,x_2}\mathbf{A}_{1,x_1}\mathbf{\Delta}_{x_3,...,x_\ell}) + (\mathbf{A}_{1,x_1}\mathbf{\Delta}_{x_2,....,x_\ell}) + \mathbf{\Delta}_{x_1,...,x_\ell}$$

We can go one step further and see that

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=}$$

$$\mathbf{A}_{1,x_1}\left(\mathbf{A}_{2,x_2}\left[...\left[\mathbf{A}_{d-2,x_{d-2}}\left(\mathbf{A}_{d-1,x_{d-1}}(\mathbf{\Delta}_{x_d,...,x_\ell}) + \mathbf{\Delta}_{x_{d-1},...,x_\ell}\right) + \mathbf{\Delta}_{x_{d-1},...,x_\ell}\right] ...\right] + \mathbf{\Delta}_{x_2,...,x_\ell}\right) + \mathbf{\Delta}_{x_1,...,x_\ell}$$

It will be useful to view $\mathcal{E}$ in this telescoping form. Note that

$$\mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=}$$

$$\mathbf{A}_{1,x_1}\left(\mathbf{A}_{2,x_2}\left[...\left[\mathbf{A}_{d-2,x_{d-2}}\left(\cancel{\mathbf{A}_{d-1,x_{d-1}}(\mathbf{\Delta}_{x_d,...,x_\ell})} + \mathbf{\Delta}_{x_{d-1},...,x_\ell}\right) + \mathbf{\Delta}_{x_{d-1},...,x_\ell}\right] ...\right] + \mathbf{\Delta}_{x_2,...,x_\ell}\right) + \mathbf{\Delta}_{x_1,...,x_\ell}$$

In other words, $\mathcal{E}^{d,\ell}$ is just $\mathcal{E}^{d-1,\ell}$ with an extra $\mathbf{A}$ and $\mathbf{\Delta}$ term. An astute reader might already be able to see how this might be useful for a hybrid argument. This gives us the following identity:

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=} \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) + \prod_{i=1}^{d-1} \mathbf{A}_{i,x_i} \mathbf{\Delta}_{x_d,...,x_\ell} \tag{6.3}$$

In addition, note that $\mathcal{E}^{1,\ell}(x_1, ..., x_\ell) \stackrel{\mathsf{def}}{=} \mathbf{\Delta}_{x_1,...,x_\ell}$.

Bounds on $\mathcal{E}$ Before we go further, it will be useful to have a bound on the maximum size that $\mathcal{E}$ can attain. We prove this in the following lemma:

Lemma 6.2. *Let m, q, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. Let the function $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.1, with all relevant parameters defined there as well.*

It is the case that every entry in the matrix output by $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ has norm at most $2m^\ell B$.

Proof. Recall that we can write, for d and ℓ large enough,

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\text{def}}{=} \left(\left[\prod_{i=1}^{d-1} \mathbf{A}_{i,x_i}\right] \boldsymbol{\Delta} x_d, ...x_\ell\right) + \left(\left[\prod_{i=1}^{d-2} \mathbf{A}_{i,x_i}\right] \boldsymbol{\Delta} x_{d-1}, x_d, ...x_\ell\right) + ...$$

$$(\mathbf{A}_{2,x_2}\mathbf{A}_{1,x_1}\boldsymbol{\Delta}_{x_3,...,x_\ell}) + (\mathbf{A}_{1,x_1}\boldsymbol{\Delta}_{x_2,....,x_\ell}) + \boldsymbol{\Delta}_{x_1,...,x_\ell}$$

By definition, each individual entry in all of the $\boldsymbol{\Delta}$ matrices is B-bounded. The dot product of any vector with B-bounded entries and a binary vector must be at most mB. Chaining this property through all d matrices and assuming that $m \geq 2$ allows us to complete the proof. ∎

6.3 The Noisy Pseudorandom Synthesizer

With our error function $\mathcal{E}$ now defined (in definition 6.1), we can define our general noisy pseudorandom synthesizer, which we will call $\mathcal{G}$ (in another nod to the proof of [BP14]. We define this function below:

Definition 6.3. *Let m, q, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_\ell]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_2^{m \times m}$ be a uniformly distributed full-rank matrix.*

We next define a series of $\boldsymbol{\Delta}$ terms. For $x_1 \in [1, ..., k_1]$, $x_2 \in [1, ..., k_2]$, ... , $x_\ell \in [1, ..., k_\ell]$ we let each term of one of the following forms

$$\boldsymbol{\Delta}_{x_\ell}, \boldsymbol{\Delta}_{x_{\ell-1},x_\ell}, ...\boldsymbol{\Delta}_{x_1,x_2,...,x_\ell}$$

be sampled such that each entry is sampled independently from some B-bounded distribution ψ.

Let $\mathbf{S}_{x_{d+1},...,x_\ell} \in \mathbb{Z}_q^{m \times m}$ be sampled uniformly at random for each different value of the set $(x_{d+1}, ..., x_\ell)$. Let the function $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.1.

The function $\mathcal{G}^{d,\ell} : [1, ..., k_1] \times ... \times [1, ..., k_\ell] \to \mathbb{Z}_q^{m \times m}$ is defined in the following way:

$$\mathcal{G}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\text{def}}{=} \prod_{i=1}^{d} (\mathbf{A}_{i,x_i}) \mathbf{S}_{x_{d+1},...,x_\ell} + \mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$$

First, note that $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ is exactly the noisy synthesizer corresponding to the proper synthesizer S^ℓ. To see this (and, more generally, some insight on the hybrids), note that $\mathcal{G}^{d,\ell}$ can be written in the following way:

$$\mathcal{G}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\text{def}}{=}$$

$$\mathbf{A}_{1,x_1}\left(\mathbf{A}_{2,x_2}\left[...\left[\mathbf{A}_{d-2,x_{d-2}}\left(\mathbf{A}_{d-1,x_{d-1}}\left(\mathbf{A}_{d,x_d}\mathbf{S}_{x_{d+1},...x_\ell} + \boldsymbol{\Delta}_{x_d,...,x_\ell}\right) + \boldsymbol{\Delta}_{x_{d-1},...,x_\ell}\right) + ...\right]...\right] + ...\right) + \boldsymbol{\Delta}_{x_1,...,x_\ell} \tag{6.4}$$

Hybrid Argument. We are now in position to state (and prove) a hybridizing lemma. Since the functions $\mathcal{G}^{d,\ell}$ can have superpolynomially long output for certain choices of d and ℓ, we will let the adversary adaptively query $\mathcal{G}^{d,\ell}$ at certain points $(x_1, ..., x_\ell)$ rather than send them all of the data (since we cannot possibly send a superpolynomial amount of data). In the case where $\mathcal{G}^{d,\ell}$ only has a polynomial number of outputs, we assume that an adversary can just query all of them if they like.

Lemma 6.4. *Let m, q, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. Let $\mathcal{T} \in \mathbb{Z}^m$ be defined to be the distribution of random binary vectors over $\mathbb{Z}^m$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_\ell]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_2^{m \times m}$ be a uniformly distributed full-rank matrix.*

We next define a series of $\mathbf{\Delta}$ terms. For $x_1 \in [1, ..., k_1]$, $x_2 \in [1, ..., k_2]$, ... , $x_\ell \in [1, ..., k_\ell]$we let each term of one of the following forms

$$\mathbf{\Delta}_{x_\ell}, \mathbf{\Delta}_{x_{\ell-1}, x_\ell}, ...\mathbf{\Delta}_{x_1, x_2, ..., x_\ell}$$

be sampled such that each entry is sampled independently from some B-bounded distribution ψ.

Let $\mathbf{S}_{x_{d+1}, ..., x_\ell} \in \mathbb{Z}_q^{m \times m}$ be sampled uniformly at random for each different value of the set $(x_{d+1}, ..., x_\ell)$. Let the function $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.1 and let the function $\mathcal{G}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.3. Let $d \in [1, \ell]$ be an integer.

Any adversary that can distinguish $\mathcal{G}^{d-1,\ell}(x_1, ..., x_\ell)$ from $\mathcal{G}^{d,\ell}(x_1, ..., x_\ell)$ with advantage ε in Q queries to $\mathcal{G}$ of the form $(x_1, ..., x_\ell)$ can be used to solve the $\left(q, m, \psi, \mathcal{U}_q^m, \mathcal{T}\right)$-LWE problem with advantage $\frac{\varepsilon}{mQ}$.

Proof. Suppose we are given a collection of $\left(q, m, \psi, \mathcal{U}_q^m, \mathcal{T}\right)$-LWE^{m,Q} oracles $\mathcal{O}_z^{\text{LWE}^m}$ for $z \in [1, ..., Q]$, all of which are either of the form $\mathcal{O}_{\mathbf{s}}^{\text{LWE}^m} \stackrel{\text{def}}{=} \mathcal{O}_{\mathbf{S}}^{\text{LWE}}$ or $\mathcal{O}_{\$}^{\text{LWE}^m} \stackrel{\text{def}}{=} \mathcal{O}_{\$}^{\text{LWE}}$. We must respond to an adversary's queries of the form $(x_1, ..., x_\ell)$ and simulate either the function $\mathcal{G}^{d-1,\ell}$ (in the case we received oracles of the form $\mathcal{O}_{\$}^{\text{LWE}}$) or $\mathcal{G}^{d,\ell}$ (in the case we received oracles of the form $\mathcal{O}_{\mathbf{S}}^{\text{LWE}}$).

In order to do this, we make the following tables

- ◇ Table $T_{\mathbf{\Delta}}$ consisting of all terms of the form $\mathbf{\Delta}_{x_{d-1}, ..., x_\ell}$, $\mathbf{\Delta}_{x_{d-2}, x_{d-1}, ..., x_\ell}$, ... , $\mathbf{\Delta}_{x_1, ..., x_\ell}$ for all $x_{d-1} \in [1, ..., k_{d-1}], ..., x_\ell \in [1, ..., k_\ell]$
- ◇ Table $T_{\mathbf{A}}$ consisting of all terms of the form $\mathbf{A}_{i,x_i}$ for $i \in [1, d-1]$.
- ◇ Table $T_{\mathcal{O}}$ consisting of a mapping between a set of partial inputs $(x_d, ..., x_\ell)$ and a set of oracle numbers $[1, ..., Q]$
- ◇ Table T_{out} consisting of sets of the form $(i, j, \mathbf{C}) \in [1, ..., Q] \times [1, ..., k_d] \times \mathbb{Z}_q^{m \times m}$. These sets correspond to the jth set of m concatenated output values from the LWE oracle indexed by i.

Note that the $\mathbf{\Delta}$ terms may be exponential in number, but that this is OK since we will sample them lazily. Additionally, we comment that T_{out} has k_d possible values for each choice of i because this is the total number of matrices of the form $\mathbf{A}_{x_d, j}$.

We respond to queries from the adversary of the form $(x_1, ..., x_\ell)$ in the following way:

1. Check if the string of partial inputs $(x_d, ..., x_\ell)$ has a value in table $T_{\mathcal{O}}$. If it does, set $\mathcal{O}^{\text{LWE}}$ equal to the LWE oracle indexed by $T_{\mathcal{O}}[(x_d, ..., x_\ell)]$. Otherwise, select the unused oracle with the lowest index c in the set $[1, ..., Q]$, write the mapping $(x_d, ..., x_\ell) \to c$ in the table $T_{\mathcal{O}}$, and set $\mathcal{O}^{\text{LWE}}$ equal to this oracle.

2. Check the table $T_{\mathbf{\Delta}}$ to see if the terms $\mathbf{\Delta}_{x_{d-1},...,x_\ell}$, $\mathbf{\Delta}_{x_{d-2},x_{d-1},...,x_\ell}$, ... , $\mathbf{\Delta}_{x_1,....,x_\ell}$ have been already added. If not, sample them appropriately and add them to the table.

3. Check the table $T_{\mathbf{A}}$ to see if the terms $\mathbf{A}_{i,x_i}$ are included. If not, sample them appropriately and add them.

4. Check if table T_{out} has a set of the form $(c, x_d, \mathbf{C})$ for some $\mathbf{C}$. If it does, set $\mathbf{Y} \in \mathbb{Z}_q^{m\times m} = \mathbf{C}$. If not, query $\mathcal{O}^{\text{LWE}}$ m times and concatenate the result into a matrix $\mathbf{Y} \in \mathbb{Z}_q^{m\times m}$. Add the entry $(c, x_d, \mathbf{Y})$ to the table T_{out}.

5. Compute the function $\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\,\mathbf{Y}\right] + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell)$ and send it to the adversary.

We claim that this faithfully simulates $\mathcal{G}^{d-1,\ell}$ if we were given parallel instances of $\mathcal{O}_{\$}^{\text{LWE}}$ and $\mathcal{G}^{d,\ell}$ if we were given parallel instances of $\mathcal{O}_{\mathbf{S}}^{\text{LWE}}$. To see this, let's consider the output in each case. Suppose we start with the case where we were given parallel instances of $\mathcal{O}_{\mathbf{S}}^{\text{LWE}}$. We have:

$$\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\,\mathbf{Y}\right] + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) = \left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right]\left(\mathbf{A}_{d,x_d}\mathbf{S}_{x_{d+1},...,x_\ell} + \mathbf{\Delta}_{x_d,...,x_\ell}\right) + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell)$$

Recall that, from equation 6.3

$$\mathcal{E}^{d,\ell}(x_1, ..., x_\ell) \stackrel{\text{def}}{=} \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) + \prod_{i=1}^{d-1}\mathbf{A}_{i,x_i}\mathbf{\Delta}_{x_d,...,x_\ell} \tag{6.5}$$

Thus, we can say that

$$\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right]\left(\mathbf{A}_{d,x_d}\mathbf{S}_{x_{d+1},...,x_\ell} + \mathbf{\Delta}_{x_d,...,x_\ell}\right) + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) =$$

$$\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right]\left(\mathbf{A}_{d,x_d}\mathbf{S}_{x_{d+1},...,x_\ell}\right) + \left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right]\mathbf{\Delta}_{x_d,...,x_\ell} + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) =$$

$$\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right]\left(\mathbf{A}_{d,x_d}\mathbf{S}_{x_{d+1},...,x_\ell}\right) + \mathcal{E}^{d,\ell}(x_1, ..., x_\ell) = \mathcal{G}^{d,\ell}(x_1, ..., x_\ell)$$

This proves the 'real' case. Now assume instead we were given parallel instances of pure randomness (or, as we *eloquently* write, $\mathcal{O}_{\$}^{\text{LWE}}$). In this case we have

$$\left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\,\mathbf{Y}\right] + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell) = \left[\prod_{i=1}^{d-1}(\mathbf{A}_i)\right](\mathbf{S}_{x_d,...,x_\ell}) + \mathcal{E}^{d-1,\ell}(x_1, ..., x_\ell)$$

$$= \mathcal{G}^{d-1,\ell}(x_1, ..., x_\ell)$$

This shows the 'random' case. Thus, we have faithfully simulated $\mathcal{G}^{d-1,\ell}$ if we were given parallel instances of $\mathcal{O}_{\$}^{\text{LWE}}$ and $\mathcal{G}^{d,\ell}$ if we were given parallel instances of $\mathcal{O}_{\mathbf{S}}^{\text{LWE}}$.

Going back to our overall argument, after Q queries, the adversary responds to us with a guess bit b_g, indicating either $\mathcal{G}^{d-1,\ell}$ (zero) or $\mathcal{G}^{d,\ell}$ (one). If the adversary has guessed $G^{d-1,\ell}$, we guess that we were given random data, and adversary has guessed $G^{d,\ell}$, we guess that we were given a true parallel LWE instance. Suppose that the adversary has advantage ε. Since we have

faithfully simulated the proper distributions, we know that the adversary can distinguish our parallel LWE oracle problem with advantage ε. As we discussed in section 2.3, this means that the adversary can distinguish a true LWE oracle from random with advantage $\frac{\varepsilon}{mq}$, completing the proof. ∎

Given our hybrid argument, we are now in position to make the claim that our noisy pseudorandom synthesizer ($G^{\ell,\ell}$) is indistinguishable from random. We state this in a lemma below.

Lemma 6.5. *Let m, q, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. Let $\mathcal{T} \in \mathbb{Z}^m$ be defined to be the distribution of random binary vectors over $\mathbb{Z}^m$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_\ell]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_2^{m \times m}$ be a uniformly distributed full-rank matrix.*

We next define a series of $\mathbf{\Delta}$ terms. For $x_1 \in [1, ..., k_1]$, $x_2 \in [1, ..., k_2]$, ... , $x_\ell \in [1, ..., k_\ell]$we let each term of one of the following forms

$$\mathbf{\Delta}_{x_\ell}, \mathbf{\Delta}_{x_{\ell-1},x_\ell}, ...\mathbf{\Delta}_{x_1,x_2,...,x_\ell}$$

be sampled such that each entry is sampled independently from some B-bounded distribution ψ.

Let $\mathbf{S}_{x_{d+1},...,x_\ell} \in \mathbb{Z}_q^{m \times m}$ be sampled uniformly at random for each different value of the set $(x_{d+1}, ..., x_\ell)$. Let the function $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.1 and let the function $\mathcal{G}^{d,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.3. Let $d \in [1, \ell]$ be an integer.

Any adversary that can distinguish our noisy synthesizer $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from the truly random function $\mathcal{G}^{0,\ell}(x_1, ..., x_\ell)$ with advantage ε in Q queriesof the form $(x_1, ..., x_\ell)$ can be used to solve the $\left(q, m, \psi, \mathcal{U}_q^m, \mathcal{T}\right)$-LWE problem with advantage $\frac{\varepsilon}{mQ\ell}$.

Proof. This follows by a basic hybrid argument over the choice of d in $\mathcal{G}^{d,\ell}$ using lemma 6.4. ∎

6.4 Handling the Rounding

We have now shown that our noisy pseudorandom synthesizer $\mathcal{G}^{d,\ell}$ is indistinguishable from random. As we mentioned in the proof outline, what remains is to show that the rounded noisy synthesizer is indistinguishable from the actual synthesizer. This corresponds to dealing with what [BLMR13] and [BP14] refer to as the BAD event. Our argument is technically almost identical to that of [BP14], although we phrase it a little bit differently.

Lemma 6.6. *Let m, q, p, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution for some real number B. We also require that $q \geq 2m^\ell B m^{\omega(1)} p$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. Let the noisy pseudorandom synthesizer $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.3.*

Any adversary that can find some input $(x_1, ..., x_\ell)$ such that value of $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ has an entry within $2m^\ell B$ of an integer multiple of $\frac{q}{p}$ (a rounding boundary) can be used to distinguish the function $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from random.

Proof. This follows almost immediately from our choices of p and q. Since $q \geq 2m^\ell B m^{\omega(1)} p$ is superpolynomially large in p, the probability that a truly random function $\mathcal{U}(x_1, ..., x_\ell) \to \mathbb{Z}_q^{m \times m}$ outputs a matrix with an entry close to an integer multiple of $\frac{q}{p}$ is any polynomial number of samples is negligble.

Thus, any adversary that can find such an output is immediately a distinguisher, and the advantage ε is preserved. ∎

Lemma 6.7. *Let m, q, p, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution for some real number B. We also require that $q \geq 2m^{\ell} B m^{\omega(1)} p$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. Let the noisy pseudorandom synthesizer $calG^{\ell,\ell}(x_1, ..., x_\ell)$ be defined as in definition 6.3, and let the synthesizer $S_\ell(x_1, ..., x_\ell)$ be defined as in definition 4.1.*

Any adversary that can distinguish $\left\lceil \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell) \right\rfloor_p$ from $S_\ell(x_1, ..., x_\ell)$ with advantage ε can be used to distinguish $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from random with advantage ε.

Proof. Let S'_ℓ be the unrounded version of the synthesizer. In other words,

$$S'_\ell(i_1, ...i_\ell) \stackrel{\text{def}}{=} \prod_{i_1,...,i_\ell} [\mathbf{A}_{1,i_1} ... \mathbf{A}_{\ell,i_\ell}] \mathbf{S}$$

where all of the terms are as defined in 4.1. Note that, by definition

$$S'_\ell(x_1, ..., x_\ell) + \mathcal{E}^{\ell,\ell}(x_1, ..., x_\ell) \stackrel{\text{def}}{=} \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$$

Recall from lemma 6.2 that every entry in the matrix output by $\mathcal{E}^{d,\ell}(x_1, ..., x_\ell)$ has norm at most $2m^{\ell} B$. Thus, the only time that $\left\lceil \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell) \right\rfloor_p$ and $S_\ell(x_1, ..., x_\ell)$ actually output a different value is when some entry of $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ is within $2m^{\ell} B$ of an integer multiple of $\frac{q}{p}$.

Thus, in order to distinguish $\left\lceil \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell) \right\rfloor_p$ from $S_\ell(x_1, ..., x_\ell)$, an adversary must be able to output some value $(x_1, ..., x_\ell)$ such that some entry of $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ is within a factor of $2m^{\ell} B$ of an integer multiple of $\frac{q}{p}$. In lemma 6.6, we showed that any adversary that could do this with advantage ε could distinguish $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from random with advantage ε, which completes our proof. ∎

6.5 Putting It All Together

We have now worked out the proofs for all of our hybrids and can state a theorem regarding the security of our synthesizer S_ℓ.

Theorem 6.8. *Let m, n, q, p, and ℓ be integers, and let $\psi \in \mathbb{Z}$ be a B-bounded noise distribution. We additionally require that $q \geq 2m^{\ell+\omega(1)} Bp$ and that $m = n \log q$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. Let $\mathcal{U}_q^{m \times m}$ denote the uniform distribution over matrices in $\mathbb{Z}_q^{m \times m}$ and let $\mathcal{T} \in \mathbb{Z}^m$ be defined to be the distribution of random binary vectors over $\mathbb{Z}^m$. Any adversary that can distinguish S_ℓ from $\left\lceil \mathcal{U}_q^{m \times m} \right\rfloor_p$ with advantage ε can be used to solve the $\left(q, n, \psi, \mathcal{U}_q^n, U_q^n\right)$-*LWE *problem with advantage $\frac{\varepsilon}{2mQ}$.*

Proof. This follows from our series of hybrids and a theorem from [BLMR13] about low-norm LWE samples, which we will list below:

1. In lemma 6.5, we showed that any adversary that could distinguish our noisy synthesizer $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from $\mathcal{U}_q^{m \times m}$ (random) could be used to solve the listed LWE problem with advantage $\frac{\varepsilon}{mQ}$. This immediately implies that any adversary that can distinguish $\left\lceil \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell) \right\rfloor_p$ from $\left\lceil \mathcal{U}_q^{m \times m} \right\rfloor_p$ can also be used to solve the $\left(q, m, \psi, \mathcal{U}_q^m, \mathcal{T}\right)$-LWE problem with advantage $\frac{\varepsilon}{mQ}$.

2. In lemma 6.7, we showed that any adversary that can distinguish $\left\lceil \mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell) \right\rfloor_p$ from $S_\ell(x_1, ..., x_\ell)$ with advantage ε can be used to distinguish $\mathcal{G}^{\ell,\ell}(x_1, ..., x_\ell)$ from random with advantage ε.

3. In theorem 2.6 from [BLMR13], it is shown that any adversary that can solve the $\left(q, m, \psi, \mathcal{U}_q^m, \mathcal{T}\right)$-LWE problem with advantage ε can be used to solve the $\left(q, n, \psi, \mathcal{U}_q^n, U_q^n\right)$-LWE problem with advantage ε.

Combining these three results means that any adversary that can distinguish S_ℓ from $\left\lceil \mathcal{U}_q^{m\times m} \right\rfloor_p$ with advantage ε can be used to solve the $\left(q, n, \psi, \mathcal{U}_q^n, U_q^n\right)$-LWE problem (the standard version of LWE) with advantage $\frac{\varepsilon}{2mQ}$. ∎

It is important to note that the final output $\left\lceil \mathcal{U}_q^{m\times m} \right\rfloor_p$ is only uniform if p divides q, but negligibly far from uniform if $\frac{q}{p}$ is superpolynomially large. Since our theorem only holds if ratio $\frac{q}{p}$ to be superpolynomially large, we do not worry about this issue.

An astute reader may also notice that we have technically proven that a single instance of S_ℓ is hard, rather than parallel instances with the same key. However, since we have not restricted the number of different queries in a particular coordinate (i.e. the number of matrices $\mathbf{A}_{i,x_i}$ for a fixed i) we are essentially allowing the adversary to query any combination of input matrices they like, giving the adversary more power than they would have in the parallel game (they can effectively mix and match matrices from the different parallel instances in our reduction above).

7 Key Subset-Product Pseudorandom Synthesizer

In this section we define a new pseudorandom synthesizer inspired by (and very similar to) the PRFs invented in [BPR12]. We call this new pseudorandom synthesizer K_ℓ as it uses a key subset-product like these PRFs.

7.1 Synthesizer Construction

Definition 7.1. *Let m, q, p, and ℓ be integers such that $p \leq q$ and let $\mathbf{\Psi} \in \mathbb{Z}^{m\times m}$ be a noise distribution over matrices where each entry is B-bounded. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_i]$ let $\mathbf{S}_{i,j} \in \mathbb{Z}^{m\times m}$ be a matrix sampled randomly from $\mathbf{\Psi}$. Let $\mathbf{A} \in \mathbb{Z}_q^{m\times m}$ be a matrix sampled uniformly at random.*

We define the synthesizer $K_\ell : \mathbb{Z}_q^{m\times m} \times \left[(\mathbb{Z}^{m\times m})^{k_1} \times ... \times (\mathbb{Z}^{m\times m})^{k_\ell}\right] \to \mathbb{Z}_p^{k_1 m\times ...\times k_\ell m}$ in the following way: for each $m \times m$ block of output of K_ℓ, define

$$K_\ell\left(x_1, ...x_\ell\right) \stackrel{\text{def}}{=} \left\lceil \mathbf{A}\left[\prod_{i=1}^{\ell} \mathbf{S}_{i,x_i}\right]\right\rfloor_p \tag{7.1}$$

where each $x_i \in [1, k_i]$.

We now offer some comments on our synthezier K_{LWE}. We are essentially reversing the traditional notion of 'key' and 'sample' in our synthesizer definition, as the synthesizer 'key' is the LWE sample, and the synthesizer 'samples' are the LWE secrets. This works because we do not necessarily have to keep the key of a keyed pseudorandom synthesizer secret in order to maintain security.

In addition, note that it is a bit more complicated to 'chain' this synthesizer than it was to 'chain' S_ℓ. However, as long as we make sure that it takes less than $m^2 \log p$ bits of entropy (the amount of entropy in a random matrix over $Z_p^{m\times m}$) to sample a new matrix from $\mathbf{\Psi}$, we can still 'chain' this synthesizer. In other words, if this is the case, we can modify the output of each (i, j)-block of the synthesizer $K_{\text{LWE}(i,j)}$ to be a randomly sampled matrix from $\mathbf{\Psi}$. This is obviously not as straightforward of a transformation as that of S_ℓ, but it is still possible to do in a number of different ways. We will discuss some of these later in the paper.

We note that the security of this synthesizer follows almost immediately from the proof of security of the 'direct' PRF construction in [BPR12]. We discuss this more in section 9.

7.2 Synthesizer Construction from Ring LWE

Unlike the [BLMR13]-inspired synthesizer S_ℓ, we can build pseudorandom synthesizers based on ring-LWE using key subset product constructions. This is because ring-LWE with low norm samples is not a hard problem, but ring-LWE where the key is drawn from the noise distribution is a hard problem. This allows us to build more efficient synthesizers (and thus more efficient PRFs), although at the cost of potentially stronger ring-based lattice assumptions.

Definition 7.2. *Let m, q, p, and ℓ be integers such that $p \leq q$, let R be a polynomial ring of degree m, and let $\boldsymbol{\psi} \in R$ be a noise distribution over R where each entry is B-bounded. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$ and $j \in [1, k_i]$ let $\mathbf{s}_{i,j} \in R$ be ring elements matrix sampled randomly from $\boldsymbol{\psi}$. Let $\mathbf{a} \in R_q$ be a ring element sampled uniformly at random.*

We define the synthesizer $K_{R,\ell} : R_q \times \left[(R)^{k_1} \times ... \times (R)^{k_\ell}\right] \to R_p^{k_1 \times ... \times k_\ell}$ in the following way: for each R 'block' of output of $K_{R,\ell}$, define

$$K_{R,\ell}(x_1, ...x_\ell) \stackrel{\mathsf{def}}{=} \left\lfloor \mathbf{a} \left[\prod_{i=1}^{\ell} \mathbf{s}_{i,x_i} \right] \right\rceil_p \tag{7.2}$$

where each $x_i \in [1, k_i]$.

Again, the security of this synthesizer follows almost immediately from the proof of security of the 'direct' PRF construction in [BPR12]. This is also discussed in section 9.

8 Constructions of PRFs from K_ℓ and $K_{R,\ell}$

In this section we show how our synthesizer K_ℓ can be used to build PRFs. Before we do, however, we need an additional lemma to make sure that our synthesizer constructions compose properly.

8.1 LWE with Large Uniform Key and Noise

Recall that in [ACPS09], the authors show that LWE is hard even when the key is drawn from the noise distribution. We state this result in theorem 2.4. However, in our synthesizer constructions from K_ℓ and $K_{R,\ell}$, we would like to avoid having to continually sample Gaussian (or other complicated) noise distributions, as these complicate and slow down the construction.

To avoid such complicated noise distributions, below we show that LWE is hard when both the key and noise are sampled from a (superpolynomially large) uniform distribution. This result obviously necessitates a superpolynomially large modulus q (and thus larger approximation factors in our reductions from worst-case lattice problems), but this is already needed by our PRF constructions, so making this relaxation does not end up hurting our PRF parameters substantially. The proof is essentially just based on the technique of 'noise flooding'.

Lemma 8.1. *Consider integers n and q and some B-bounded distribution ψ over $\mathbb{Z}_q$. Let $p < 2q$ be an integer such that $p \geq B \cdot n^{\omega(1)}$. In addition, let $\boldsymbol{\psi} \in \mathbb{Z}^n$ be the distribution made by concatenating n samples from ψ into a vector.*

The $(q, n, \mathcal{U}_p, \mathcal{U}_p^n, \mathcal{U}_q^n)$-LWE problem is at least as hard as the $\left(q, n, \psi, \boldsymbol{\psi}, \mathcal{U}_q^n\right)$-LWE problem.

Proof. This result follows simply. Suppose we are given an LWE oracle $\mathcal{O}^{\text{LWE}}$ corresponding to the $\left(q, n, \psi, \boldsymbol{\psi}, \mathcal{U}_q^n\right)$-LWE problem which is either $\mathcal{O}_{\mathbf{s}}^{\text{LWE}}$ or $\mathcal{O}_{\$}^{\text{LWE}}$. We get samples of the form $(\mathbf{a}_i, \mathbf{r}_i)$ where r_i is either equal to $\mathbf{a}_i\mathbf{s} + \delta_i$ or truly random.

Now suppose we sample some $\mathbf{t} \in \mathbb{Z}^m$ from $\mathcal{U}_p^m$. In addition, for each sample we receive from the oracle, we sample some $\mathbf{b}_i$ from $\mathcal{U}_p^m$ as well. Then we take each sample from $\mathcal{O}^{\text{LWE}}$ and add $\mathbf{a}_i\mathbf{t} + \mathbf{b}_i$ to the second term. Since the uniformly random terms are superpolynomially larger than the key from $\boldsymbol{\psi}$ and the noise sample from ψ, the resulting sum is statistically close to $\mathbf{a}_i\mathbf{t} + \mathbf{b}_i$ if we were given $\mathcal{O}_{\mathbf{s}}^{\text{LWE}}$ and still uniformly random if we were given $\mathcal{O}_{\$}^{\text{LWE}}$, meaning we have successfully completed the simulation. ∎

We also note that a similar lemma follows for rings.

Lemma 8.2. *Consider integers n and q and let R be a polynomial ring of degree n, as well as some B-bounded distribution ψ over R. Let $p < 2q$ be an integer such that $p \geq B \cdot n^{\omega(1)}$.*

The $(R, q, n, \mathcal{U}_p, \mathcal{U}_p^n, \mathcal{U}_q^n)$-RLWE problem is at least as hard as the $\left(R, q, n, \psi, \psi, \mathcal{U}_q^n\right)$-RLWE problem.

Proof. The proof follows using the exact same argument as in lemma 8.1. ∎

8.2 Synthesizer Statement

As with S_ℓ, we state an overall theorem about the security of PRFs constructed from our pseudorandom synthesizers using the synthesizer construction of [NR95]. This theorem follows almost immediately from applying the synthesizer construction security theorem of [NR95] as we stated in theorem 3.6 to our theorem 9.1 proving our synthesizer K_ℓ secure.

As discussed in section 9, we know that K_ℓ and $K_{R,\ell}$ are secure for a wide variety of parameters. However, in order to make our synthesizers compose properly into PRFs, we must ensure that the output of each synthesizer has enough randomness to sample a new input. To make this really easy (and to optimize our efficiency) we will choose our parameters so that the inputs and outputs to our synthesizer are identical. In our case, both will be uniform elements from $\mathbb{Z}_p^{m \times m}$ (and R_p for rings) for some p that is of size at least $m^{\omega(q)}$. While we lose some efficiency due to the superpolynomial size of p, we gain a lot back because we can use the uniform distribution and simple rounding. The hardness of the LWE instances underlying these concrete synthesizers therefore follows from lemmas 8.1 and 8.2 (which we just showed earlier in this section) when coupled with theorems 2.4 and 2.5, respectively.

Theorem 8.3. *Let m, q, p, λ, and ℓ be integers. In words, m will be our lattice dimension, q is our modulus, p is our rounding parameter, λ is our PRF length, and ℓ is our synthesizer parameter. Let $\psi \in \mathbb{Z}$ be the uniform distribution over $\mathbb{Z}_p$, let $\boldsymbol{\psi} \in \mathbb{Z}^m$ be the noise distribution created by concatenating m samples from ψ, and let $\boldsymbol{\Psi} \in \mathbb{Z}^{m \times m}$ be the distribution created by concatenating m samples from $\boldsymbol{\psi}$.*

We additionally require that $q \geq p\ell\,(Cpm)^\ell \cdot m^{\omega(1)}$ for some universal constant C. Let the PRF P^ℓ defined by applying the synthesizer construction defined in definition 3.5 to the synthesizer K_ℓ defined in definition 7.1.

Any adversary that can distinguish P^ℓ from random as defined in definition 2.1 with non-negligible advantage ε can be used to solve the $\left(q, m, \mathcal{U}_p, \mathcal{U}_p^m, U_q^m\right)$-LWE problem with non-negligible advantage.

As long as we follow the parameter choices implied by theorem 8.3, we can build PRFs from K_ℓ for any choice of ℓ. In fact, as with S_ℓ, we could mix and match different values of ℓ in a single PRF, but again we do not see any logical reason to do so.

8.3 The Synthesizer K_2

We start by analyzing the synthesizer K_2. From theorem 8.3, we know that P^2–the PRF built using K_2–is secure as long as $q \geq p\ell \left(Cpm\right)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C.

Let's evaluate our PRF construction P^2. Since it must be the case that $q \geq p\ell \left(Cpm\right)^{\ell} \cdot m^{\omega(1)}$ which, since $p = m^{\omega(1)}$ is at most $C^2 m^{\omega(1)}$, we know that

$$\log q = O\left(\omega\left(1\right)\left(\log m\right)\right)$$

Efficiency Calculations. At level i of the synthesizer tree, we do $2 \cdot 2^{\log(\lambda)-i}$ matrix multiplications and $2^{\log(\lambda)-i}$ matrix rounding operations. If we sum over all levels of the tree, we perform $4\lambda - 1$ matrix multiplications and $2\lambda - 1$ matrix rounding operations. Since both our inputs and outputs can be thought of as uniform matrices over $\mathbb{Z}_p^{m \times m}$, the output of our synthesizer can be immediately used at the next level, and we do not have to spend any computational time reformatting this output.

Our matrices are of dimension size m. If we set τ to be the exponent corresponding to optimal matrix multiplication, this means our PRF can be computed in time $\tilde{O}\left(\lambda m^{\tau}\right)$. In addition, note that we output $m^2 \log p$ bits, so our operations per output bit is $\tilde{O}\left(\lambda m^{\tau-2}\right)$.

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of matrix operations if this is what we are only allowed (i.e. multiply, add, and round) and absolute complexity. Our PRF P^2 clearly has $O\left(\log \lambda\right)$ matrix operation complexity, since the longest potential path from root to leaf on our synthesizer tree has $2\log\left(\lambda\right)$ multiplies and $\log\left(\lambda\right)$ rounding operations. Since matrix multiplication is in NC^1 [RW04] and our synthesizer tree has depth $\log \lambda$, this means P^2 is in NC^2.

Key and Public Parameter Size. The one area that our construction F^2 does not do well on is key size. For our key, we need $\log \lambda$ uniform matrices in $\mathbb{Z}_q^{m \times m}$ as well as 2λ matrices in $\mathbb{Z}_p^{m \times m}$. We note that the $\mathbb{Z}_q^{m \times m}$ matrices can be made public without any loss of security (since they are what would traditionally be the public matrices of an LWE instance). This gives us public parameter sizes of $\log\left(\lambda\right)\left(m^2 \log q\right) = \tilde{O}\left(m^2\right)$ and secret key sizes of $2\lambda m^2 \log p = \tilde{O}\left(\lambda m^2\right)$.

8.4 The Synthesizer for Rings

We can easily adapt what we just did above to the ring setting, immediately getting the following lemma:

Theorem 8.4. *Let m, q, p, λ, and ℓ be integers. Let R be acyclotomic polynomial ring $R \stackrel{\mathsf{def}}{=} \mathbb{Z}\left[x\right]/\left(x^m + 1\right)$ where m is a power of 2. In words, q is our modulus, p is our rounding parameter, λ is our PRF length, and ℓ is our synthesizer parameter. Let $\boldsymbol{\psi} \in \mathbb{Z}$ be the uniform distribution over R_p.*

We additionally require that $q \geq p\ell \left(Cpm \cdot \omega\left(\sqrt{\log m}\right)\right)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C. Let the PRF $P^{R,\ell}$ defined by applying the synthesizer construction defined in definition 3.5 to the synthesizer $K_{R,\ell}$ defined in definition 7.2.

Any adversary that can distinguish P^{ℓ} from random as defined in definition 2.1 with non-negligible advantage ε can be used to solve the $\left(R, q, m, \mathcal{U}_p, \mathcal{U}_p^m, U_q^m\right)$-RLWE problem with non-negligible advantage.

8.5 The Synthesizer $K_{R,2}$

We continue by analyzing the ring-based synthesizer $K_{R,2}$. From lemma 8.4, we know that $P^{R,2}$–the PRF built using $K_{R,2}$–is secure as long as $q \geq p\ell \left(Cpm \cdot \omega\left(\sqrt{\log m}\right)\right)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C.

Let's evaluate our PRF construction $P^{R,2}$. Since it must be the case that $q \geq p\ell \left(Cpm \cdot \omega\left(\sqrt{\log m}\right)\right)^{\ell} \cdot m^{\omega(1)}$ which, since $p = m^{\omega(1)}$ is at most $C^2 m^{\omega(1)}$, we know that

$$\log q = O\left(\omega\left(1\right)\left(\log m\right)\right)$$

Efficiency Calculations. At level i of the synthesizer tree, we do $2 \cdot 2^{\log(\lambda)-i}$ ring multiplications and $2^{\log(\lambda)-i}$ ring rounding operations. If we sum over all levels of the tree, we perform $4\lambda - 1$ ring multiplications and $2\lambda - 1$ ring rounding operations. Since both our inputs and outputs can be thought of as uniform ring elements over R_p, the output of our synthesize can be immediately used at the next level, and we do not have to spend any computational time reformatting this output.

Our base ring R is of degree m. Since we can do modular ring multiplications in the power basis with $m \log m$ modular scalar operations [LPR10], this means our PRF can be computed in time $\tilde{O}\left(\lambda m\right)$. In addition, note that we output $m \log p$ bits, so our operations per output bit is just $\tilde{O}\left(\lambda\right)$.

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of ring operations if this is what we are only allowed (i.e. multiply, add, and round) and absolute complexity. Our PRF $P^{R,2}$ clearly has $O\left(\log \lambda\right)$ ring operation complexity, since the longest potential path from root to leaf on our synthesizer tree has $2\log\left(\lambda\right)$ multiplies and $\log\left(\lambda\right)$ rounding operations. Since ring multiplication is in NC^1 [RW04] and our synthesizer tree has depth $\log \lambda$, this means $P^{R,2}$ is in NC^2.

Key and Public Parameter Size. As with all of our synthesized PRFs, the one area that our construction $P^{R,2}$ does not do well on is key size. For our key, we need $\log \lambda$ uniform ring elements in R_q as well as 2λ ring elements that effecively live in R_p. We note that the R_q elements can be made public without any loss of security (since they are what would traditionally be the public matrices of an RLWE instance). This gives us public parameter sizes of $\log\left(\lambda\right)\left(m \log q\right) = \tilde{O}\left(m\right)$ and secret key sizes of $2\lambda m \log p = \tilde{O}\left(\lambda m\right)$.

8.6 Using K with larger ℓ

In section 10, we pick out synthesizers K_ℓ for larger choices of ℓ, analyze them, and point out what makes them interesting–namely, the circuit complexity for $K_{R,\ell}$.

9 Security of the Key Product Synthesizers K_ℓ and $K_{R,\ell}$

In this section, we discuss the security of our synthesizer K_ℓ. As we mentioned earlier, the security of K_ℓ follows almost exactly from the security of the direct PRF construction presented in [BPR12]. Suppose we call this PRF construction F_{BPR}. Recall that F_{BPR} has the following form:

Let m, q, p, and ℓ be integers such that $p \leq q$ and let $\mathbf{\Psi} \in \mathbb{Z}^{m \times m}$ be a noise distribution over matrices where each entry is B'-bounded. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers. For $i \in [1, \ell]$

and $j \in [0,1]$ let $\mathbf{S}_{i,j} \in \mathbb{Z}^{m \times m}$ be a matrix sampled randomly from $\mathbf{\Psi}$. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times m}$ be a matrix sampled uniformly at random.

The function $F_{BPR} : \{0,1\}^\ell \to \mathbb{Z}_p^{m \times m}$ is defined in the following way: for each $m \times m$ block of output of K_ℓ, define

$$F_{BPR}(\mathbf{x}) \stackrel{\text{def}}{=} \left[\mathbf{A} \left[\prod_{i=1}^{\ell} \mathbf{S}_{i,x_i} \right] \right]_p \tag{9.1}$$

Note that this is almost exactly the definition of K_ℓ. The only difference is that K_ℓ generalizes F_{BPR} by having several choices of matrix for each input 'position' rather than just two. If we generalized F_{BPR} to have higher-order inputs (one example would be input 'bytes' that had 8 matrices, one for each input value) we would exactly get K_ℓ. This only changes the security proof *very* slightly–we just need to simulate more LWE instances at each level (corresponding to the additional keys) and we are done. Thus, security follows immediately from the security proof of F_{BPR}. We state this in a lemma below, which is essentially just theorem 5.2 from [BPR12] with slightly different notation.

Lemma 9.1. *Let K_ℓ be a synthesizer defined as in definition 7.1 with all associated parameters. Let ψ be a 'noise' distribution over $\mathbb{Z}$ such that ψ is B-bounded, and let $\boldsymbol{\psi}$ be the 'noise' distribution over $\mathbb{Z}^m$ where each entry is sampled independently from ψ. Let $q \geq p\ell (CBm)^\ell \cdot m^{\omega(1)}$ for some universal constant C.*

Any adversary $\mathcal{A}$ that can win the pseudorandom synthesizer game for K_ℓ defined in definition 3.2 can be used to solve the $\left(q, m, \psi, \boldsymbol{\psi}, \mathcal{U}_q^m\right) - \text{LWE}$ problem.

We note that this immediately translates to 'standard' LWE hardness via theorem 2.4. In addition, we can state a similar theorem for $K_{R,\ell}$, our ring LWE-based synthesizer. The following is a essentially a restatement of theorem 5.3 from [BPR12] to mesh with our notation:

Lemma 9.2. *Let $K_{R,\ell}$ be a synthesizer defined as in definition 7.2 with all associated parameters. Let $\boldsymbol{\psi}$ be a 'noise' distribution over R such that ψ is B-bounded. Let $q \geq p\ell \left(Bm \cdot \omega\left(\sqrt{\log m}\right)\right)^\ell \cdot m^{\omega(1)}$.*

Any adversary $\mathcal{A}$ that can win the pseudorandom synthesizer game for $K_{R,\ell}$ defined in definition 3.2 can be used to solve the $(R, q, m, \boldsymbol{\psi}, \boldsymbol{\psi}, \mathcal{U}(R_q))$-Ring LWE problem.

We again note that this immediately translates to typical ring LWE hardness via theorem 2.5.

10 Constructions of PRFs with Large ℓ

In this section we use our synthesizers K_ℓ and $K_{R,\ell}$ to build two PRFs for two choices of ℓ other than two and analyze these PRFs. In particular, we choose larger choices of ℓ that correspond to different hardness assumptions of the underlying LWE problem from which we reduce our PRF hardness. These two PRFs don't seem to have many practical applications, but we think that they are interesting to examine from a circuit complexity perspective. We don't bother to compute some of the parameter sizes because they are complicated and not particularly interesting.

The main point of building these new PRFs is the circuit complexity. Suppose we again denote the main direct PRF construction of [BPR12] by F_{BPR}. In [BPR12], the authors cleverly show how to compute the ring version of this PRF $F_{R,BPR}$ in NC^1. Without too much work, we can show how to adapt their techniques to our ring-based synthesizer $K_{R,\ell}$.

The basic idea is the following: let $\mathbf{a}, \mathbf{s}_1, ..., \mathbf{s}_\ell \in R_q$ be ring elements, where R is a cyclotomic polynomial ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/(x^n+1)$ such that m is a power of 2. When we are attempting to compute a subset product over rings $\mathbf{a}\left[\prod_{i=1}^{\ell} \mathbf{s}_i\right]$, we can store all of these ring elements as vectors in $\mathbb{Z}_q^m$ using the discrete Fourier transform or the 'Chinese remainder' representation modulo q (see [LPR10] for more details on this exact formulation).

To do this, we just evaluate the ring elements $\mathbf{a}, \mathbf{s}_1, ..., \mathbf{s}_\ell$ as polynomials at the m roots of our cyclotomic polynomial $x^m + 1 \mod q$. This allows us to compute the multiplication of two ring elements as a coordinate-wise product of their two vectors. Then, to evaluate our overall subset product, we would only need to compute a subset product element-wise on the appropriate vectors. We could then finish up and move back to the power-basis representation using an m-dimensional Fourier transform over $\mathbb{Z}_q$. At this point, we could round our result as required by our constructions.

This gives us essentially four steps for each synthesizer computation: converting the input to a 'Chinese remainder' form, computing the subset product, returning to the power basis, and then rounding. We can use a fast Fourier transform to convert the input to a 'Chinese remainder' form in circuit depth $O(\log m)$ [SS91]. Iterated subset product over the integers is also in NC^1 [IL95], as is the discrete Fourier transform that allows us to move back to the power basis [RT92]. Rounding is also an NC^1 operation.

Thus, we can compute our synthesizer $K_{R,\ell}$ in NC^1. While this isn't particularly surprising given that $F_{R,BPR}$ can be computed in NC^1, it is important that we can chain our calculations. If we want our synthesizers to compose gracefully into PRFs with low circuit complexity, we cannot do any preprocessing operations that are not in NC^1. While we cannot do some of the further optimizations discussed in [BPR12] like computing/having discrete logs, we can still fit everything in an NC^1 circuit since the only preprocessing we need is to convert the ring elements we are given into the 'Chinese remainder' representation.

With all of this in mind, we discuss some synthesizers that utilize this fact.

10.1 The Synthesizer $K_{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}}$

We next analyze a synthesizer that doesn't seem to have much use (because it is over the integers, and not rings): $K_{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}}$. The notable thing about this synthesizer is it allows a PRF tree of depth $\sqrt{\log \lambda}$. From theorem 8.3, we know that $P^{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}}$–the PRF built using $K_{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}}$– is secure as long as $q \geq p\ell\,(Cpm)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C. We once again set $p = m^{\omega(1)}$.

Let's evaluate our PRF construction $P^{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}}$. We start by noting that, due to our choice of parameters, $q \geq p\,(Cpm)^{\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}} \cdot m^{\omega(1)}$, which, although it is superpolynomially large, still is smaller than m^{λ^c} for any constant $c \geq 0$. We can use this fact to evaluate $\log q$:

$$\log q = O\left(\left(\omega(1)\,\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)} + \omega(1)\right)\log(m)\right) = O\left(\omega(1)\,\lambda^{\left(\frac{1}{\sqrt{\log \lambda}}\right)}\log m\right)$$

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of matrix operations if this is what we are only allowed (i.e. multiply, add, and round) and circuit complexity. From a circuit complexity standpoint, this PRF is in NC^2 since matrix

multiplication is in NC^1. In addition, $P^{\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$ clearly has $O(\log\lambda)$ parallel matrix operation complexity, since, in practice and if we are only allowed matrix operations, we will need to compute all of the matrix multiplications in a tree-like way. This is the case for all of our PRF constructions (and all known lattice-based PRF constructions that are not more sequential).

10.2 The Synthesizer $K_{R,\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$

We next analyze a synthesizer that will give us a parallel construction with very interesting properties: $K_{R,\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$. The notable thing about this synthesizer is it allows a PRF tree of depth $\sqrt{\log\lambda}$. From theorem 8.4, we know that $P^{R,\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$–the PRF built using $K_{R,\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$– is secure as long as $q \geq p\ell\left(Cpm\right)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C. We once again set $p = m^{\omega(1)}$.

Let's evaluate our PRF construction $P^{R,\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$. We start by noting that, due to our choice of parameters, $q \geq p\left(Cpm\right)^{\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}} \cdot m^{\omega(1)}$, which, although it is superpolynomially large, still is smaller than m^{λ^c} for any constant $c \geq 0$. We can use this fact to evaluate $\log q$:

$$\log q = O\left(\left(\omega\left(1\right)\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)} + \omega\left(1\right)\right)\log\left(m\right)\right) = O\left(\omega\left(1\right)\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}\log m\right)$$

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of ring operations if this is what we are only allowed (i.e. multiply, add, and round) and circuit complexity. The circuit complexity is where things get interesting. Note that we have $\sqrt{\log\lambda}$ layers of synthesizers, each of which can be computed in NC^1 as we discussed in the beginning of this section. Moreover, these synthesizers compose without any additional complexity. Thus, we achieve an $NC^{1.5}$ circuit complexity[10].

In addition, $P^{\lambda^{\left(\frac{1}{\sqrt{\log\lambda}}\right)}}$ clearly has $O(\log\lambda)$ parallel matrix operation complexity, since, in practice and if we are only allowed matrix operations, we will need to compute all of the matrix multiplications in a tree-like way. This is the case for all of our PRF constructions (and all known lattice-based PRF constructions that are not more sequential).

10.3 The Synthesizer K_{λ^c}

The penultimate synthesizer we examine will allow us to build PRFs that aren't very interesting but that do have interesting ring variants. For some constant $c \in (0,1)$, the PRF P^{λ^c} has tree depth of only $\frac{1}{c}$ in terms of calls to our synthesizer K_{m^λ} in order to obtain a PRF with λ output bits. From theorem 8.3, we know that P^{λ^c}–the PRF built using K_{λ^c}–is secure as long as $q \geq p\ell\left(Cpm\right)^{\ell} \cdot m^{\omega(1)}$ for some universal constant C. We once again set $p = m^{\omega(1)}$.

Let's evaluate our PRF construction P^{λ^c}. We start by noting that, due to our choice of parameters, $q \geq p\ell\left(Cpm\right)^{\lambda^c} \cdot m^{\omega(1)}$, which is subexponetially large. However, note that c can be an arbitrarily small constant. We can use this fact to evaluate $\log q$:

$$\log q = O\left(\left(\omega\left(1\right)\lambda^c + \omega\left(1\right)\right)\log\left(m\right)\right) = O\left(\omega\left(1\right)\lambda^c\log\left(m\right)\right)$$

[10] Yes, this is a slight abuse of notation, but we think it is illustrative.

Parallel Complexity. This PRF does not have very interesting complexity, unfortunately. We only have synthesizer tree depth of $\frac{1}{c}$ for some constant $c \in (0,1)$, but we have to do quite a few matrix multiplies in each synthesizer computation. Our PRF parallel complexity is still NC^2 in this case.

As with all of our PRFs, F^{λ^c} clearly has $O(\log \lambda)$ matrix operation complexity, since, in practice and if we are only allowed matrix operations, we will need to compute all of the matrix multiplications in a tree-like way.

10.4 The Synthesizer K_{R,λ^c}

The final synthesizer we examine will allow us to build maximally parallel PRFs with the strongest assumptions possible. For some constant $c \in (0,1)$, the PRF P^{R,λ^c} has tree depth of only $\frac{1}{c}$ in terms of calls to our synthesizer K_{R,m^λ} in order to obtain a PRF with λ output bits. From theorem 8.4, we know that P^{R,λ^c}–the PRF built using K_{R,λ^c}–is secure as long as $q \geq p\ell \left(Cpm\right)^\ell \cdot m^{\omega(1)}$ for some universal constant C. We once again set $p = m^{\omega(1)}$.

Let's evaluate our PRF construction P^{R,λ^c}. We start by noting that, due to our choice of parameters, $q \geq p\ell \left(Cpm\right)^{\lambda^c} \cdot m^{\omega(1)}$, which is subexponetially large. However, note that c can be an arbitrarily small constant. We can use this fact to evaluate $\log q$:

$$\log q = O\left(\left(\omega\left(1\right)\lambda^c + \omega\left(1\right)\right)\log\left(m\right)\right) = O\left(\omega\left(1\right)\lambda^c \log\left(m\right)\right)$$

Parallel Complexity. This PRF has quite interesting parallel complexity. We only have synthesizer tree depth of $\frac{1}{c}$ for some constant $c \in (0,1)$, which means that we only have to compute a constant number of synthesizers in any given branch of our PRF calculation. Since we showed that each synthesizer of this form can be computed in NC^1, we know that the parallel complexity in this case is still NC^1. To our knowledge, this is the weakest LWE assumption (smallest modulus) that is known to admit PRFs in NC^1.

As with all of our PRFs, F^{λ^c} clearly has $O(\log \lambda)$ ring operation complexity, since, in practice and if we are only allowed matrix operations, we will need to compute all of the matrix multiplications in a tree-like way.

11 Tree-Based Pseudorandom Synthesizer

In this section we build our most general (and complicated) pseduorandom synthesizer T_ℓ and its ring version $T_{R,\ell}$, which are based upon the general PRF construction in [BP14]. There are only two slight differences between T_ℓ and the PRF from [BP14]: first, T_ℓ has an arbitrary polynomial amount of input matrices at each layer (instead of just two for the whole PRF). In addition, in order to make T_ℓ compose nicely, we need to use a non-square secret matrix $\mathbf{S}$ instead of a vector or square matrix (this allows us to extract more entropy from our outputs), which amounts to just running multiple instances of the original PRF in parallel with different keys. This is in fact mentioned as an optimization in the original work [BP14] since it allows for a better ratio of output bits to computation time.

These differences don't change the fundamental structure of the function (and the security proof is essentially identical) but do make bookkeeping substantially more difficult. In addition, the syntax of the construction will look more complicated than the [BP14] PRF construction, but we again want to emphasize that they are actually very, very similar.

11.1 Synthesizer Construction

Definition 11.1. *Let m, c, q, p, and ℓ be integers such that $p \leq q$. Let $\mathcal{T}$ be a binary tree such that $\ell = |\mathcal{T}|$. Suppose we order the leaf nodes of T in order from left to right, assigning each one an integer in $[1, \ell]$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers, and let $\mathbf{z} \in \mathbb{Z}^\ell$ be a vector such that $\mathbf{z}_i \in [1, k_\ell]$ For $i \in [1, \ell]$ and $j \in [1, k_i]$ let $\mathbf{A}_{i,j} \in \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$ be matrices selected uniformly at random. Let $\mathbf{S} \in \mathbb{Z}_q^{cm \times m}$ be a matrix sampled uniformly at random.*

Let the tree-navigating function $\mathbf{A}_\mathcal{T} : [1, k_1] \times [1, k_2] \times ... \times [1, k_\ell] \to \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$ be defined in the following way:

$$\mathbf{A}_\mathcal{T}(\mathbf{y}) = \left\{ \begin{array}{cc} \mathbf{A}_y & \text{if } |\mathcal{T}| = 1 \\ \mathbf{A}_{\mathcal{T}_l(\mathbf{y}_l)} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{\mathcal{T}_r(\mathbf{y}_r)}\right) & \text{otherwise} \end{array} \right\}$$

where $\mathbf{y}_l$ corresponds to the portion of the vector $\mathbf{y}$ with indices corresponding to leaf nodes in $\mathcal{T}_l$ and $\mathbf{y}_r$ corresponds to the portion of $\mathbf{y}$ with indices corresponding to leaf nodes in $\mathcal{T}_r$.

We define the synthesizer $T_\ell : \mathbb{Z}_q^{cm \times m} \times \left[\left(\mathbb{Z}_2^{m \times m}\right)^{k_1} \times ... \times \left(\mathbb{Z}_2^{m \times m}\right)^{k_\ell}\right] \to \mathbb{Z}_p^{c(k_1 m \times ... \times k_\ell m)\lceil \log q \rceil}$ in the following way: for each $c\,(m \times m)$ block of output of S_ℓ, define

$$T_\ell(x_1, ...x_\ell) \stackrel{\text{def}}{=} \lceil \mathbf{S} \cdot \mathbf{A}_\mathcal{T}(\mathbf{x}) \rfloor_p \tag{11.1}$$

where $\mathbf{x}$ is the vector made by the concatentation of the x_is and each $x_i \in [1, k_i]$.

We now offer some comments on our synthezier T_ℓ. First, note that the 'key' $\mathbf{S}$ is of larger dimension than the matrices $\mathbf{A}$. This is because we need to be able to have an output with higher dimension than $m \times m$ if our inputs are uniformly random matrices over $\mathbb{Z}_q^{m \times m}$ and we are rounding.

We note that security follows almost immediately from the PRF construction of [BP14]. We discuss this more in section 13. In addition, we can define a ring form of this synthesizer as well.

Definition 11.2. *Let m, c, q, p, and ℓ be integers such that $p \leq q$, and let R be a cyclotomic polynomial ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/(x^n + 1)$ where m is a power of 2. Let $\mathcal{T}$ be a binary tree such that $\ell = |\mathcal{T}|$. Suppose we order the leaf nodes of T in order from left to right, assigning each one an integer in $[1, \ell]$. Let $k_1, ..., k_\ell \in \mathbb{Z}$ be positive integers, and let $\mathbf{z} \in \mathbb{Z}^\ell$ be a vector such that $\mathbf{z}_i \in [1, k_\ell]$ For $i \in [1, \ell]$ and $j \in [1, k_i]$ let $\mathbf{A}_{i,j} \in R_q^{\lceil \log q \rceil}$ be* row vectors *of rings selected uniformly at random. Let $\mathbf{S} \in R_q^c$ be a* column vector *of rings sampled uniformly at random.*

Let the tree-navigating function $\mathbf{A}_\mathcal{T} : [1, k_1] \times [1, k_2] \times ... \times [1, k_\ell] \to R_q^{\lceil \log q \rceil}$ be defined in the following way:

$$\mathbf{A}_\mathcal{T}(\mathbf{y}) = \left\{ \begin{array}{cc} \mathbf{A}_x & \text{if } |\mathcal{T}| = 1 \\ \mathbf{A}_{\mathcal{T}_l(\mathbf{y}_l)} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{\mathcal{T}_r(\mathbf{y}_r)}\right) & \text{otherwise} \end{array} \right\}$$

where $\mathbf{y}_l$ corresponds to the portion of the vector $\mathbf{y}$ with indices corresponding to leaf nodes in $\mathcal{T}_l$ and $\mathbf{y}_r$ corresponds to the portion of $\mathbf{y}$ with indices corresponding to leaf nodes in $\mathcal{T}_r$.

We define the synthesizer $T_{R,\ell} : R_q^c \times \left[(R_q)^{k_1} \times ... \times (R_q)^{k_\ell}\right] \to R_p^{c(k_1 \times ... \times k_\ell)\lceil \log q \rceil}$ in the following way: for each R_p^c-block of output of T_ℓ, define

$$T_{R,\ell}(x_1, ...x_\ell) \stackrel{\text{def}}{=} \lceil \mathbf{S} \cdot \mathbf{A}_\mathcal{T}(\mathbf{x}) \rfloor_p \tag{11.2}$$

where $\mathbf{x}$ is the vector made by the concatentation of the x_is and each $x_i \in [1, k_i]$.

12 Constructions of PRFs from T_ℓ and $T_{R,\ell}$

In this section we show how our synthesizer T_ℓ can be used to build PRFs.

12.1 Synthesizer Statement

While we have proven T_ℓ and $T_{R,\ell}$ secure for a wide variety of parameters, we have one issue left: our synthesizers don't compose perfectly because our input and output don't exactly match: our inputs are matrices in $\mathbb{Z}_q^{m\times m\lceil\log q\rceil}$ and row vectors of rings $R_q^{\lceil\log q\rceil}$, respectively, and our outputs are matrices in $\mathbb{Z}_p^{cm\times m\lceil\log q\rceil}$ and vectors of rings $R_p^{c\times\lceil\log q\rceil}$. If we set $p=\frac{q}{c}$ for some small constant c, then it becomes easy to solve this issue: we can just combine multiple $\mathbb{Z}_p$ (or R_p) elements to form a $\mathbb{Z}_q$ or (R_q) element in the straightforward manner. With this in mind, building a PRF from T_ℓ becomes simple.

As with S_ℓ and K_ℓ, we state an overall theorem about the security of PRFs constructed from our pseudorandom synthesizers using the synthesizer construction of [NR95]. This theorem follows almost immediately from applying the synthesizer construction security theorem of [NR95] as we stated in theorem 3.6 to our theorem 13.1 proving our synthesizer T_ℓ secure.

Theorem 12.1. *Let T_ℓ be a synthesizer defined as in definition 11.1 with all associated parameters, and let λ be an integer. Let ψ be a 'noise' distribution over $\mathbb{Z}$ such that ψ is B-bounded. Let $q\geq pB\ell\left(m\left\lceil\log q\right\rceil\right)^{e(\mathcal{T})}\cdot m^{\omega(1)}$ such that $\frac{q}{p}=c$. In words, m will be our lattice dimension, q is our modulus, p is our rounding parameter, λ is our PRF length, and ℓ is our synthesizer parameter.*

*Let the PRF B^ℓ defined by applying the synthesizer construction defined in definition 3.5 to the synthesizer T_ℓ defined in definition 11.1. Any adversary that can distinguish B^ℓ from random as defined in definition 2.1 with non-negligible advantage ε can be used to solve the $\left(q,m,\psi,\mathcal{U}_q^m,\mathcal{U}_q^m\right)$-*LWE *problem with non-negligible advantage.*

As long as we follow the parameter choices implied by lemma 13.1, we can build PRFs from T_ℓ for any choice of ℓ. In fact, we can make PRFs by mixing and matching different size synthesizers T_ℓ and even different trees $\mathcal{T}$, but we don't really see any applications that would really benefit from this sort of variety (perhaps there could be some parallel PRF evaluation with odd hardware restrictions, but this seems a bit farfetched.

In the rest of the section, we examine the PRF B^2, which we build from T_2, and the PRF $B^{R,2}$, which we build from $T_{R,2}$. These seem to be the most useful parameter choices we can make with this generic construction, although there certainly could be many more useful constructions that we have simply overlooked.

12.2 The Synthesizer T_2

We start by analyzing the simplest, and yet one of the most efficient, synthesizers: T_2. From theorem 12.1, we know that B^2–the PRF built using T_2–is secure as long as $q\geq pB\ell\left(m\left\lceil\log q\right\rceil\right)^{e(\mathcal{T})}\cdot m^{\omega(1)}$.

Let's evaluate our PRF construction [2]. For our lemmas to hold, it must be the case that $q\geq pB\ell\left(m\left\lceil\log q\right\rceil\right)^{e(\mathcal{T})}\cdot m^{\omega(1)}$. we know that

$$\log q=O\left(\omega\left(1\right)\left(\log m\right)+\log p+\log B+\log\ell\right)$$

Efficiency Calculations. While the full tree construction of T_ℓ can be complicated to evaluate (see the original work [BP14] for more details), the construction is much simpler when we restrict trees to two leaf nodes as we do for this choice of ℓ. At level i of the synthesizer tree, we effectively do $2^{\log(\lambda)-i}$ computations of T_2.

For each T_2 calculation, we have to do one G^{-1} calculation over a matrix of size $\mathbb{Z}_q^{m\times m\lceil\log q\rceil}$ and then two matrix multiplications (of dimensions consist of constant or logarithmic numbers of $m \times m$ blocks). If we sum over all of the levels of the tree, we end up going $2n-1$ calculations of T_2 and $n-1$ operations converting multiplie outputs over $\mathbb{Z}_p$ to a single output over $\mathbb{Z}_q$, the latter of which requires substantially less computation than the former.

Our matrices are of dimension size $\tilde{O}(m)$. If we set τ to be the exponent corresponding to optimal matrix multiplication, this means our PRF can be computed in time $\tilde{O}(\lambda m^\tau)$. In addition, note that we output $\tilde{O}\left(m^2\right)$ bits, so our operations per output bit is $\tilde{O}\left(\lambda m^{\tau-2}\right)$.

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of matrix operations if this is what we are only allowed (i.e. multiply, add, and round) and absolute complexity. Our PRF B^2 clearly has $O\log(\lambda)$ matrix operation complexity, since the longest potential path from root to leaf on our synthesizer tree has $2\log(\lambda)$ multiplies and $\log(\lambda)$ rounding operations. Since matrix multiplication is in NC^1 [RW04] and our synthesizer tree has depth $\log\lambda$, this means B^2 is in NC^2.

Key and Public Parameter Size. The one area that our construction B^2 does not do well on is key size. For our key, we need $\log\lambda$ uniform matrices in $\mathbb{Z}_q^{cm\times m}$ as well as 2λ matrices in $\mathbb{Z}_q^{m\times m\lceil\log q\rceil}$. We note that the $\mathbb{Z}_q^{m\times m\lceil\log q\rceil}$ matrices can be made public without any loss of security (since they are what would traditionally be the public matrices of an LWE instance). This gives us public parameter sizes of $\tilde{O}\left(\lambda m^2\right)$ and secret key sizes of $\tilde{O}\left(m^2\right)$.

12.3 The Synthesizer for Rings

Like K_ℓ, we can easily adapt what we just did above to the ring setting, immediately getting the following lemma:

Theorem 12.2. *Let $T_{R,\ell}$ be a synthesizer defined as in definition 11.1 with all associated parameters, and let λ be an integer. Let $\boldsymbol{\psi}$ be a 'noise' distribution over the ring R such that $\boldsymbol{\psi}$ is B-bounded. Let $q \geq pB\ell\left(m\lceil\log q\rceil\right)^{e(\mathcal{T})}\cdot m^{\omega(1)}$ with $\frac{q}{p}=c$. In words, m will be the degree of our polynomial ring, q is our modulus, p is our rounding parameter, λ is our PRF length, and ℓ is our synthesizer parameter.*

Let the PRF $B^{R,\ell}$ defined by applying the synthesizer construction defined in definition 3.5 to the synthesizer $T_{R,\ell}$ defined in definition 11.2. Any adversary that can distinguish $B^{R,\ell}$ from random as defined in definition 2.1 with non-negligible advantage ε can be used to solve the $(R,q,m,\psi,\mathcal{U}(R_q),\mathcal{U}(R_q))$-Ring LWE *problem with non-negligible advantage.*

As long as we follow the parameter choices implied by lemma 13.1, we can build PRFs from $T_{R,\ell}$ for any choice of ℓ. In fact, we can make PRFs by mixing and matching different size synthesizers $T_{R,\ell}$ and even different trees $\mathcal{T}$, but we don't really see any applications that would really benefit from this sort of variety (perhaps there could be some parallel PRF evaluation with odd hardware restrictions, but this seems a bit farfetched.

12.4 The Synthesizer $T_{R,2}$

From theorem 12.1, we know that $B^{R,2}$–the PRF built using $T_{R,2}$–is secure as long as $q \geq pB\ell \left(m \left\lceil \log q \right\rceil\right)^{e(\mathcal{T})} \cdot m^{\omega(1)}$.

Let's evaluate our PRF construction B^2. For our lemmas to hold, it must be the case that $q \geq pB\ell \left(m \left\lceil \log q \right\rceil\right)^{e(\mathcal{T})} \cdot m^{\omega(1)}$. we know that

$$\log q = O\left(\omega\left(1\right)\left(\log m\right) + \log p + \log B + \log \ell\right)$$

Efficiency Calculations. While the full tree construction of T_ℓ can be complicated to evaluate (see the original work [BP14] for more details), the construction is much simpler when we restrict trees to two leaf nodes as we do for this choice of ℓ. At level i of the synthesizer tree, we effectively do $2^{\log(\lambda)-i}$ computations of $T_{R,2}$.

For each $T_{R,2}$ calculation, we have to do one G^{-1} calculation over a row vector of rings $R_q^{\lceil \log q \rceil}$ and then a logarithmic number of ring multiplications. If we sum over all of the levels of the tree, we end up going $2n-1$ calculations of $T_{R,2}$ and $n-1$ operations converting multiple outputs over R_p to a single output over R_q, the latter of which requires substantially less computation than the former.

Our ring elements are of degree m. Since we can do modular ring multiplications in the power basis with $m \log m$ modular scalar operations [LPR10], this means our PRF can be computed in time $\tilde{O}\left(\lambda m\right)$. In addition, note that we output $cm \log p$ bits, so our operations per output bit is $\tilde{O}\left(\lambda\right)$.

Parallel Complexity. We subdivide parallel complexity into two categories: complexity in terms of matrix operations if this is what we are only allowed (i.e. multiply, add, and round) and absolute complexity. Our PRF $B^{R,2}$ clearly has $O \log\left(\lambda\right)$ ring operation complexity, since the longest potential path from root to leaf on our synthesizer tree has $2\log\left(\lambda\right)$ multiplies and $\log\left(\lambda\right)$ rounding operations. Since ring element multiplication is in NC^1 [RW04] and our synthesizer tree has depth $\log \lambda$, this means $B^{R,2}$ is in NC^2.

Key and Public Parameter Size. We repeat, ad nauseam: the one area that our construction $B^{R,2}$ does not do well on is key size. For our key, we need $c \log \lambda$ uniform ring elements in R_q as well as $2\lambda \left\lceil \log q \right\rceil$ ring elements in R_q. We note that the latter $\tilde{O}\left(\lambda\right)$ ring elements can be made public without any loss of security (since they are what would traditionally be the public matrices of a ring LWE instance). This gives us public parameter sizes of $\tilde{O}\left(\lambda m\right)$ and secret key sizes of $\tilde{O}\left(m\right)$.

13 Security of the Tree Synthesizers T_ℓ and $T_{R,\ell}$

In this section, we discuss the security of our synthesizer T_ℓ and its ring variant $T_{R,\ell}$. As we mentioned earlier, the security of T_ℓ follows almost exactly from the security of the family of PRF constructions presented in [BP14]. Suppose we call this PRF construction F_{BP}. Recall that F_{BP} has the following form:

Let m, q, and p be integers, and let $\mathcal{T}$ be a non-empty binary tree with ℓ leaf nodes. Let the matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$ be chosen uniformly at random. Let the function $\mathbf{A}_\mathcal{T} : \{0,1\}^{|\mathcal{T}|} \to \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$ be defined recursively as follows:

$$\mathbf{A}_\mathcal{T}\left(\mathbf{x}\right) = \left\{ \begin{array}{cc} \mathbf{A}_x & \text{if } |\mathcal{T}| = 1 \\ \mathbf{A}_{\mathcal{T}_l(\mathbf{x}_l)} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{\mathcal{T}_r(\mathbf{x}_r)}\right) & \text{otherwise} \end{array} \right\}$$

where $\mathbf{x}$ can be parsed as $\mathbf{x} = \mathbf{x}_l || \mathbf{x}_r$ for $\mathbf{x}_l \in \{0,1\}^{|\mathcal{T}_l|}$ and $\mathbf{x}_r \in \{0,1\}^{|\mathcal{T}_r|}$.

The function $F_{BP} : \{0,1\}^{\ell} \rightarrow \mathbb{Z}_p^{m\lceil \log q \rceil}$ is defined in the following way:

$$F_{BP}(\mathbf{x}) \stackrel{\mathsf{def}}{=} \lceil \mathbf{s}^{\intercal} \mathbf{A}_{\mathcal{T}}(\mathbf{x}) \rfloor_p \tag{13.1}$$

There are only two differences between T_ℓ and F_{BP}. First, we use a matrix $\mathbf{S}$ as a secret key in $\mathcal{T}_\ell$ instead of a vector $\mathbf{s}$ as in F_{BP}. The authors of [BP14] actually allude to the fact that this can actually be used as a technique to gain more output bits per computational work in their paper, and it does not change the security proof substantially (a proof would need a hybrid argument over multiple LWE instances, but this is widely presented without a formal proof and sometimes the LWE problem itself is even defined in this way).

The only other difference is how we handle leaf nodes. In F_{BP}, once we reach a leaf node in the recursive tree evaluation, we choose one of two input matrices–either $\mathbf{A}_0$ or $\mathbf{A}_1$–depending on a single input bit. In T_ℓ, when we reach a leaf node we choose one of many different input matrices based on a particular index in a larger set. This does not substantively change the security proof–we just query for more LWE samples at each level of the hybrid in the obvious manner. In addition, our leaf nodes have different sets of matrices, whereas all leaf nodes in F_{BP} use the same two matrices. This is essentially just an optimization that the authors of [BP14] use in order to reduce the public parameter size.

Thus, the security proof of T_ℓ follows immediately from that of F_{BP}. We state this in a lemma below, which is essentially just a restatement of theorem 2.3 from [BP14].

Lemma 13.1. *Let T_ℓ be a synthesizer defined as in definition 11.1 with all associated parameters. Let ψ be a 'noise' distribution over $\mathbb{Z}$ such that ψ is B-bounded. Let $q \geq pB\ell(m\lceil \log q \rceil)^{e(\mathcal{T})} \cdot m^{\omega(1)}$.*

Any adversary $\mathcal{A}$ that can win the pseudorandom synthesizer game for T_ℓ defined in definition 3.2 can be used to solve the $(q, m, \psi, \mathcal{U}_q^m, \mathcal{U}_q^m) - \mathrm{LWE}$ problem.

We can also state a similar lemma for $T_{R,\ell}$, our ring LWE-based synthesizer.

Lemma 13.2. *Let $T_{R,\ell}$ be a synthesizer defined as in definition 11.2 with all associated parameters. Let $\boldsymbol{\psi}$ be a 'noise' distribution over R such that $\boldsymbol{\psi}$ is B-bounded. Let $q \geq pB\ell(m\lceil \log q \rceil)^{e(\mathcal{T})} \cdot m^{\omega(1)}$.*

Any adversary $\mathcal{A}$ that can win the pseudorandom synthesizer game for $T_{R,\ell}$ defined in definition 3.2 can be used to solve the $(R, q, m, \boldsymbol{\psi}, \mathcal{U}(R_q), \mathcal{U}(R_q)) - RLWE$ problem.

14 Conclusion and Open Problems

In this paper, we showed how to build more efficient lattice-based PRFs using keyed pseudorandom synthesizers. We constructed PRFs with only slightly superpolynomial modulus (independent of the number of input bits) that match the efficiency of the otherwise most efficient known constructions. We also show how to build other PRFs that imply interesting results on the parallel circuit complexity of lattice PRFs.

14.1 Open Problems

We conclude the paper by stating two open problems. We think these problems are very important for lattice cryptography in general, as well as (obviously) PRFs.

LWR with Polynomial Modulus. Recall that the learning with rounding problem states that, informally, it is hard to distinguish samples of the form $\left(\mathbf{a}_i, \lceil \mathbf{a}_i{}^\intercal \mathbf{s} \rfloor_p\right)$ from random for uniformly random samples $\mathbf{a}_i$ and secret key $\mathbf{s}$. For an unbounded number of samples (what is needed for a PRF reduction), the hardness of LWR is only known when the modulus q of the problem instance is superpolynomially large. This doesn't seem natural to us, and we think attempting to prove that LWR is hard for a polynomial modulus and unbounded samples (or showing evidence that this is not true, although we consider this unlikely) is a worthwhile endeavor. While there have been several papers establishing the hardness of LWR with polynomial hardness and a *fixed* numbers of samples [AKPW13] [BGM+16] [BLL+15] [AA16], the security of LWR with a polynomial modulus and unbounded polynomial samples is still unknown.

Earlier this year, Montgomery [Mon18] showed how to build a (highly nonstandard) variant of learning with rounding with polynomial modulus and unbounded samples. However, at a first glance, this new LWR variant does not seem to be able to be used with our techniques (or any others, for that matter) to build PRFs with polynomial modulus. Constructing a variant of LWR that can be used to build PRFs with polynomial modulus (or showing a security proof with polynomial modulus and unbounded samples for the regular version of LWR) remains an important open problem in our opinion.

Subset Product LWE with Polynomial Modulus. Let $\mathbf{A}_{i,b_i} \in \mathbb{Z}_q^{m \times m}$ for $i \in [1, ...\lambda]$ and $b_i \in \{0, 1\}$ be matrices selected uniformly at random, and let $\mathbf{s} \in \mathbb{Z}_q^m$ also be sampled uniformly from random. We call the following function subset product LWE:

$$F\left(x_1 ... x_\lambda\right) = \prod_{i=1}^{\lambda} \mathbf{A}_{i,x_i} \mathbf{s} + \boldsymbol{\delta}_{\mathbf{x}}$$

where $\boldsymbol{\delta}_{\mathbf{x}}$ is a noise vector selected independently at random for each input $\mathbf{x}$.

Currently, for polynomial λ, this function F can only be shown to be hard for modulus on the order of m^λ. Like LWR, we see no real reason why this problem should not be hard for a polynomial modulus: there are no known attacks, and the large modulus seems to be a relic of the hybrid argument in the proofs. We think attempting to prove this function secure for smaller modulus (and thus achieve better lattice hardness results) is a great candidate for future research.

References

[AA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from LWE to LWR. *IACR Cryptology ePrint Archive*, 2016:589, 2016.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Ban15] Abhishek Banerjee. New constructions of cryptographic pseudorandom functions. Ph.D. Thesis, 2015. https://smartech.gatech.edu/bitstream/handle/1853/53916/BANERJEE-DISSERTATION-2015.pdf?sequence=1&isAllowed=y.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th Annual Symposium on Foundations of Computer Science*, pages 514–523, Burlington, Vermont, October 14–16, 1996. IEEE Computer Society Press.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.

[BFP+15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 31–60, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 209–224, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. Lecture Notes in Computer Science, pages 415–445. Springer, Heidelberg, Germany, 2017.

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 3–24, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BMR10] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th*

Conference on Computer and Communications Security, pages 131–140, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[BR17] Andrej Bogdanov and Alon Rosen. *Pseudorandom Functions: Three Decades Later*, pages 79–158. Springer International Publishing, Cham, 2017.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC 2017: 15th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 264–302. Springer, Heidelberg, Germany, March 2017.

[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 Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 1–30, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. Lecture Notes in Computer Science, pages 446–476. Springer, Heidelberg, Germany, 2017.

[DKW16] Apoorvaa Deshpande, Venkata Koppula, and Brent Waters. Constrained pseudorandom functions for unconstrained inputs. Lecture Notes in Computer Science, pages 124–153. Springer, Heidelberg, Germany, 2016.

[DS15] Nico Döttling and Dominique Schröder. Efficient pseudorandom functions via on-the-fly adaptation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 329–350, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[Fat06] Richard J Fateman. When is fft multiplication of arbitrary-precision polynomials practical? *University of California, Berkeley*, 2006.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[IL95] Neil Immerman and Susan Landau. The complexity of iterated multiplication. *Information and Computation*, 116(1):103–116, 1995.

[JKP18] Tibor Jager, Rafael Kurek, and Jiaxin Pan. Simple and more efficient prfs with tight security from lwe and matrix-ddh. Cryptology ePrint Archive, Report 2018/826, 2018. `https://eprint.iacr.org/2018/826`.

[KSN+04] Donald E Knuth, Hiroaki Saitou, Takahiro Nagao, Shougo Matui, Takao Matui, and Hitoshi Yamauchi. *of Book: The Art of Computer Programming.-Volume 2, Seminumerical Algorithms (Japanese Edition)*, volume 2. ASCII, 2004.

[KW17] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. Lecture Notes in Computer Science, pages 503–536, Santa Barbara, CA, USA, 2017. Springer, Heidelberg, Germany.

[LMR14] Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Improved constructions of PRFs secure against related-key attacks. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14: 12th International Conference on Applied Cryptography and Network Security*, volume 8479 of *Lecture Notes in Computer Science*, pages 44–61, Lausanne, Switzerland, June 10–13, 2014. Springer, Heidelberg, Germany.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[MG12] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2012.

[Mon18] Hart Montgomery. A nonstandard variant of learning with rounding with polynomial modulus and unbounded samples. Cryptology ePrint Archive, Report 2018/100, 2018. `https://eprint.iacr.org/2018/100`.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages

21–39, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th Annual Symposium on Foundations of Computer Science*, pages 170–181, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In *49th Annual ACM Symposium on Theory of Computing*, pages 461–473. ACM Press, 2017.

[PS17] Chris Peikert and Sina Shiehian. Privately constraining and programming prfs, the lwe way. Cryptology ePrint Archive, Report 2017/1094, 2017. `https://eprint.iacr.org/2017/1094`.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press.

[Rei] Omer Reingold. Pseudorandom synthesizers, functions, and permutations.

[RT92] John H Reif and Stephen R Tate. On threshold circuits and polynomial computation. *SIAM Journal on Computing*, 21(5):896–908, 1992.

[RW04] Steven Rudich and Avi Wigderson. *Computational complexity theory*, volume 10. American Mathematical Soc., 2004.

[SS91] Victor Shoup and Roman Smolensky. Lower bounds for polynomial evaluation and interpolation problems. In *32nd Annual Symposium on Foundations of Computer Science*, pages 378–383, San Juan, Puerto Rico, October 1–4, 1991. IEEE Computer Society Press.