Key Homomorphic PRFs and Their Applications*

Dan Boneh† Kevin Lewi† Hart Montgomery† Ananth Raghunathan†

February 2, 2014

Abstract

A pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is said to be key homomorphic if given $F(k_1, x)$ and $F(k_2, x)$ there is an efficient algorithm to compute $F(k_1 \oplus k_2, x)$, where $\oplus$ denotes a group operation on k_1 and k_2 such as `xor`. Key homomorphic PRFs are natural objects to study and have a number of interesting applications: they can simplify the process of rotating encryption keys for encrypted data stored in the cloud, they give one round distributed PRFs, and they can be the basis of a symmetric-key proxy re-encryption scheme. Until now all known constructions for key homomorphic PRFs were only proven secure in the random oracle model. We construct the first provably secure key homomorphic PRFs in the standard model. Our main construction is based on the learning with errors (LWE) problem. In the proof of security we need a variant of LWE where query points are non-uniform and we show that this variant is as hard as the standard LWE. We also construct key homomorphic PRFs based on the decision linear assumption in groups with an ℓ-linear map. We leave as an open problem the question of constructing standard model key homomorphic PRFs from more general assumptions.

Keywords. Pseudorandom functions, Key homomorphism, Non-uniform learning with errors.

1 Introduction

Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a secure Pseudorandom Function (PRF) and suppose that the key space $\mathcal{K}$ has a group structure where $\oplus$ denotes the group action. We say that F is *key homomorphic* if given $F(k_1, x)$ and $F(k_2, x)$ there is an efficient procedure that outputs $F\big(k_1 \oplus k_2, x\big)$. That is, the PRF is homomorphic with respect to its key. We show below that key homomorphic PRFs have several important applications that are practically motivated.

Constructing key homomorphic PRFs in the random oracle model is straightforward. Let $\mathbb{G}$ be a finite cyclic group of prime order q and let $H_1 : \mathcal{X} \to \mathbb{G}$ be a hash function modeled as a random oracle. Define the function $F_{\text{DDH}} : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}$ as

$$F_{\text{DDH}}(k, x) \leftarrow H_1(x)^k,$$

and observe that $F_{\text{DDH}}(k_1 + k_2, x) = F_{\text{DDH}}(k_1, x) \cdot F_{\text{DDH}}(k_2, x)$. Naor, Pinkas, and Reingold [NPR99] showed that F_{DDH} is a secure PRF in the random oracle model assuming the Decision Diffie-Hellman assumption holds in $\mathbb{G}$. This PRF is clearly key homomorphic.

*This is the full version of a paper that appeared in Crypto 2013 [BLM+13].

†Stanford University. Email: {dabo,klewi,hartm,ananthr}@cs.stanford.edu

Similarly, we can construct random oracle key homomorphic PRFs from hard lattice problems. Let $p < q$ be two primes and let $H_2 : \mathcal{X} \to \mathbb{Z}_q^n$ be a hash function modeled as a random oracle. Define the function $F_{\text{LWR}} : \mathbb{Z}_q^n \times \mathcal{X} \to \mathbb{Z}_p$ as

$$F_{\text{LWR}}(\mathbf{k}, x) \leftarrow \lfloor \langle H_2(x), \mathbf{k} \rangle \rfloor_p ,$$

where $\lfloor x \rfloor_p$ denotes rounding an element $x \in \mathbb{Z}_q$ to $\mathbb{Z}_p$ by multiplying it by (p/q) and rounding the result (as defined in Section 2), and $\langle \cdot, \cdot \rangle$ denotes inner product. The function F_{LWR} can be easily shown to be a secure PRF in the random oracle model whenever the Learning with Rounding (LWR) assumption [BPR12] holds. Because rounding is not linear (i.e. it can happen that $\lfloor a + b \rfloor_p \neq \lfloor a \rfloor_p + \lfloor b \rfloor_p$) the function F_{LWR} is not key homomorphic. However, it comes very close and is sufficiently homomorphic for our applications. In particular, F_{LWR} is "almost" key homomorphic in the sense that

$$F_{\text{LWR}}(\mathbf{k}_1 + \mathbf{k}_2, x) = F_{\text{LWR}}(\mathbf{k}_1, x) + F_{\text{LWR}}(\mathbf{k}_2, x) + e$$

where e is small; namely, $e \in \{0, 1, 2\}$.

1.1 Our Contributions

Key homomorphic PRFs in the standard model. We construct the first (almost) key homomorphic PRFs *without* using random oracles. Our main construction, given in Section 5, is a lattice-based almost key homomorphic PRF based on the Learning with Errors (LWE) assumption [Reg05]. The PRF uses two public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m \times m}$ where the entries of these matrices are sampled uniformly at random from $\{0, 1\}$. The dimension m is derived from the security parameter. The key for the PRF is a single vector $\mathbf{k} \in \mathbb{Z}_q^m$ and its domain is $\{0, 1\}^\ell$. The PRF at the point $x = x_1 \cdots x_\ell \in \{0, 1\}^\ell$ is defined as

$$F_{\text{LWE}}(\mathbf{k}, x) = \left\lfloor \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rfloor_p \in \mathbb{Z}_p^m \ . \tag{1.1}$$

This function satisfies $F_{\text{LWE}}(\mathbf{k}_1 + \mathbf{k}_2, x) = F_{\text{LWE}}(\mathbf{k}_1, x) + F_{\text{LWE}}(\mathbf{k}_2, x) + \mathbf{e}$ where the error term $\mathbf{e} \in \{0, 1, 2\}^m$. Furthermore, if we require that $p \mid q$, then $\mathbf{e}$ must lie in $\{0, 1\}^m$. Therefore this function is almost key homomorphic in the same sense as F_{LWR}, which is sufficient for our applications. We prove that F_{LWE} is a secure PRF based on the LWE assumption in the standard model.

The construction in Eq. (1.1) is closely related to an elegant non-key homomorphic PRF due to Banerjee, Peikert, and Rosen [BPR12], but is technically quite different from it. The secret key in [BPR12] is a collection of ℓ matrices while our secret key is only a single vector $\mathbf{k} \in \mathbb{Z}_q^m$. The public parameters in [BPR12] consist of one matrix while our public parameters consist of two matrices. An important step in our proof of security requires that the two public matrices $\mathbf{A}_0, \mathbf{A}_1$ used in our PRF be low-norm matrices (e.g. binary) and this poses a challenge in proving security from the standard LWE assumption.

To prove security we define a variant of LWE called the *non-uniform* LWE problem and show that it is at least as hard as the standard LWE problem. Recall that the standard LWE assumption states that for a random $\mathbf{s} \in \mathbb{Z}_q^n$, the following two oracles are indistinguishable:

$$\mathcal{O}_{\text{LWE}} : \ \left(\mathbf{v}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n, \quad \langle \mathbf{v}_i, \mathbf{s} \rangle + \chi_i \right) \qquad \text{and} \qquad \mathcal{O}_{\$} : \ \left(\mathbf{v}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n, \quad x_i \xleftarrow{\text{R}} \mathbb{Z}_q \right)$$

where χ_i is sampled from a suitable low-norm noise distribution. We show that the LWE assumption implies that these two oracles are indistinguishable even when the vectors $\mathbf{v}_i$ are sampled from certain distributions of *low norm* vectors in $\mathbb{Z}_q^n$ or even as binary vectors in $\{0,1\}^n$. However, the dimension n must be increased—in general, the lower the norm of each $\mathbf{v}_i$, the larger n needs to be. While this low norm version of the LWE assumption is precisely what we need to prove security of the PRF F_{LWE}, this assumption may be of independent interest and useful in other settings.

Key homomorphic PRFs from ℓ-linear maps. In Section 6 we present an algebraic ℓ-bit key homomorphic PRF built from ℓ-linear maps $\hat{e} : \mathbb{G}^\ell \to \mathbb{G}_\ell$, where $\mathbb{G}_\ell$ is the ℓ^{th} target group. PRF security is based on the ℓ-decision linear assumption [BBS04, Sha07b, HK07a] in $\mathbb{G}$. For a generator $g \in \mathbb{G}$, the public parameters for the PRF are $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$ where $\mathbf{A}_0, \mathbf{A}_1$ are two matrices in $\mathbb{Z}_p^{\ell \times \ell}$ generated at random (here, the notation $g^{\mathbf{A}_0}$ denotes component-wise exponentiation). The secret key for the PRF is a single vector $\mathbf{k} \in \mathbb{Z}_p^\ell$. The PRF at the point $x = x_1 \dots x_\ell \in \{0,1\}^\ell$ is defined as

$$F_{\text{DLIN}}(\mathbf{k}, x) = (g_\ell)^{\mathbf{w}} \quad \in (\mathbb{G}_\ell)^\ell \qquad \text{where} \qquad \mathbf{w} = \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_\ell} \cdot \mathbf{k} \quad \in \mathbb{Z}_p^\ell \tag{1.2}$$

where g_ℓ is a generator of $\mathbb{G}_\ell$. Evaluating the PRF at the point x given the public parameters $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$ and key $\mathbf{k}$ can be done using a graded ℓ-linear map as explained in Section 6. The PRF $F_{\text{DLIN}}(\mathbf{k}, x)$ is clearly homomorphic with respect to the secret key $\mathbf{k}$.

This PRF is related to the Naor-Reingold DDH-based PRF [NR97], but since the DDH assumption is false in groups with an ℓ-linear map ($\ell > 1$), the relation is closer to the Lewko-Waters [LW09] variant which is proven secure under the ℓ-decision linear assumption in $\mathbb{G}$. The secret key in the Lewko-Waters PRF consists of ℓ secret matrices while in construction (1.2) the secret key is only a single vector $\mathbf{k} \in \mathbb{Z}_p^\ell$ and this enables the key homomorphic property. The proof that (1.2) is a secure PRF is somewhat different from the Naor-Reingold and Lewko-Waters proofs and more closely resembles the original proof of the GGM PRF [GGM86].

Instantiating construction (1.2) requires a graded ℓ-linear map on a group $\mathbb{G}$ for which the ℓ-decision linear assumption holds. There are currently two candidate ℓ-linear map schemes: one due to Garg, Gentry, and Halevi [GGH13] and another due to Coron, Lepoint, Tibouchi [CLT13]. Garg, Gentry and Halevi [GGH13] observe that the ℓ-decision linear problem is easy for their candidate ℓ-linear map which can make Construction (1.2) insecure. The ℓ-linear map scheme of Coron, Lepoint, Tibouchi [CLT13] appears to satisfy the ℓ-decision linear assumption and gives a possible instantiation for Construction (1.2).

We note, however, that both ℓ-linear map candidates introduce noise and consequently instantiating Construction (1.2) with either candidate will result in an almost key homomorphic PRF which is no better than the LWE-based construction (1.1). We therefore view our LWE-based construction as our primary key homomorphic PRF and await for other (noiseless) ℓ-linear map candidates to instantiate our second scheme.

Constructions (1.2) and (1.1) can be computed using a circuit whose depth depends only logarithmically on ℓ. Interestingly, both constructions are *puncturable* in the sense of [SW13]. Currently, these are the only known log-depth puncturable PRFs.

1.2 Key homomorphic PRFs: Applications

Our interest in key homomorphic PRFs stems from a number of applications for such functions. We describe them briefly here and provide more details in Section 7.

Distributed PRFs. In a one-time password system [MMP+11] such as RSA SecurID, users are given a small cryptographic token containing a PRF secret key. The token displays PRF outputs that are used as one-time passwords. An authentication server verifies a given one-time password by comparing it to its own evaluation of the PRF using the same PRF secret key. Since the server knows the secret PRF keys for all users, these authentication servers are a prime target for attacks [Cov12]. In response to attacks RSA Inc. introduced *Distributed Credential Protection* where PRF keys are split among two or more key servers and all (or most) servers have to be compromised to recover the keys. Currently this design does not provide true key splitting since the PRF in use is AES for which there is no known simple key splitting mechanism (although see [KSS12]).

More generally, *distributed PRFs* support splitting the secret key among n key servers so that at least t servers are needed to evaluate the PRF. Evaluating the PRF is done without reconstructing the key at a single location. Distributed PRFs have a long history [MS95, NR97, NPR99, Nie02, DS02, Dod03], but all previous constructions either use the random oracle model, require multiple rounds of interaction or interaction among the key servers, or scale badly with the threshold t. As a simple example, we briefly mention a practical construction of Mical and Sidney [MS95] that works well for either very small or very large thresholds t. For a standard secure PRF $F : \{0,1\}^k \times \mathcal{X} \to \mathcal{Y}$ define the xor-PRF as $F_{\oplus}\big(\ (k_1, k_2, k_3),\ x\big) = F(k_1, x) \oplus F(k_2, x) \oplus F(k_3, x)$. A 2-out-of-3 key distribution for $F_{\oplus}$ is quite simple: give one server the keys $\{k_1, k_2\}$, another server $\{k_1, k_3\}$, and the third server $\{k_2, k_3\}$. Now, a client who sends x to all three servers and receives a response from any two of them (the response is the evalution of F at x under the server's keys) can easily compute $F_{\oplus}$ at x. Evaluation uses one-round of communication and no interaction among the key servers. While this construction works well for a very small or a very large threshold t, communication increases exponentially with $\min(t, n-t)$. As mentioned above, other distributed PRF constructions require multiple rounds or use the random oracle model. Additional results on distributing symmetric keys focus on distributing the key of a Pseduorandom Permutation (PRPs) [BCF00, MSNW+05, DYY06], but these again need interaction among the key servers or multiple rounds.

Key homomorphic PRFs give a clean one-round solution to distributing PRF keys for any threshold with no interaction among the key servers: a client who wants to evaluate the PRF at a point x sends a single short message to each key server and receives a single response back from each key server. No interaction between the key servers is needed. For example, for an n-out-of-n sharing, key server i stores a random key k_i and the overall PRF key is $k = k_1 \oplus \cdots \oplus k_n$, where $\oplus$ is the group action over the key space. To evaluate $F(k, x)$ the client sends x to all key servers and each server responds with $y_i = F(k_i, x)$. The client combines the results to obtain $F(k, x)$ using the key homomorphism property. To provide t-out-of-n sharing, the client first homomorphically "multiplies" the responses from the key servers by the appropriate Lagrange coefficients and then homomorphically "adds" the results using the key homomorphism property. We give the details in Section 7.1. This application still works with an *almost* key homomorphic PRF as long as the PRF range is sufficiently larger than the homomorphism error term. The output is then defined as the high order bits of the computed value $F(k, x)$ so as to eliminate the homomorphism error.

The single round distributed PRFs obtained from our key homomorphic PRFs provide a solution to an old open problem of Canetti and Goldwasser [CG99] who needed them to construct a threshold public-key encryption system secure against adaptive chosen ciphertext attack. Without one-round distributed PRFs the system of [CG99] needs a large amout of pre-shared randomness among the key servers. This can be eliminated using our distributed PRFs: the key servers simply share the key for a one-round distributed PRF and derive the per-ciphertext randomness by applying the

PRF to the ciphertext being decrypted.

Symmetric-key proxy re-encryption. Key homomorphic PRFs provide the symmetric-key analogue of public-key proxy re-encryption [BBS98, AFG+06, CH07, ABH09, LV11]. Given a message from a client encrypted under one symmetric key, a proxy can translate that ciphertext to a different symmetric key (associated with another client) without knowledge of either key. To do so, the proxy is provided with a short re-encryption token Δ that enables it to transform the symmetric encryption of the data m from key k to key k' without knowing either key.

A key homomorphic PRF directly gives a symmetric-key proxy re-encryption scheme. To see how, let $F(k, m)$ be a key homomorphic PRF satisfying $F(k \oplus k', x) = F(k, x) \otimes F(k', x)$ where both $\oplus$ and $\otimes$ are group operations. Suppose the data m is encrypted using randomized counter mode based on F—that is, the j^{th} block of m is encrypted as $c_j \leftarrow m_j \otimes F(k, N + j)$ where N is an encryption nonce. Now, to re-encrypt from key k to key k', the client sends the re-encryption token $\Delta = -k \oplus k'$ to the proxy. The proxy computes the following on every ciphertext block:

$$c'_j \leftarrow c_j \otimes F(\Delta, N + j).$$

By the key homomorphism property, $c'_j = m_j \otimes F(k, N+j) \otimes F(\Delta, N+j) = m_j \otimes F(k \oplus \Delta, N+j) = m_j \otimes F(k', N + j)$ and therefore c'_j is the encryption of m_j under key k', as required. We discuss the security of this construction in Section 7. This application works equally well with an *almost* key homomorphic PRF except that we need to pad each message block m_j with a constant number of zeros on the right to ensure that the small additive homomorphic error term $\mathbf{e}$ does not affect the encrypted plaintext after several re-encryptions.

Basic symmetric-key proxy re-encryption can also be done using a seed homomorphic pseudo-random generator (PRG) — a PRG $G : \mathcal{S} \to \mathcal{Y}$ such that $G(s_2)$ can be efficiently computed from $G(s_1)$ and $\Delta = -s_1 \oplus s_2$. We give examples of such PRGs in Section 3.2 and explain how to use them for symmetric proxy re-encryption in Section 7. However, encrypting with a key homomorphic PRF using randomized counter-mode provides a simple proxy re-encryption scheme secure against chosen-plaintext attacks, thereby enabling a single key to encrypt multiple messages.

We note that the encryption scheme can be made to provide integrity without disrupting the key homomorphism property by using "MAC then encrypt with counter-mode," which is known to provide secure authenticated encryption (see e.g., [Kra01]).

Updatable encryption. Symmetric-key proxy re-encryption built from key homomorphic PRFs elegantly solves a common problem facing companies who store encrypted data in the cloud. Let m be some data and suppose the company stores the symmetric encryption of m under key k in the cloud. Any employee who knows k has access to m. As employees leave the company there is a need to rotate the encryption key (i.e. re-encrypt m under a new key k') to ensure that ex-employees lose access to the data. Often key rotation happens at fixed time intervals (e.g. once a month). We define the security requirments for key rotation in Section 7.3. At a high level the rotated ciphertext should be distributed as a fresh encryption of the data under the new key k'. In particular, the encryption randomness should be independent of the original ciphertext. The cloud should learn nothing about the plaintext.

One naïve approach is to download the entire ciphertext from the cloud, re-encrypt under a new key, and upload the new ciphertext to the cloud. If the cloud provider is trusted to delete the old ciphertext then this ensures that employees who leave the company lose access even if they are able

to access the current data stored in the cloud. Unfortunately, downloading and re-uploading all the data from the cloud just for the purpose of key rotation results in considerable wasted bandwidth and cost.

A better solution is to encrypt the data m using a symmetric-key proxy re-encryption scheme and use the cloud as the proxy holding the company's encrypted data. Now, by simply sending to the cloud the re-encryption token $\Delta = -k \oplus k'$, the cloud can translate the ciphertext from key k to key k' in place without doing any large data transfers. As before, if the cloud is trusted to delete the old re-encryption tokens Δ and the old versions of the ciphertext, then employees who leave the company lose access to m even if they can access the current data stored with the cloud. Notice that to satisfy the security requirement for key rotation, the proxy in the proxy re-encryption scheme must output ciphertexts distributed as a fresh encryption of the data independent of the input ciphertext.

One may try to implement key rotation in the cloud using ad-hoc solutions such as nested encryption, but these solutions do not satisfy our security requirements. Moreover, they result in increased storage needs or increased decryption time or do not fully prevent a revoked employee from decrypting cloud data. A fast key homomorphic PRF provides a clean solution that does not increase storage requirements and has no impact on encryption or decryption time.

PRFs secure against related-key attacks. A related-key attack (RKA) on a PRF models a situation where an adversary is able to manipulate the secret key used in the PRF. Bellare and Cash [BC10] construct RKA-secure PRFs under the Decision Diffie-Hellman assumption and also under the decision linear assumption. One important ingredient in their constructions is a PRF that satisfies "key malleability"—informally, an adversary can transform an output of the PRF on a secret key $k \in \mathcal{K}$ and input $x \in \mathcal{X}$ into an output of the PRF on a related key $\phi(k)$ (where $\phi : \mathcal{K} \to \mathcal{K}$ is a member of a class of functions Φ) and input x *without having access to* k. For PRFs, key homomorphism implies key malleability with respect to the class $\Phi_\oplus = \{\phi : \phi(k) = k \oplus k'\}_{k' \in \mathcal{K}}$, where $\oplus$ represents the group action over $\mathcal{K}$. However, the converse does not hold in general— key malleability over $\Phi_\oplus$ is not known to imply key homomorphism.

Bellare and Cash give several constructions of RKA-secure PRFs based on various standard assumptions which are secure for certain restricted classes Φ of related-key deriving functions. We show in Section 7.4 that any key homomorphic PRF that satisfies an additional syntactic property can be used to construct an RKA-secure PRF for a larger class Φ than in [BC10].

1.3 Related work

Much recent work has focused on preventing related key attacks [BK03, BC10, BPT12]. Key homomorphic PRFs are on the other end of the spectrum where key homomorphism is encouraged in support of specific applications.

Key homomorphic PRFs give rise to one-round distributed PRFs. More generally, threshold cryptosystems [DF89] have typically been constructed for public-key primitives such an public-key encryption [SG02, CG99, DJ01, Ped91], digital signatures [DF91, FGM+97, GRJ+07, Sho00], key generation [BF97, ACS02, GJK+99], and for general functionalities [SDF+94].

Syalim, Nishide, and Sakurai [SNS11] describe a symmetric proxy re-encryption scheme based on the all or nothing transform (AONT) in the random oracle model. Cook and Keromytis [CK05] propose to use double encryption to provide one-hop symmetric proxy re-encryption.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we recollect a few preliminaries required for the rest of the paper. In Section 3 we define key homomorphic and *almost* key homomorphic PRFs. In Section 4, we introduce the problem of non-uniform learning with errors and show a reduction from the standard learning with errors problem. In Section 5 we construct an almost key homomorphic PRF under the LWE assumption. In Section 6 we construct a (fully) key homomorphic PRF under the DLIN assumption. In Section 7 we show applications of (almost) key homomorphic PRFs to constructing distributed PRFs and updatable encryption schemes. Finally, we conclude the paper in Section 8.

2 Preliminaries

Notation. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$, and by U_n the uniform distribution over the set $\{0,1\}^n$. 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 denote by $\mathbf{x}$ a vector $(x_1, \ldots, x_{|\mathbf{x}|})$. For two bit-strings x and y we denote by $x \| y$ their concatenation. For a bit string $x \in \{0,1\}^\ell$, for every $j \in [\ell]$, let $x|_j$ denote the bit string comprising the bits j through ℓ of x. A non-negative function $f : \mathbb{N} \to \mathbb{R}$ is negligible if it vanishes faster than any inverse polynomial. For a group $\mathbb{G}$ of order p, element $g \in \mathbb{G}$ and a matrix $\mathbf{M} \in \mathbb{Z}_p^{n \times m}$ (for any n and m in $\mathbb{N}$), we denote the matrix in $\mathbb{G}^{n \times m}$ whose $(i,j)^{\text{th}}$ entry is $g^{m_{i,j}}$ by $g^{\mathbf{M}}$. 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.

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 $\lfloor \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor x \rfloor_p = i$, 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 $\lfloor \mathbf{v} \rfloor_p$ 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.

PRFs and PRGs. Recall that a *pseudorandom generator* (PRG) is an efficiently computable function $G : \mathcal{S} \to \mathcal{R}$ such that for uniform s in $\mathcal{S}$ and uniform r in $\mathcal{R}$, the distribution $\{G(s)\}$ is computationally indistinguishable from the distribution $\{r\}$. A *pseudorandom function* (PRF) [GGM86] is an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for a uniform k in $\mathcal{K}$ and a uniform function $f : \mathcal{X} \to \mathcal{Y}$, an oracle for $F(k, \cdot)$ is computationally indistinguishable from an oracle for $f(\cdot)$. We let $\mathbf{Adv}^{\mathsf{PRF}}[F, \mathcal{A}]$ denote the advantage of adversary $\mathcal{A}$ in distinguishing the PRF F from a random function $f : \mathcal{X} \to \mathcal{Y}$. In this paper, we allow our PRFs and PRGs to be further parameterized by a public parameter pp. When needed, this pp is generated by a Setup algorithm. For completeness, a more detailed definition of secure PRFs is given in Appendix A.1.

2.1 Lattice Preliminaries

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 2.1 (Learning With Errors)**.** For integers $q = q(n) \geq 2$ and a noise distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $(\mathbb{Z}_q, n, \chi)$-LWE is to distinguish between the following pairs of distributions:

$$\{\mathbf{A}, \mathbf{A}^\intercal \mathbf{s} + \boldsymbol{\chi}\} \qquad \text{and} \qquad \{\mathbf{A}, \mathbf{u}\},$$

where $m = \text{poly}(n)$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{\chi} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$. We refer to the m columns of the matrix $\mathbf{A}$ as the LWE sample points.

Regev [Reg05] shows that for a certain noise distribution χ, for n polynomial in λ, and a sufficiently large q, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction (see also [Pei09, BLP+13]). These results have been extended to show that $\mathbf{s}$ 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 [ACP+09]. Similarly, the noise distribution $\boldsymbol{\chi}$ can be a simple low-norm distribution [MP13].

2.2 The Decision Linear Assumption

We review the matrix form of the decision linear assumption, which is implied by the standard decisional linear assumption (see, e.g., [BHH+08, NS12]). Recall that $\text{Rk}_i\left(\mathbb{Z}_p^{\kappa \times \kappa}\right)$ denotes the set of $\kappa \times \kappa$ matrices over $\mathbb{Z}_p$ of rank i.

Definition 2.2. The matrix form of the κ-linear (κ-DLIN) assumption states that the distributions

$$\left\{\left(\mathbb{G}, g, g^{\mathbf{X}}\right)\right\}_{\mathbf{X} \leftarrow \text{Rk}_{\kappa-1}\left(\mathbb{Z}_p^{\kappa \times \kappa}\right)} \qquad \text{and} \qquad \left\{\left(\mathbb{G}, g, g^{\mathbf{Y}}\right)\right\}_{\mathbf{Y} \leftarrow \text{Rk}_{\kappa}\left(\mathbb{Z}_p^{\kappa \times \kappa}\right)}$$

are computationally indistinguishable, where $\mathbb{G}$ is a group of prime order p, and g a generator for $\mathbb{G}$.

The 2-linear assumption is identical to the standard Decision Diffie-Hellman (DDH) problem in $\mathbb{G}$. For $\kappa = 3$ we obtain the decision linear assumption defined in [BBS04]. For larger κ we obtain the generalized linear assumption defined in [Sha07a, HK07b]. It is not difficult to show that if the κ-linear assumption holds for $\mathbb{G}$ then so does the κ'-linear assumption for $\kappa' > \kappa$. It is believed that the larger κ is the weaker the assumption becomes. In particular, the 3-linear assumption may hold in groups where the 2-linear assumption (a.k.a DDH) does not.

3 Key Homomorphic PRFs and Seed Homomorphic PRGs

In this section, we define key homomorphic PRFs and "almost" key homomorphic PRFs. We also introduce the concept of seed homomorphic PRGs and give example instantiations from standard assumptions.

3.1 Key Homomorphic Pseudorandom Functions

Definition 3.1 (Key homomorphic PRF)**.** Consider an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $(\mathcal{K}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are both groups. We say that the tuple $(F, \oplus, \otimes)$ is a key homomorphic PRF (KHPRF) if the following two properties hold:

1. F is a secure pseudorandom function.
2. For every $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$, $F(k_1, x) \otimes F(k_2, x) = F(k_1 \oplus k_2, x)$.

In Section 1 we gave an example of a key homomorphic PRF in the random oracle model due to Naor, Pinkas, and Reingold [NPR99]. While property 2 is very desirable, it is helpful to also modify the homomorphic requirement to only being approximately correct when $\mathcal{Y} = \mathbb{Z}_p^m$. We call this variant an *almost* key homomorphic PRF (AKHPRF). An AKHPRF has a parameter γ that reflects the amount of error allowed in the homomorphism.

Definition 3.2 (γ-Almost key homomorphic PRF)**.** Let $F : \mathcal{K} \times \mathcal{X} \to \mathbb{Z}_p^m$ be an efficiently computable function such that $(\mathcal{K}, \oplus)$ is a group. We say that the tuple $(F, \oplus)$ is a γ-almost key homomorphic PRF (γ-AKHPRF) if the following two properties hold:

1. F is a secure pseudorandom function.
2. For every $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$, there exists a vector $\mathbf{e} \in [0, \gamma]^m$ such that

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

For example, the function F_{LWR} from Section 1 is 1-almost key homomorphic. Such γ-almost key homomorphic functions for small γ are sufficient for the applications we have in mind.

3.2 Seed Homomorphic Pseudorandom Generators

To explain our PRF constructions it is instructive to first consider pseudorandom generators (PRGs) that are homomorphic with respect to their seed.

Definition 3.3 (Seed homomorphic PRG)**.** An efficiently computable function $G : \mathcal{X} \to \mathcal{Y}$, where $(\mathcal{X}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are groups, is is said to be seed homomorphic if the following two properties hold:

1. G is a secure PRG.
2. For every $s_1, s_2 \in \mathcal{X}$ we have that $G(s_1) \otimes G(s_2) = G(s_1 \oplus s_2)$.

3.2.1 Examples of Seed Homomorphic PRGs

We give two example seed homomorphic PRGs, one based on the Decision Diffie-Hellman (DDH) assumption and the other based on lattices.

Seed homomorphic PRGs from DDH and DLIN. Let $\mathbb{G}$ be a group of order p in which the DDH assumption holds. Consider a PRG $G_{\text{DDH}} : \mathbb{Z}_p \to \mathbb{G} \times \mathbb{G}$ with public parameter $\mathsf{pp} = (g, h)$ where g, h are chosen uniformly in $\mathbb{G}$ during setup. The generator G_{DDH} with parameter pp is defined as follows:

$$G_{\text{DDH}}(s) = (g^s, h^s) .$$

Security of this PRG follows immediately from the DDH assumption: when s is uniform in $\mathbb{Z}_p$ then $G_{\text{DDH}}(s)$ is indistinguishable from a random sample in $\mathbb{G} \times \mathbb{G}$. It should also be clear that this PRG is seed homomorphic since for all $s_1, s_2 \in \mathbb{Z}_p$,

$$G_{\text{DDH}}(s_1 + s_2) = G_{\text{DDH}}(s_1) \cdot G_{\text{DDH}}(s_2)$$

where $\cdot$ is component-wise multiplication.

More generally, for any $\ell > 0$ we can generalize G_{DDH} to obtain a seed homomorphic PRG with seed space $\mathbb{Z}_p^\ell$ which is secure under the weaker ℓ-DLIN assumption in $\mathbb{G}$. Its public parameters

pp are $g^{\mathbf{A}_0}, g^{\mathbf{A}_1} \in \mathbb{G}^{\ell\times\ell}$ where $\mathbf{A}_0$ and $\mathbf{A}_1$ are random matrices in $\mathbb{Z}_p^{\ell\times\ell}$ (here $g^{\mathbf{A}_0}$ refers to component-wise exponentiation of entries in $\mathbf{A}_0$). The generator is defined as:

$$G_{\text{DLIN}}(\mathsf{pp}, \mathbf{s}) = \left(g^{\mathbf{A}_0\mathbf{s}},\ g^{\mathbf{A}_1\mathbf{s}}\right) \in \mathbb{G}^\ell \times \mathbb{G}^\ell$$

Since $\mathbf{s}$ is given as the input to G_{DLIN} it is straightforward to compute $g^{\mathbf{A}_0\mathbf{s}}$ from $g^{\mathbf{A}_0}$ with pp, and the same holds for $g^{\mathbf{A}_1\mathbf{s}}$. Note that for $\ell = 1$ the generator G_{DLIN} is identical to G_{DDH}.

Almost seed homomorphic PRGs from LWR. Let $p < q$ and $n < m$ be parameters. Then the following PRG $G_{\text{LWR}} : \mathbb{Z}_q^n \to \mathbb{Z}_p^m$, with public parameters pp being a random matrix $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$, is secure assuming the Learning With Rounding (LWR) problem is hard for the given parameters p, q, n, m:

$$G_{\text{LWR}}(\mathbf{s}) = \lfloor \mathbf{A}^\intercal \cdot \mathbf{s} \rfloor_p \tag{3.1}$$

While this PRG is not seed homomorphic, it is close to seed homomorphic in the following sense:

$$G_{\text{LWR}}(\mathbf{s}_1 + \mathbf{s}_2) = G_{\text{LWR}}(\mathbf{s}_1) + G_{\text{LWR}}(\mathbf{s}_2) + \mathbf{e}$$

where $\mathbf{e} \in \{0, 1, 2\}^m$.

Key homomorphic PRFs from the GGM construction. Using seed homomorphic PRGs we can give some high-level intuition for the constructions in the rest of the paper. Consider a seed homomorphic PRG $G : \mathcal{X} \to \mathcal{X} \times \mathcal{X}$ where $(\mathcal{X}, \oplus)$ is a group. Since the output of $G(s)$ is in $\mathcal{X} \times \mathcal{X}$ let us write $G_0(s)$ for the left half of $G(s)$ and write $G_1(s)$ for the right half. We can now construct the GGM PRF [GGM86] with key space $\mathcal{X}$ and input space $\{0,1\}^\ell$ as follows:

$$F_{\text{GGM}}(k, x = x_1 \cdots x_\ell) = G_{x_\ell}\big(G_{x_{\ell-1}}\big(\cdots G_{x_2}(G_{x_1}(k))\cdots\big)\big) \tag{3.2}$$

The standard GGM proof shows that if G is a secure PRG then F is a secure PRF. Now suppose further that the input and output homomorphisms of G are *compatible*—that is, for all $s_1, s_2 \in \mathcal{X}$ and $b \in \{0,1\}$ we have that

$$G_b(s_1 \oplus s_2) = G_b(s_1) \oplus G_b(s_2)\ .$$

Then it is not difficult to see that F_{GGM} is key homomorphic.

Unfortunately, G_{DDH} and G_{LWR} defined in Section 3.2.1 above cannot be used directly in construction (3.2). The problem is that these generators do not compose as needed for construction (3.2).

- The output of $G_{\text{DDH}}(s)$ is in the group $\mathbb{G}$ which is not the seed space of G_{DDH}.
- The output of $G_{\text{LWR}}(\mathbf{s})$ is not a properly distributed seed for G_{LWR}.

In the next few sections we show how to overcome these difficulties while preserving the key homomorphic or almost key homomorphic property of the resulting PRFs.

It is worth noting that seed homomorphic PRGs can be used for symmetric proxy re-encryption and for updateable encryption, but the PRF-based schemes provide better security.

4 Learning with Errors with Low-Norm Samples

Before we construct our lattice-based key homomorphic PRF, we first present a variant of the learning with errors assumption needed to prove security. Recall that the basic LWE assumption [Reg05] reviewed in Section 2 states that the distribution $\{\mathbf{A}, \mathbf{A}^\intercal\mathbf{s} + \boldsymbol{\chi}\}$ is indistinguishable from the distribution $\{\mathbf{A}, \mathbf{u}\}$ where the columns of $\mathbf{A}$ are sampled uniformly in $\mathbb{Z}_q^n$ and $\mathbf{s}$ is uniform in $\mathbb{Z}_q^n$.

In this section we introduce a variant of the learning with errors (LWE) problem in which the the columns of $\mathbf{A}$ (i.e., the LWE sample points) are sampled from a *non-uniform* distribution $\boldsymbol{\eta}$ over $\mathbb{Z}_q^n$. We call this variant *Non-uniform Learning with Errors*, or NLWE for short, and show that for suitable parameters it is as hard as the basic LWE problem. In what follows we let k denote the dimension of the NLWE problem and let n denote the dimension of the LWE problem. We also write $\boldsymbol{\eta}^m$ to denote m independent samples from the distribution $\boldsymbol{\eta}$.

Definition 4.1 (Non-uniform Learning with Errors)**.** For an integer $q = q(k) \geq 2$, a noise distribution $\chi = \chi(k)$ over $\mathbb{Z}_q$, and a distribution $\boldsymbol{\eta}$ over $\mathbb{Z}_q^k$, the **non-uniform learning with errors problem** $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta})$-NLWE is to distinguish between the two distributions:

$$\{\mathbf{A}, \mathbf{A}^\intercal\mathbf{s} + \boldsymbol{\chi}\} \qquad \text{and} \qquad \{\mathbf{A}, \mathbf{u}\},$$

where $m = \text{poly}(k)$, $\mathbf{A} \leftarrow \boldsymbol{\eta}^m$ (so that $\mathbf{A} \in \mathbb{Z}_q^{k \times m}$), $\mathbf{s} \leftarrow \mathbb{Z}_q^k$, $\boldsymbol{\chi} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.

We show that for certain choices of the distribution $\boldsymbol{\eta}$ there is a reduction from $(\mathbb{Z}_q, n, \chi)$-LWE to $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta})$-NLWE for some $k \geq n$. Consequently, the NLWE problem is at least as hard as the LWE problem. In particular, we show that for suitable parameters, NLWE is as hard as LWE for the following distributions $\boldsymbol{\eta}$:

- $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$: the uniform distribution on $\{0,1\}^k$ for sufficiently large k,
- $\mathcal{D}_{\mathbb{Z}^k,\sigma}$: a discrete Gaussian on $\mathbb{Z}^k$ with a sufficiently large k and standard deviation σ,
- $\boldsymbol{\eta}_V$: a uniform distribution over a sufficiently large linear subspace V of $\mathbb{Z}_q^k$.

More generally, we show that NLWE is as hard as LWE for any distribution $\boldsymbol{\eta}$ which is *coset sampleable* as defined next.

Definition 4.2 (Coset Sampleable Distributions)**.** For integers $q = q(k)$ and $n = n(k)$ we say that a distribution $\boldsymbol{\eta} = \boldsymbol{\eta}(k)$ over $\mathbb{Z}_q^k$ is *n-coset sampleable* if there are two PPT algorithms $(\mathsf{MatrixGen}, \mathsf{SamplePre})$ such that:

- $\mathsf{MatrixGen}(1^k, n, q)$ outputs a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times k}$ and auxiliary data $\mathbf{T}$,
- $\mathsf{SamplePre}(\mathbf{z} \in \mathbb{Z}_q^n, \mathbf{T})$ outputs a $\mathbf{y} \in \mathbb{Z}_q^k$ satisfying $\mathbf{My} = \mathbf{z}$. Moreover, if $\mathbf{z}$ is distributed *uniformly* in $\mathbb{Z}_q^n$ then the output of $\mathsf{SamplePre}(\mathbf{z}, \mathbf{T})$ is distributed statistically close to $\boldsymbol{\eta}$.

The following theorem shows that $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta})$-NLWE is as hard as $(\mathbb{Z}_q, n, \chi)$-LWE for any n-coset sampleable distribution $\boldsymbol{\eta}$.

Theorem 4.3. *Let $\boldsymbol{\eta} = \boldsymbol{\eta}(k)$ be an n-*coset sampleable distribution*. Suppose there is a PPT algorithm $\mathcal{A}$ that decides the $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta})$-*NLWE *problem with advantage $\varepsilon(k)$. Then there is a PPT algorithm $\mathcal{B}$ that decides the $(\mathbb{Z}_q, n, \chi)$-*LWE *problem with the same advantage $\varepsilon(k)$.*

Proof. Algorithm $\mathcal{B}$ takes an LWE instance $(\mathbf{A}, \mathbf{v})$ as input and needs to decide whether this input is sampled from $\{\mathbf{A}, \mathbf{A}^\intercal\mathbf{s} + \boldsymbol{\chi}\}$ or from $\{\mathbf{A}, \mathbf{u}\}$ where $\mathbf{A}$ is uniform in $\mathbb{Z}_q^{n\times m}$ and $\mathbf{u}$ is uniform in $\mathbb{Z}_q^m$. Algorithm $\mathcal{B}$ translates $(\mathbf{A}, \mathbf{v})$ into an NLWE instance $(\mathbf{B}, \mathbf{v}')$ and then runs $\mathcal{A}$ on $(\mathbf{B}, \mathbf{v}')$. Algorithm $\mathcal{B}$ takes $(\mathbf{A}, \mathbf{v})$ as input and wors as follows:

1. Choose a random $\mathbf{r} \leftarrow \mathbb{Z}_q^k$ and run $\mathsf{MatrixGen}(1^k, n, q)$ to obtain a matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times k}$ and $\mathbf{T}$.
2. For each column $\mathbf{a} \in \mathbb{Z}_q^n$ of $\mathbf{A}$ run $\mathsf{SamplePre}(\mathbf{a}, \mathbf{T})$ to obtain $\mathbf{b} \in \mathbb{Z}_q^k$ such that $\mathbf{Mb} = \mathbf{a}$. Assemble all such $\mathbf{b}$ into a matrix $\mathbf{B} \in \mathbb{Z}_q^{k\times m}$. Then $\mathbf{MB} = \mathbf{A}$.
3. Set $\mathbf{v}' \leftarrow \mathbf{v} + \mathbf{B}^\intercal\mathbf{r} \in \mathbb{Z}_q^m$.
4. Run $\mathcal{A}$ on input $(\mathbf{B}, \mathbf{v}')$ and output whatever $\mathcal{A}$ outputs.

It remains to show that $(\mathbf{B}, \mathbf{v}')$ is properly distributed as a $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta})$-NLWE problem instance. First, by the definition of $\mathsf{SamplePre}$, since the columns of $\mathbf{A}$ are uniform in $\mathbb{Z}_q^n$, the columns of $\mathbf{B}$ are statistically close to $\boldsymbol{\eta}$. Second, if the input $\mathbf{v}$ is uniform in $\mathbb{Z}_q^m$ then clearly $\mathbf{v}' = \mathbf{v} + \mathbf{B}^\intercal\mathbf{r}$ is uniform in $\mathbb{Z}_q^m$. Third, if the input $\mathbf{v}$ satisfies $\mathbf{v} = \mathbf{A}^\intercal\mathbf{s} + \boldsymbol{\chi}$ then $\mathbf{v}'$ satisfies $\mathbf{v}' = \mathbf{B}^\intercal(\mathbf{M}^\intercal\mathbf{s} + \mathbf{r}) + \boldsymbol{\chi}$ because $\mathbf{A}^\intercal = \mathbf{B}^\intercal\mathbf{M}^\intercal$ and

$$\mathbf{v}' = \mathbf{v} + \mathbf{B}^\intercal\mathbf{r} = \mathbf{A}^\intercal\mathbf{s} + \boldsymbol{\chi} + \mathbf{B}^\intercal\mathbf{r} = \mathbf{B}^\intercal\mathbf{M}^\intercal\mathbf{s} + \boldsymbol{\chi} + \mathbf{B}^\intercal\mathbf{r} = \mathbf{B}^\intercal(\mathbf{M}^\intercal\mathbf{s} + \mathbf{r}) + \boldsymbol{\chi}.$$

Therefore $(\mathbf{B}, \mathbf{v}')$ is a proper NLWE instance where the secret vector is $\mathbf{s}' = \mathbf{M}^\intercal\mathbf{s} + \mathbf{r}$ which is clearly uniform in $\mathbb{Z}_q^k$. It follows that $\mathcal{B}$ decides NLWE with the same advantage as $\mathcal{A}$ decides LWE. ∎

Remark 4.4. We note that while our definition of the NLWE problem requires that the secret vector $\mathbf{s} \in \mathbb{Z}_q^k$ be uniform in $\mathbb{Z}_q^k$, the proof of Theorem 4.3 can be adapted to show that NLWE problem is hard even when $\mathbf{s}$ is non-uniform and in particular distributed as $\{\mathbf{M}^\intercal\mathbf{s}'\}$ where $\mathbf{s}'$ is distributed as the secret vector in the LWE problem (e.g., uniform in $\mathbb{Z}_q^n$). The proof is adapted to a non-uniform $\mathbf{s}$ by eliminating the randomization vector $\mathbf{r}$.

Next we show a few specific distributions $\boldsymbol{\eta}$ for which the corresponding NLWE problem is as hard as LWE.

NLWE with uniform samples in $\{0,1\}^k$. Let $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$ be the uniform distribution on $\{0,1\}^k$. We show that if the $(\mathbb{Z}_q, n, \chi)$-LWE problem is hard then so is the non-uniform LWE problem where the columns of $\mathbf{A}$ are random *binary* vectors and the dimension is increased from n to $n\lceil\log_2 q\rceil$. The proof uses bit decomposition as in [BV11] and also in [AFV11, BLP+13].

Corollary 4.5. *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 $k = n\lceil\log_2 q\rceil$. Then the $(\mathbb{Z}_q, n\lceil\log_2 q\rceil, \chi, \boldsymbol{\eta}_{\mathsf{Bin}(k)})$-NLWE problem is at least as hard as the $(\mathbb{Z}_q, n, \chi)$-LWE problem.*

Proof. By Theorem 4.3 it suffices to show that $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$ is coset sampleable. Let $\mathbf{m} \in \mathbb{Z}_q^{\lceil\log q\rceil}$ be the vector $(1, 2, 2^2, \ldots, 2^{\lceil\log q\rceil - 1})$, and let $\mathbf{M} \in \mathbb{Z}_q^{n\times k}$ be the matrix $\mathbf{M} = \mathbf{m} \otimes \mathbf{I}_n$, where $\otimes$ denotes the tensor product. Algorithms $\mathsf{MatrixGen}$ and $\mathsf{SamplePre}$ are defined as follows:

- $\mathsf{MatrixGen}(1^k, n, q)$ simply outputs $\mathbf{M} = \mathbf{m} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n\times k}$ and $\mathbf{T} = (n, q)$,

- $\mathsf{SamplePre}(\mathbf{z} \in \mathbb{Z}_q^n, \mathbf{T})$ outputs a vector $\mathbf{y}$ in $\{0,1\}^k$ by setting the entry in position $(i+j\lceil \log_2 q \rceil)$ of $\mathbf{y}$ to the i^{th} bit of the j^{th} entry of the input vector $\mathbf{z}$, for $j = 0, \ldots, n-1$ and $i = 0, \ldots, \lceil \log_2 q \rceil - 1$.

By construction $\mathbf{My} = \mathbf{z}$. Moreover, if $\mathbf{z}$ is uniformly distributed in $\mathbb{Z}_q^n$ then a standard calculation shows that the statistical distance between the distribution $\mathsf{SamplePre}(\mathbf{z}, \mathbf{T})$ and $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$ is bounded from above by $n\frac{2^{\lceil \log q \rceil} - q}{q}$, which is negligible for our choice of q and n, as required. ∎

By Remark 4.4 the NLWE problem using $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$ remains as hard as LWE when $\mathbf{s} \in \mathbb{Z}_q^k$ is distributed as $\{\mathbf{r} \otimes (1, 2, 2^2, \ldots, 2^{\lceil \log q \rceil - 1})\}$ where $\mathbf{r}$ is uniform in $\mathbb{Z}_q^n$.

NLWE with samples from a discrete Gaussian. Next, we show that when the columns of $\mathbf{A}$ in LWE are sampled from a discrete Gaussian with a sufficiently large σ then the resulting problem is as hard as LWE. Let $\mathcal{D}_{\mathbb{Z}^k,\sigma}$ denote the Gaussian distribution with standard deviation σ *restricted* to $\mathbb{Z}^k$. We need the following two facts [AP09, GPV08]:

- There is an efficient randomized algorithm $\mathsf{TrapGen}(1^n, k, q)$ that given integers $n, q \geq 2$ and $k \geq 6n \log q$, outputs a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times k}$ and a 'trapdoor' $\mathbf{T_M} \in \mathbb{Z}^{k \times k}$ such that $\mathbf{M}$ is $\mathrm{negl}(n)$-close to uniform.
- There is an efficient randomized algorithm $\mathsf{SampleD}(\mathbf{M}, \mathbf{T_M}, \mathbf{u}, \sigma)$ that given $\mathbf{u} \in \mathbb{Z}_q^n$, sufficiently large $\sigma = \Omega(\sqrt{n \log q})$, and the trapdoor $\mathbf{T_M}$ outputs a vector $\mathbf{e} \in \mathbb{Z}_q^k$ such that $\mathbf{Me} = \mathbf{u}$. Moreover, when $\mathbf{u}$ is uniform in $\mathbb{Z}_q^n$, the output of $\mathsf{SampleD}(\mathbf{M}, \mathbf{T_M}, \mathbf{u}, \sigma)$ is distributed as $\mathcal{D}_{\mathbb{Z}^k,\sigma}$.

Corollary 4.6. *Let $q = q(n)$ be an integer, $k \geq 6n \log q$, and $\sigma = \Omega(\sqrt{n \log q})$. Then the problem $(\mathbb{Z}_q, k, \chi, \mathcal{D}_{\mathbb{Z}^k,\sigma})$-NLWE is at least as hard as $(\mathbb{Z}_q, n, \chi)$-LWE.*

Proof. By Theorem 4.3 it suffices to show that $\mathcal{D}_{\mathbb{Z}^k,\sigma}$ is coset sampleable. Algorithms $\mathsf{MatrixGen}$ and $\mathsf{SamplePre}$ are defined as follows:

- $\mathsf{MatrixGen}(1^k, n, q)$ runs $\mathsf{TrapGen}(1^n, k, q)$ and outputs $\mathbf{M}$ and $\mathbf{T} = (\mathbf{M}, \mathbf{T_M})$.
- $\mathsf{SamplePre}(\mathbf{z} \in \mathbb{Z}_q^n, \mathbf{T})$ outputs $\mathsf{SampleD}(\mathbf{M}, \mathbf{T_M}, \mathbf{z}, \sigma)$.

By construction, these algorithms show that $\mathcal{D}_{\mathbb{Z}^k,\sigma}$ is coset sampleable. ∎

NLWE with samples in a linear subspace. Our last example which was also studied by Pietrzak [Pie12] shows that when the columns of $\mathbf{A}$ in LWE are sampled uniformly from a linear subspace of sufficient dimension then the resulting problem is as hard as LWE.

Corollary 4.7. *Let $q = q(n)$ be an integer and $k \geq n$. Let V be a linear subspace of $\mathbb{Z}_q^k$ of dimension d where $n \leq d \leq k$. Let $\boldsymbol{\eta}_V$ be the uniform distribution on V. Then the problem $(\mathbb{Z}_q, k, \chi, \boldsymbol{\eta}_V)$-NLWE is at least as hard as $(\mathbb{Z}_q, n, \chi)$-LWE.*

Proof. By Theorem 4.3 it suffices to show that $\boldsymbol{\eta}_V$ is coset sampleable. Algorithms $\mathsf{MatrixGen}$ and $\mathsf{SamplePre}$ are defined as follows:

- $\mathsf{MatrixGen}(1^k, n, q)$ finds an arbitrary basis of V, which we denote $\mathbf{B}_V \in \mathbb{Z}_q^{k\times d}$. It then finds a matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times k}$ such that $\mathbf{M}\mathbf{B}_V = \left(\mathbf{I}_n \| \mathbf{0}^{d-n}\right)$. It outputs $\mathbf{M}$ and the pair $\mathbf{T} = (\mathbf{M}, \mathbf{B}_V)$.
- $\mathsf{SamplePre}(\mathbf{z} \in \mathbb{Z}_q^n, \mathbf{T})$ chooses a random vector $\mathbf{x} \in \mathbb{Z}_q^{d-n}$. It then outputs the vector

$$\mathbf{y} = \mathbf{B}_V \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \end{bmatrix} \quad \in \mathbb{Z}_q^k \ .$$

By construction $\mathbf{M}\mathbf{y} = \mathbf{z}$, proving correctness. Moreover, for a uniformly distributed $\mathbf{z}$, since $\mathbf{x}$ is randomly chosen, $\mathbf{y}$ is uniformly distributed over the subspace V as required. ∎

5 An LWE-based Almost Key Homomorphic PRF

In this section we construct an (almost) key homomorphic PRF from the learning with errors problem. We showed in Section 3.2.1 that applying the GGM construction to certain seed homomophic PRGs leads to a key homomorphic PRF. Unfortunately, the lattice-based seed homomorphic PRG G_{LWR} defined in (3.1) cannot be directly used for this purpose. Nevertheless, we show how to adapt this PRG to the GGM construction so as to obtain an 1-almost key homomorphic PRF from LWE in the standard model. Despite being *almost* key homomorphic, our PRF is sufficient for all the applications discussed in the introduction.

Our new PRF has performance and security parameters comparable to those of [BPR12], but the contruction is quite different. Our key is only a single vector whereas the key in [BPR12] consists of several matrices. The simple key is part of the reason why our PRF is key homomorphic.

Construction. Let q, p, n, and m be integers such that $m = n\lceil \log q \rceil$ and p divides q. We will be using the definition of the rounding function $\lfloor \cdot \rceil_p$ from Section 2, the definition of $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ from Lemma 4.5, and the standard LWE noise distribution $\overline{\Psi}_\alpha$.[1]

Let the public parameter pp be a pair of matrices of the form $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m\times m}$ where each row of $\mathbf{A}_0$ and $\mathbf{A}_1$ is sampled from $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ such that both matrices are full rank[2]. The secret key $\mathbf{k}$ is a vector in $\mathbb{Z}_q^m$. Define $F_{\text{LWE}} : \mathbb{Z}_q^m \times \{0,1\}^\ell \to \mathbb{Z}_p^m$ as follows:

$$F_{\text{LWE}}(\mathbf{k}, x) = \left\lfloor \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rceil_p . \tag{5.1}$$

Theorem 5.1. *The function F_{LWE} is pseudorandom under the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)$-LWE assumption for parameter choices satisfying $\alpha \cdot m^\ell \cdot p \le 2^{-\omega(\log n)}$.*

Parameters. Before proving Theorem 5.1, we give example parameters for different levels of security. For a given security parameter n and $\alpha > 0$ we set $q = O\left(\sqrt{n}/\alpha\right)$ and $m = \lceil n \log q \rceil$. The choice of α determines the choice of the underlying lattice assumption and bounds the parameters ℓ and p (which in turn determine the size of the inputs and outputs of F_{LWE}, respectively).

[1] For an $\alpha \in (0,1)$ and a prime q, the random variable $\overline{\Psi}_\alpha$ over $\mathbb{Z}_q$ is defined as $\lceil qX \rfloor \pmod q$ where X is a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$. [Reg05]

[2] This can be done by sampling at most $5m$ uniform vectors from $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ (see Lemma A.3).

- Assuming LWE is hard for $\alpha = 2^{-\omega(\log n)}$ lets us set $\ell = O(1)$ and $p = \mathrm{poly}(n)$.
- A slightly stronger assumption on the hardness of LWE allows us to construct a PRF with a much larger input length. For example, setting $\alpha = 2^{-\omega(\log^{1+c}(n))}$ for some constant $c > 0$ allows $\ell = O(\log n)$ with $p = 2^{\log^{1+c}(n)}$.
- And finally, assuming LWE is hard for $\alpha = 2^{-n^\varepsilon}$ for some $0 < \varepsilon < 1$ allows us to set $\ell = n^\varepsilon / \log n$ and $p = 2^{n^\varepsilon - \omega(\log n)}$.

For a sufficiently large q, Regev [Reg05] (through a quantum reduction) and Peikert [Pei09] shows that LWE is as hard as approximating the worst-case GapSVP to $\tilde{O}(n/\alpha)$ factors, which is believed to be intractable even when $\alpha = 2^{-n^\varepsilon}$ for fixed $0 < \varepsilon < 1/2$. Hence, all three examples above are justified.

Proof Overview of Theorem 5.1. In proving the pseudorandomness of F_{LWE}, we follow the outline of the standard GGM construction [GGM86]. The proof uses $\ell + 1$ hybrid experiments, where in each experiment Expt_i for $i \in [\ell + 1]$, we successively ignore additional bits of the input in computing the product of $\mathbf{A}_{x_i}$'s while replacing this product with consistent random values. As usual, experiment Expt_1 corresponds to the case where the adversary is given an oracle for a truely random function. Experiment $\mathsf{Expt}_{\ell+1}$ honestly evaluates the PRF. Therefore, it suffices to show the indistinguishability of Expt_j and Expt_{j+1} for all $j \in [\ell + 1]$.

Consider the adversary's queries to its PRF oracle. In particular, let $x \in \{0,1\}^\ell$ be one such query and set $\mathbf{P} = \prod_{i=1}^{j-1} \mathbf{A}_{x_i}$. The PRF evaluation given to the adversary in Expt_j and Expt_{j+1} is $\lfloor \mathbf{P} \cdot \mathbf{r} \rceil_p$ and $\lfloor \mathbf{P}\mathbf{A}_{x_j} \cdot \mathbf{r} \rceil_p$ respectively. Here $\mathbf{r}$ is chosen uniformly in $\mathbb{Z}_q^m$ and is kept consistent across the adversary's queries using a lookup table indexed by the $(\ell - j)$ low order bits of the query x. An LWE challenge, either of the form $(\mathbf{A}, \mathbf{A}\mathbf{s} + \boldsymbol{\delta})$ or $(\mathbf{A}, \mathbf{r})$, cannot immediately be used to simulate the above evaluations. Instead, we move to an intermediate hybrid between Expt_j and Expt_{j+1} where the evaluations given to the adversary are $\lfloor \mathbf{P}\mathbf{A}_{x_j}\mathbf{s} + \mathbf{P}\boldsymbol{\delta} \rceil_p = \lfloor \mathbf{P}(\mathbf{A}_{x_j}\mathbf{s} + \boldsymbol{\delta}) \rceil_p$, where $\mathbf{s}$ and $\boldsymbol{\delta}$ are kept consistent across the adversary's queries as was done previously. Now, it remains to show that

$$\text{(a)} \quad \lfloor \mathbf{P}\mathbf{A}_{x_j} \cdot \mathbf{r} \rceil_p \approx \lfloor \mathbf{P}\mathbf{A}_{x_j}\mathbf{s} + \mathbf{P}\boldsymbol{\delta} \rceil_p \qquad \text{and} \qquad \text{(b)} \quad \lfloor \mathbf{P}(\mathbf{A}_{x_j}\mathbf{s} + \boldsymbol{\delta}) \rceil_p \approx \lfloor \mathbf{P} \cdot \mathbf{r} \rceil_p ,$$

Together these show that Expt_j is indistinguishable from Expt_{j+1}.

Statistical indistinguishability in (a) follows from a probabilistic argument by showing that for appropriate choices of parameters, the additive term $\mathbf{P}\boldsymbol{\delta}$ has no impact on the rounding. That is, $\lfloor \mathbf{P}\mathbf{A}_{x_j} \cdot \mathbf{s} + \mathbf{P}\boldsymbol{\delta} \rceil_p$ is equal to $\lfloor \mathbf{P}\mathbf{A}_{x_j} \cdot \mathbf{s} \rceil_p$ with high probability. For this to hold we need $\mathbf{P}$ to be a low-norm matrix so that $\mathbf{P}\boldsymbol{\delta}$ is low norm. This is why we sample $\mathbf{A}_0$ and $\mathbf{A}_1$ from $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$—it ensures that $\mathbf{P} = \prod_{i=1}^{j-1} \mathbf{A}_{x_i}$ is a low norm matrix so that $\mathbf{P}\boldsymbol{\delta}$ is a low norm vector.

The two terms in (b) are of the more familiar form of an LWE challenge, but with one important distinction. In our setup the matrix $\mathbf{A}_{x_j}$ is low-norm, which is precisely the settings of the non-uniform learning with errors problem introduced in Section 4. Using Theorem 4.3 we show computational indistinguishability in (b) under the *standard* LWE assumption. ∎

In what follows, the intermediate hybrid introduced above corresponds to the experiment $\widetilde{\mathsf{Expt}}_j$.

Proof of Theorem 5.1. We denote by B a real number such that, for $\mathbf{v} \leftarrow \overline{\Psi}_\alpha^m$, it holds that $\Pr[\|\mathbf{v}\| \geq B] \leq 2^{-\omega(\log n)}$. From [GPV08, Lemma 8.2], it suffices to consider $B = q\sqrt{m}\alpha \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2 = m \cdot \omega(\sqrt{\log m})$ (from our choice of parameters). For a bit string x on ℓ bits, $x|_j$ denotes the bit string comprising bits j through ℓ of x. For the rest of this section, we will consider a fixed adversary $\mathcal{A}$ that performs Q queries when interacting with the PRF F_{LWE}. We define a series of experiments in the following manner:

Experiment Expt_j. For every $j \in [1, \ell+1]$, we define experiment Expt_j as follows:

1. The challenger samples public parameters $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{m \times m}$ by selecting each row of both matrices from $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ such that both matrices are full rank. Then the challenger sends pp to the adversary.
2. The challenger creates a lookup table L of pairs $(w, \mathbf{z}) \in \{0,1\}^{\ell-j+1} \times \mathbb{Z}_q^m$, initialized to be empty.
3. The adversary (adaptively) sends input queries $x^{(1)}, \ldots, x^{(Q)} \in \{0,1\}^\ell$ to the challenger.
4. On input $x^{(k)}$, the challenger checks to see if there is a pair $\left(x^{(k)}|_j, \mathbf{z}\right)$ in L for some $\mathbf{z} \in \mathbb{Z}_q^m$. If there is no such pair, then the challenger chooses a random $\mathbf{y} \in \mathbb{Z}_q^m$, adds the pair $\left(x^{(k)}|_j, \mathbf{y}\right)$ to L, and sets $\mathbf{z} = \mathbf{y}$. The challenger returns $\left\lfloor \prod_{i=1}^{j-1} \mathbf{A}_{x_i^{(k)}} \cdot \mathbf{z} \right\rceil_p$ to the adversary.
5. Eventually the adversary outputs a bit $b' \in \{0,1\}$.

For $j \in [1, \ell+1]$, let W_j denote the probability that $\mathcal{A}$ outputs 1 in Expt_j. Note that Expt_1 is identical to $\mathsf{Expt}_1^{\mathsf{PRF}}$, and $\mathsf{Expt}_{\ell+1}$ is identical to $\mathsf{Expt}_0^{\mathsf{PRF}}$ in Definition A.1 for F_{LWE}. The following lemma states that $|W_{j-1} - W_j|$ is negligible under the LWE assumption for our choice of parameters. As Expt_1 is identical to $\mathsf{Expt}_1^{\mathsf{PRF}}$ and $\mathsf{Expt}_{\ell+1}$ is identical to $\mathsf{Expt}_0^{\mathsf{PRF}}$, ℓ applications of the following lemma and a standard triangle inequality over all the terms $|W_{j-1} - W_j|$ concludes the proof of Theorem 5.1. ∎

For a decision problem P and adversary $\mathcal{A}$, let $\mathbf{Adv}^P[\mathcal{A}]$ denote the advantage of $\mathcal{A}$ in deciding P.

Lemma 5.2. *Given a probabilistic polynomial time PRF adversary $\mathcal{A}$, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ (running in almost the same time as $\mathcal{A}$) such that for all $j \in [2, \ell+1]$,*

$$\left| W_{j-1} - W_j \right| \leq \frac{2^\ell (2Bm^\ell + 1)mp}{q} + Q \cdot \mathbf{Adv}^{(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)\text{-LWE}}[\mathcal{B}].$$

Proof. Fix a $j \in [2, \ell+1]$. To prove Lemma 5.2, we introduce another experiment.

Experiment $\widetilde{\mathsf{Expt}}_j$. We define an experiment $\widetilde{\mathsf{Expt}}_j$ *that differs from* Expt_j in steps (2) and (4) as follows:

2. The challenger creates a lookup table L of triples $(w, \mathbf{y}, \mathbf{z}) \in \{0,1\}^{\ell-j+1} \times \mathbb{Z}_q^m \times \mathbb{Z}_q^m$, initialized to be empty.
4. On input $x^{(k)}$, the challenger checks to see if there is a triple $(x^{(k)}|_{j-1}, \mathbf{z}, \boldsymbol{\delta})$ in L for some $\mathbf{z} \in \mathbb{Z}_q^m$ and $\boldsymbol{\delta} \leftarrow \overline{\Psi}_\alpha^m$. If there is no such pair, then the challenger chooses a random $\mathbf{y} \in \mathbb{Z}_q^m$ and random $\mathbf{v}_0, \mathbf{v}_1 \leftarrow \overline{\Psi}_\alpha^m$, adds the triples $\left(0 \| \left(x^{(k)}|_j\right), \mathbf{y}, \mathbf{v}_0\right)$ and $\left(1 \| \left(x^{(k)}|_j\right), \mathbf{y}, \mathbf{v}_1\right)$ to L, and sets $\mathbf{z} = \mathbf{y}$ and $\boldsymbol{\delta} = \mathbf{v}_{x_{j-1}^{(k)}}$ (i.e., $\mathbf{v}_0$ or $\mathbf{v}_1$ depending on the $j-1^{\text{th}}$ bit of $x^{(k)}$). The challenger returns $\left\lfloor \prod_{i=1}^{j-2} \mathbf{A}_{x_i^{(k)}} \cdot (\mathbf{A}_{x_{j-1}^{(k)}} \cdot \mathbf{z} + \boldsymbol{\delta}) \right\rceil_p$ to the adversary.

For $j \in [1, \ell+1]$, let $\widetilde{W}_j$ denote the probability that $\mathcal{A}$ outputs 1 in $\widetilde{\mathsf{Expt}}_j$. The proof of Lemma 5.2 follows from Claims 5.3 and 5.4 below.

Claim 5.3. *For all $j \in [2, \ell+1]$, it holds that*

$$\left|W_j - \widetilde{W}_j\right| \leq \frac{2^\ell (2Bm^\ell + 1)mp}{q}.$$

Proof. In Expt_j, let $\lfloor \mathbf{y} \rfloor_p$ denote the value returned when the adversary sends a query for a point $x \in \{0,1\}^\ell$. Recall that $\mathbf{y} = \prod_{i=1}^{j-1} \mathbf{A}_{x_i^{(k)}} \cdot \mathbf{z}$ for some $\mathbf{z}$. Let Bad_x denote the event that, there exists a $\boldsymbol{\delta} \in \mathbb{Z}_q^m$ such that $\|\boldsymbol{\delta}\| < B$ and for $\mathbf{w} = \prod_{i=1}^{j-2} \mathbf{A}_{x_i^{(k)}} \cdot \boldsymbol{\delta}$ we have that $\lfloor \mathbf{y} \rfloor_p \neq \lfloor \mathbf{y} + \mathbf{w} \rfloor_p$. When Bad_x does not happen, the returned value for a query at x in Expt_j is the same as in $\widetilde{\mathsf{Expt}}_j$.

Recall that the matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ are drawn from $\boldsymbol{\eta}_{\mathsf{Bin}}$. From Lemma A.2, we have that with high probability, $\|\mathbf{w}\| \leq m^\ell \cdot \|\boldsymbol{\delta}\| \leq m^\ell \cdot B$. Observe that $\lfloor \mathbf{y} + \mathbf{w} \rfloor_p \neq \lfloor \mathbf{y} \rfloor_p$ only when there is some coordinate in $\mathbf{y}$ that is within Bm^ℓ of the nearest multiple of q/p. Since $\mathbf{A}_0$ and $\mathbf{A}_1$ are full rank, the product of these matrices is also full rank. Since $\mathbf{z}$ is drawn uniformly at random from $\mathbb{Z}_q^m$, $\mathbf{y}$ is distributed uniformly in $\mathbb{Z}_q^m$ by Lemma A.4. Thus, the probability that there is a coordinate of $\mathbf{y}$ within Bm^ℓ of the nearest multiple of q/p is at most $(2Bm^\ell + 1)p/q$. A straightforward union bound over the m coordinates of $\mathbf{y}$ implies that $\Pr[\mathsf{Bad}_x] \leq (2Bm^\ell + 1)mp/q$.

If we let Bad denote the event that there exists some input $x \in \{0,1\}^\ell$ such that Bad_x occurred, a union bound implies $\Pr[\mathsf{Bad}] \leq 2^\ell \cdot \Pr[\mathsf{Bad}_x]$ which still remains negligible. Note that Expt_j is identical to $\widetilde{\mathsf{Expt}}_j$ when conditioned on the event Bad not occurring. ∎

Claim 5.4. *Given a probabilistic polynomial time algorithm $\mathcal{A}$ interacting with experiments Expt_j and $\widetilde{\mathsf{Expt}}_j$, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ (running in almost the same time as $\mathcal{A}$) such that for all $j \in [2, \ell+1]$, it holds that*

$$\left|W_{j-1} - \widetilde{W}_j\right| \leq Q \cdot \mathbf{Adv}^{(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)\text{-LWE}}[\mathcal{B}]. \tag{5.2}$$

Proof. Applying Theorem 4.3 for parameter choices $k = m$ and $\boldsymbol{\eta} = \boldsymbol{\eta}_{\mathsf{Bin}}$ we obtain that for every NLWE adversary $\mathcal{B}_1$ there exists an efficient LWE adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{(\mathbb{Z}_q, m, \boldsymbol{\eta}_{\mathsf{Bin}}, \overline{\Psi}_\alpha)\text{-NLWE}}[\mathcal{B}_1] = \mathbf{Adv}^{(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)\text{-LWE}}[\mathcal{B}].$$

Thus, it suffices to build an adversary $\mathcal{B}_1$ for NLWE satisfying the bound in (5.2).

Let Q be the maximum number of queries made by adversary $\mathcal{A}$. To prove the claim we use a Q-iterated variant of NLWE, denoted NLWE(Q), that is equivalent to the standard NLWE problem. In the problem NLWE(Q) the adversary's goal is to distinguish the distributions:

$$\{\mathbf{A},\ \mathbf{A}^\intercal \mathbf{s}_1 + \boldsymbol{\chi}_1, \ldots, \mathbf{A}^\intercal \mathbf{s}_Q + \boldsymbol{\chi}_Q\} \qquad \text{from} \qquad \{\mathbf{A},\ \mathbf{u}_1, \ldots, \mathbf{u}_Q\}$$

where $\mathbf{A} \leftarrow \boldsymbol{\eta}_{\mathsf{Bin}}^{m \times 2m}$, and for $i \in [Q]$, $\mathbf{s}_i \leftarrow \mathbb{Z}_q^m$, $\boldsymbol{\chi}_i \leftarrow \chi^{2m}$, and $\mathbf{u}_i \leftarrow \mathbb{Z}_q^{2m}$. A standard hybrid argument shows that NLWE and NLWE(Q) are equivalent where the distinguishing probabilities differ by at most a factor of Q. In particular, for every polynomial time $\mathcal{B}_2$ there is a polynomial time $\mathcal{B}_1$ such that $\mathbf{Adv}^{(\mathbb{Z}_q, m, \boldsymbol{\eta}_{\mathsf{Bin}}, \overline{\Psi}_\alpha)\text{-NLWE}(Q)}[\mathcal{B}_2]$ is at most $Q \cdot \mathbf{Adv}^{(\mathbb{Z}_q, m, \boldsymbol{\eta}_{\mathsf{Bin}}, \overline{\Psi}_\alpha)\text{-NLWE}}[\mathcal{B}_1]$.

We construct an algorithm $\mathcal{B}_2$ for the NLWE(Q) problem that uses the PRF adversary $\mathcal{A}$ and has advantage $\left|W_{j-1} - \widetilde{W}_j\right|$. Algorithm $\mathcal{B}_2$ receives Q challenges of the form $\{\mathbf{A}, \mathbf{v}_1, \ldots, \mathbf{v}_Q\}$, where $\mathbf{A} \leftarrow \boldsymbol{\eta}_{\mathsf{Bin}}^{m \times 2m}$ and $\mathbf{v}_k \in \mathbb{Z}_q^{2m}$ for all $k \in [Q]$. For each $k \in [Q]$, $\mathcal{B}_2$ creates two vectors $\mathbf{v}_{k,0}, \mathbf{v}_{k,1} \in \mathbb{Z}_q^m$ such that $\mathbf{v}_{k,0}$ consists of the first m entries of $\mathbf{v}_k$ and $\mathbf{v}_{k,1}$ consists of the last m entries of $\mathbf{v}_k$. Similarly, $\mathcal{B}_2$ sets the rows of $\mathbf{A}_0 \in \mathbb{Z}_q^{m \times m}$ to be the first m columns of $\mathbf{A}$, and the rows of $\mathbf{A}_1 \in \mathbb{Z}_q^{m \times m}$ to be the last m columns of $\mathbf{A}$. $\mathcal{B}_2$ then creates a lookup table of pairs $\mathsf{L} : \{0,1\}^{\ell - j} \times \mathbb{Z}_q^m$, initialized to be empty. $\mathcal{B}_2$ also keeps a counter $\mathsf{k} \in \mathbb{Z}$, initialized to 1. $\mathcal{B}_2$ sends $\mathsf{pp} = \{\mathbf{A}_0, \mathbf{A}_1\}$ to the adversary.

Now, when the adversary $\mathcal{A}$ makes a query $\hat{x} \in \{0,1\}^\ell$, algorithm $\mathcal{B}_2$ first checks if the pair $(\hat{x}|_{j-1}, \mathbf{z})$ exists in L, for some $\mathbf{z}$. If not, it adds the pairs $(0 \| (\hat{x}|_j), \mathbf{v}_{\mathsf{k},0})$ and $(1 \| (\hat{x}|_j), \mathbf{v}_{\mathsf{k},1})$ to the table L, and sets $\mathbf{z} = \mathbf{v}_{\mathsf{k},\hat{x}_{j-1}}$. Then, $\mathcal{B}_2$ responds to the adversary's query by returning $\left\lfloor \prod_{i=1}^{j-2} \mathbf{A}_{\hat{x}_i} \cdot \mathbf{z} \right\rceil_p$ and increments k by 1. Finally, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}_2$ outputs b' and terminates. Note that the counter k will never exceed Q since $\mathcal{A}$ makes at most Q queries, and therefore $\mathcal{B}_2$ is well-defined.

If the challenge $\{\mathbf{A}, \mathbf{v}_1, \ldots, \mathbf{v}_Q\}$ given to $\mathcal{B}_2$ is such that each $\mathbf{v}_k$ is distributed uniformly and independently in $\mathbb{Z}_q^{2m}$, then $\mathbf{z}$ is also distributed uniformly and independently for each query (consistently). Therefore $\mathcal{B}_2$ responds to queries $x \in \{0,1\}^\ell$ with $\left\lfloor \prod_{i=1}^{j-2} \mathbf{A}_{x_i} \cdot \mathbf{z} \right\rceil_p$, as in Expt_{j-1}. If the Q challenges $\{\mathbf{A}, \mathbf{v}_1, \ldots, \mathbf{v}_Q\}$ are such that each $\mathbf{v}_k$ is of the form $\mathbf{A}^\intercal \mathbf{s}_k + \boldsymbol{\chi}_k$, then $\mathbf{z}$ is of the form $\mathbf{A}_{x_{j-1}} \cdot \mathbf{s}_k + \boldsymbol{\delta}$ for each query. Therefore $\mathcal{B}_2$ responds to queries $x \in \{0,1\}^\ell$ with $\left\lfloor \prod_{i=1}^{j-2} \mathbf{A}_{x_i} \cdot (\mathbf{A}_{x_{j-1}} \cdot \mathbf{s}_k + \boldsymbol{\delta}) \right\rceil_p$, as in $\widetilde{\mathsf{Expt}}_j$. We therefore conclude that $\left|W_{j-1} - \widetilde{W}_j\right| = \mathbf{Adv}^{(\mathbb{Z}_q, m, \boldsymbol{\eta}_{\mathsf{Bin}}, \overline{\Psi}_\alpha)\text{-NLWE}(Q)}[\mathcal{B}_2]$, as required. ∎

By the triangle inequality we have $\left|W_{j-1} - W_j\right| \leq \left|W_j - \widetilde{W}_j\right| + \left|W_{j-1} - \widetilde{W}_j\right|$ and therefore applying Claims 5.3 and 5.4 concludes the proof of Lemma 5.2. ∎

Key homomorphism. To conclude this section we briefly show that F_{LWE} is indeed a 2-almost key homomorphic PRF, as required.

Theorem 5.5. *The tuple $(F_{\text{LWE}}, +)$ is a 2-almost key homomorphic PRF, where $+$ denotes addition over $\mathbb{Z}_q^m$.*

Proof. From Theorem 5.1, F_{LWE} is a pseudorandom function. We now show that for any $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{Z}_q^m$, for all inputs $x \in \{0,1\}^\ell$, there is some vector $\mathbf{e} \in \{0,1,2\}^m$ such that

$$F(\mathbf{k}_1, x) + F(\mathbf{k}_2, x) = F(\mathbf{k}_1 + \mathbf{k}_2, x) + \mathbf{e}.$$

Let $\mathbf{v} = \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k}_1$ and $\mathbf{w} = \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k}_2$, and let vectors $\mathbf{v}', \mathbf{w}' \in [0, \lfloor q/p \rfloor - 1]^m$ be such that $\mathbf{v} = \lfloor q/p \rfloor \lfloor \mathbf{v} \rceil_p + \mathbf{v}'$ and $\mathbf{w} = \lfloor q/p \rfloor \lfloor \mathbf{w} \rceil_p + \mathbf{w}'$. It follows that

$$\lfloor \mathbf{v} + \mathbf{w} \rceil_p = \left\lfloor \lfloor q/p \rfloor \lfloor \mathbf{v} \rceil_p + \lfloor q/p \rfloor \lfloor \mathbf{w} \rceil_p + \mathbf{v}' + \mathbf{w}' \right\rceil_p.$$

Note that $\mathbf{v}' + \mathbf{w}' \in [0, 2\lfloor q/p \rfloor - 2]$, and so the last quantity is equal to $\lfloor \mathbf{v} \rceil_p + \lfloor \mathbf{w} \rceil_p + \mathbf{e}$ for some $\mathbf{e} \in \{0,1,2\}^m$, which concludes the proof. ∎

we note that if p divides q, then $(F_{\text{LWE}}, +)$ is in fact a 1-almost key homomorphic PRF.

6 A DLIN-based Key Homomorphic PRF

In this section we present a key homomorphic PRF based on the DLIN assumption. The construction presented is *fully* key homomorphic (as opposed to the LWE-based construction in Section 5 which was almost key homomorphic), but requires a group equipped with a *multilinear map* [BS03].

A construction of Garg, Gentry, and Halevi [GGH13] uses ideal lattices to approximate multilinear maps. Unfortunately, they observe that the DLIN assumption does not hold for their general construction. The ℓ-linear map scheme of Coron, Lepoint, Tibouchi [CLT13] appears to satisfy the ℓ-decision linear assumption and may give a possible instantiation for Construction (1.2).

Groups with multilinear maps. We let MMGen be a probabilistic polynomial-time algorithm that takes as input a security parameter 1^λ and a positive integer ℓ. $\mathsf{MMGen}(1^\lambda, \ell)$ outputs a sequence of groups $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_\ell)$ each of prime order p (for a λ-bit prime p). Let g_i denote a canonical generator of $\mathbb{G}_i$ and $g = g_1$. We assume the existence of a sequence of "graded" pairings $\hat{e}_i : \mathbb{G}_1 \times \mathbb{G}_i \to \mathbb{G}_{i+1}$ for all $i \leq \ell - 1$ such that $\hat{e}_i\left(g^a, g_i^b\right) = g_{i+1}^{ab}$ for all $a, b \in \mathbb{Z}_p$. We consider a composed pairing operation, the multilinear map $\hat{e} : \mathbb{G}_1^\ell \to \mathbb{G}_\ell$, defined as

$$\hat{e}\left(g^{a_1}, \ldots, g^{a_\ell}\right) = \hat{e}_{\ell-1}\left(g^{a_1}, \hat{e}_{\ell-2}\left(g^{a_2}, \cdots, \hat{e}_1\left(g^{a_{\ell-1}}, g^{a_\ell}\right)\right)\right),$$

which satisfies the following property: for all $a_1, \ldots, a_\ell \in \mathbb{Z}_p$, $\hat{e}(g^{a_1}, \ldots, g^{a_\ell}) = g_\ell^{a_1 a_2 \cdots a_\ell}$.

The DLIN assumption in multilinear groups. We define the equivalent of the decision linear assumption from Section 2 in the presence of an ℓ-linear map. Consider a sequence of groups $\vec{\mathbb{G}}$ with a set of bilinear maps $\hat{e}_i$ output by $\mathsf{MMGen}(1^\lambda, \ell)$.

Definition 6.1. The matrix form of the κ-linear (κ-DLIN) assumption in the presence of an ℓ-linear map states that for all $\ell \leq j < \kappa$ the distributions

$$\left\{\left(\vec{\mathbb{G}}, p, g, \{\hat{e}_i\}_{i\in[\ell-1]}, g^{\mathbf{X}}\right)\right\}_{\mathbf{X}\leftarrow \mathrm{Rk}_j\left(\mathbb{Z}_p^{\kappa\times\kappa}\right)} \quad \text{and} \quad \left\{\left(\vec{\mathbb{G}}, p, g, \{\hat{e}_i\}_{i\in[\ell-1]}, g^{\mathbf{Y}}\right)\right\}_{\mathbf{Y}\leftarrow \mathrm{Rk}_\kappa\left(\mathbb{Z}_p^{\kappa\times\kappa}\right)}$$

are computationally indistinguishable, where $\left(\vec{\mathbb{G}}, p, g, \{\hat{e}_i\}_{i\in[\ell-1]}\right) \leftarrow \mathsf{MMGen}(1^\lambda, \ell)$.

The distinguishing problem for a specific $j \in [\ell, \kappa - 1]$ is no harder (up to polynomial factors) than the distinguishing problem for any other $j \in [\ell, \kappa - 1]$ (see e.g., [BHH+08, NS12]) and therefore for all $j \in [\ell, \kappa - 1]$ we obtain a class of equivalent assumptions. Consequently, we simply refer to the κ-linear assumption. In our construction, we use the DLIN assumption with parameters $\kappa = 2\ell$ and $j = \ell$ in the presence of an ℓ-linear map.

We note that the κ-DLIN assumption is only plausible in the presence of an ℓ-linear map if $\kappa > \ell$. To see this, note that an ℓ-linear map allows for efficient evaluation of degree ℓ polynomials of x_i in the exponent, given only terms of the form g^{x_i}. In particular, this allows for efficient computation of determinants of all matrices of dimension at most ℓ, thereby distinguishing the two distributions of the DLIN challenge.

PRF Construction. Our key homomorphic PRF is parameterized by ℓ, the input length of the PRF. Let $\left(\vec{\mathbb{G}}, p, g, \{\hat{e}_i\}_{i\in[\ell-1]}, \hat{e}\right) \leftarrow \mathsf{MMGen}(1^\lambda, \ell)$ be a sequence of ℓ groups equipped with a multilinear map. The key homomorphic PRF we construct, denoted F_{DLIN}, consists of public

parameters $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$, where $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathrm{Rk}_\ell\left(\mathbb{Z}_p^{\ell\times\ell}\right)$. The secret key $\mathbf{k}$ is a vector in $\mathbb{Z}_p^\ell$. Define $F_{\text{DLIN}}\colon \mathbb{Z}_p^\ell \times \{0,1\}^\ell \to (\mathbb{G}_\ell)^\ell$ as follows:

$$F_{\text{DLIN}}(\mathbf{k}, x) = (g_\ell)^P \quad \text{where} \quad P = \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k} \quad \in \mathbb{Z}_p^\ell \ . \tag{6.1}$$

The PRF can be evaluated at a point $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$ given the the public parameters pp and secret key $\mathbf{k} \in \mathbb{Z}_p^\ell$ using the graded bilinear maps $\hat{e}_i : \mathbb{G}_1 \times \mathbb{G}_i \to \mathbb{G}_{i+1}$. We simply carry out the matrix multiplication one step at a time by nesting these bilinear maps as follows:

$$F_{\text{DLIN}}(\mathbf{k}, x) = \hat{e}_{\ell-1}\left(g^{\mathbf{A}_{x_1}},\ \hat{e}_{\ell-2}\left(g^{\mathbf{A}_{x_2}},\ \ldots \hat{e}_2\left(g^{\mathbf{A}_{x_{\ell-2}}},\ \hat{e}_1\left(g^{\mathbf{A}_{x_{\ell-1}}},\ \left(g^{\mathbf{A}_{x_\ell}}\right)^{\mathbf{k}}\right)\right)\right)\right)$$

Here a pairing $\hat{e}(g^{\mathbf{A}_0}, g^{\mathbf{A}_1})$ of matrices given in the exponent is done by computing the component-wise dot products of rows of $\mathbf{A}_0$ with columns of $\mathbf{A}_1$ using the bilinear map $\hat{e}$.

Key Homomorphism. The pseudorandom function F_{DLIN} satisfies the following homomorphism property: for all $x \in \{0,1\}^\ell$, for all $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{Z}_p^\ell$, it holds that $F(\mathbf{k}_1, x) \cdot F(\mathbf{k}_2, x) = F(\mathbf{k}_1 + \mathbf{k}_2, x)$, where $+$ is addition in $\mathbb{Z}_p^\ell$, and $\cdot$ denotes component-wise multiplication in $(\mathbb{G}_\ell)^\ell$.

Security. The function F_{DLIN} can be seen as a modified instantiation of the GGM construction using the seed homomorphic PRG G_{DLIN} (from Section 3). The modification uses a sequence of pseudorandom generators

$$G^{(i)}\left(\mathsf{pp}, g_i^{\mathbf{s}}\right) = \left(\ \hat{e}_i\left(g^{\mathbf{A}_0}, g_i^{\mathbf{s}}\right)\ ,\ \hat{e}_i\left(g^{\mathbf{A}_1}, g_i^{\mathbf{s}}\right)\ \right) \in \mathbb{G}_{i+1}^2$$

for all $i \in [\ell - 1]$. Each generator $G^{(i)}$ takes as input a public parameter $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$ and a seed $g_i^{\mathbf{s}}$ and outputs a pair of seeds for the next level of the sequence. When $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathrm{Rk}_\ell\left(\mathbb{Z}_p^{\ell\times\ell}\right)$ and $\mathbf{s} \leftarrow \mathbb{Z}_p^\ell$ these pseudorandom generators are secure under the 2ℓ-DLIN assumption (with $j = \ell$). The GGM PRF constructed from this sequence of pseudorandom generators is essentially construction (6.1). A slight modification of the GGM security proof implies the pseudorandomness of F_{DLIN}. In what follows, we leverage the random self-reduction of DLIN to prove security of F_{DLIN} with a tighter reduction than GGM. This avoids the factor-Q loss in advantage in GGM, where Q denotes the number of queries made by the adversary.

Theorem 6.2. *Let $\left(\vec{\mathbb{G}}, p, g, \{\hat{e}_i\}_{i\in[\ell-1]}, \hat{e}\right) \leftarrow \mathsf{MMGen}(1^\lambda, \ell)$ be a sequence of groups with an ℓ-linear map. Then, the function F_{DLIN} is a pseudorandom function under the 2ℓ-DLIN assumption.*

Proof. For a bit string x on ℓ bits, let $x|_j$ denote the integer whose bit string consists of bits j through ℓ of x. Let $x|_{\ell+1}$ denote the empty string ε^*.

Let $\mathcal{A}$ be an efficient adversary that distinguishes function F_{DLIN} from a random function making at most Q queries. We consider the following series of experiments with the adversary $\mathcal{A}$. For every $j \in [1, \ell + 1]$, define Expt_j as follows:

1. The challenger samples public parameters $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathrm{Rk}_\ell(\mathbb{Z}_p^{\ell\times\ell})$ and publishes $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$ to the adversary.
2. The challenger creates a lookup table L of pairs $(w, \mathbf{z}) \in \{0,1\}^{\ell-j+1} \times \mathbb{Z}_p^\ell$, and initializes L to contain only the pair $(\varepsilon^*, \mathbf{r})$ for some random $\mathbf{r} \in \mathbb{Z}_p^\ell$.

3. The adversary (adaptively) sends input queries $x^{(1)}, \ldots, x^{(Q)} \in \{0,1\}^\ell$ to the challenger.
4. On input $x^{(k)}$, the challenger checks to see if there is a pair $\left(x^{(k)}|_j, \mathbf{z}\right)$ in L for some $\mathbf{z} \in \mathbb{Z}_p^\ell$. If there is no such pair, then the challenger chooses a random $\mathbf{y} \in \mathbb{Z}_p^m$, adds the pair $\left(x^{(k)}|_j, \mathbf{y}\right)$ to L, and sets $\mathbf{z} = \mathbf{y}$. The challenger computes $P = \prod_{i=1}^{j-1} \mathbf{A}_{x_i^{(k)}} \cdot \mathbf{z}$ and returns $g_\ell^P \in \mathbb{G}_\ell$ to the adversary.
5. Eventually the adversary outputs a bit $b' \in \{0,1\}$.

For $j \in [1, \ell+1]$, let W_j denote the probability that $\mathcal{A}$ outputs 1 in Expt_j. Note that Expt_1 is identical to $\mathsf{Expt}_1^{\mathsf{PRF}}$, and $\mathsf{Expt}_{\ell+1}$ is identical to $\mathsf{Expt}_0^{\mathsf{PRF}}$ in Definition A.1 for F.

Lemma 6.3. *Under the 2ℓ-DLIN assumption, for all $j \in [1,\ell]$, $|W_{j+1} - W_j|$ is negligible.*

Proof. Given an adversary $\mathcal{A}$, we construct an efficient algorithm $\mathcal{B}$ that breaks the 2ℓ-DLIN assumption with advantage $|W_{j+1} - W_j|$. The algorithm $\mathcal{B}$ is given a 2ℓ-DLIN challenge that comprises $\left(\vec{\mathbb{G}}, g, p, \{\hat{e}_i\}_{i\in[\ell-1]}, g^{\mathbf{Z}}\right)$ for some $\mathbf{Z} \in \mathbb{Z}_p^{2\ell\times 2\ell}$ and must decide whether $\mathbf{Z}$ is of rank 2ℓ or rank ℓ (we consider the 2ℓ-DLIN assumption with $j = \ell$ in Definition 6.1).

The algorithm $\mathcal{B}$ first *random self-reduces* the DLIN challenge as follows. It first parses $g^{\mathbf{Z}}$ into four $\ell \times \ell$ matrices.

$$g^{\mathbf{Z}} = \begin{pmatrix} g^{\mathbf{U}_0} & g^{\mathbf{D}_0} \\ g^{\mathbf{U}_1} & g^{\mathbf{D}_1} \end{pmatrix}. \tag{6.2}$$

For each $i \in [Q]$, and each bit $b \in \{0,1\}$, the algorithm samples $\mathbf{t}_i \leftarrow \mathbb{Z}_p^\ell$ and sets

$$g^{\mathbf{r}_{i,b}} = g^{\mathbf{D}_b \cdot \mathbf{t}_i} \in \mathbb{Z}_p^\ell .$$

Now, we state the following two claims about the vectors $(g^{\mathbf{r}_{i,b}})_{i\in[Q],b\in\{0,1\}}$ that consider the cases when $\mathrm{Rk}(\mathbf{Z}) = \ell$ and $\mathrm{Rk}(\mathbf{Z}) = 2\ell$ respectively.

Claim 6.4. *If $\mathrm{Rk}(\mathbf{Z}) = \ell$, then the vectors $(\mathbf{r}_{i,b})_{i\in[Q],b\in\{0,1\}}$ are distributed as $\mathbf{r}_{i,b} = \mathbf{U}_b \cdot \mathbf{v}_i$ where for $i \in [Q]$ the vectors $\mathbf{v}_i \in \mathbb{Z}_p^\ell$ are uniformly and independently distributed in $\mathbb{Z}_p^\ell$ (even given $\mathbf{U}_0$ and $\mathbf{U}_1$).*

Proof. Since $\mathbf{Z} \in \mathbb{Z}_p^{2\ell\times 2\ell}$ is a random rank ℓ matrix, there exists a matrix $\mathbf{W} \in \mathbb{Z}_p^{\ell\times\ell}$ such that

$$\begin{pmatrix} \mathbf{U}_0 \\ \mathbf{U}_1 \end{pmatrix} \cdot \mathbf{W} = \begin{pmatrix} \mathbf{D}_0 \\ \mathbf{D}_1 \end{pmatrix} .$$

With high probability $\mathbf{W}$ is full rank. Setting $\mathbf{v}_i = \mathbf{W} \cdot \mathbf{t}_i$ implies that $\mathbf{r}_{i,b} = \mathbf{U}_b \cdot \mathbf{v}_i$, as required. Moreover, since $\mathbf{W}$ is full-rank it maps uniformly distributed vectors $\mathbf{t}_i$ over $\mathbb{Z}_p^\ell$ to uniformly distributed vectors $\mathbf{v}_i$ over $\mathbb{Z}_p^\ell$ (since p is prime). ∎

Claim 6.5. *If $\mathrm{Rk}(\mathbf{Z}) = 2\ell$, then the vectors $(\mathbf{r}_{i,b})_{i\in[Q],b\in\{0,1\}}$ are uniformly and independently distributed in $\mathbb{Z}_p^\ell$ (even given $\mathbf{U}_0$ and $\mathbf{U}_1$).*

Proof. If $\mathrm{Rk}(\mathbf{Z}) = 2\ell$, then the sub-matrix $[\mathbf{D}_0 \,|\mathbf{D}_1]$ is *independent* of $[\mathbf{U}_0 \,|\mathbf{U}_1]$. Additionally, with overwhelming probability $\mathbf{D}_0$ and $\mathbf{D}_1$ are full-rank matrices that map (independent of $\mathbf{U}_0$ and $\mathbf{U}_1$) uniformly distributed vectors $\mathbf{t}_i$ over $\mathbb{Z}_p^\ell$ to uniformly distributed vectors $\mathbf{r}_{i,0}$ and $\mathbf{r}_{i,1}$ respectively. ∎

Having self-reduced the DLIN challenge, algorithm $\mathcal{B}$ proceeds as follows. The algorithm embeds the DLIN challenge by setting $g^{\mathbf{A}_b} = g^{\mathbf{U}_b}$ for $b \in \{0,1\}$. Algorithm $\mathcal{B}$ then creates a lookup table of pairs $\mathsf{L} : \{0,1\}^{\ell-j+1} \times \mathbb{G}^m$, initializing the table to contain the pair $(\varepsilon^*, g^{\mathbf{r}^*})$ for a randomly chosen $\mathbf{r}^* \leftarrow \mathbb{Z}_p^m$. $\mathcal{B}$ also keeps a counter $\mathsf{k} \in \mathbb{Z}$, initialized to 1. $\mathcal{B}$ sends $\mathsf{pp} = \left(g^{\mathbf{A}_0}, g^{\mathbf{A}_1}\right)$ to the adversary. To answer queries, $\mathcal{B}$ uses the random self-reduction to construct $g^{\mathbf{r}_{i,b}}$ for $i \in [Q]$ and $b \in \{0,1\}$.

On query x, algorithm $\mathcal{B}$ first checks if the pair $(x|_j, \mathbf{z})$ exists in L, for some $\mathbf{z}$. If not, the algorithm adds the pairs $\left(0\|\left(x|_{j+1}\right), g^{\mathbf{r}_{\mathsf{k},0}}\right)$ and $\left(1\|\left(x|_{j+1}\right), g^{\mathbf{r}_{\mathsf{k},1}}\right)$ to the table L, sets $g^{\mathbf{z}} = g^{\mathbf{r}_{\mathsf{k},x_j}}$, and increments k by 1. Then, $\mathcal{B}$ responds to the adversary's query with g_ℓ^P where $P = \prod_{i=1}^{j-1} \mathbf{A}_{x_i} \cdot \mathbf{z}$. (We note that this can be computed only given matrices $g^{\mathbf{A}_0}$, $g^{\mathbf{A}_1}$, and $g^{\mathbf{r}_{\mathsf{k},x_j}}$ using the multilinear map $\hat{e}$.) Finally, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ also outputs b'. Note that the counter k will never exceed Q, since $\mathcal{A}$ makes at most Q queries, and therefore $\mathcal{B}$ successfully simulates the adversary. Consider the following two cases:

- **Case 1:** If $\mathrm{Rk}(\mathbf{Z}) = \ell$, from Claim 6.4, even given the public parameters $g^{\mathbf{A}_0}$ and $g^{\mathbf{A}_1}$, for $i \in [Q]$ and $b \in \{0,1\}$, each $\mathbf{r}_{i,b}$ is of the form $\mathbf{A}_b \cdot \mathbf{v}_i$. Therefore, in each query, P is of the form $\left(\prod_{i=1}^{j-1} \mathbf{A}_{x_i}\right) \mathbf{A}_{x_j} \cdot \mathbf{v}$ for uniform and independently distributed vectors $\mathbf{v} \in \mathbb{Z}_p^\ell$. From this, it follows that the adversary's view is identical to that of Expt_{j+1}.
- **Case 2:** If $\mathrm{Rk}(\mathbf{Z}) = 2\ell$, from Claim 6.5, even given the public parameters $g^{\mathbf{A}_0}$ and $g^{\mathbf{A}_1}$, for $i \in [Q]$ and $b \in \{0,1\}$, each $\mathbf{r}_{i,b}$ is independently and uniformly distributed over $\mathbb{Z}_p^\ell$. Therefore, in each query, P is of the form $\prod_{i=1}^{j-1} \mathbf{A}_{x_i} \cdot \mathbf{r}$ for uniform and independently distributed $\mathbf{r} \in \mathbb{Z}_p^\ell$. From this, it follows that the adversary's view is identical to that of Expt_j.

The proof follows in a straightforward manner—adversary $\mathcal{B}$ perfectly simulates either Expt_{j+1} or Expt_j and under the DLIN assumption, $|W_{j+1} - W_j|$ is negligible. ∎

As Expt_1 is identical to $\mathsf{Expt}_1^{\mathsf{PRF}}$ and $\mathsf{Expt}_{\ell+1}$ is identical to $\mathsf{Expt}_0^{\mathsf{PRF}}$, ℓ applications of Lemma 6.3 and a standard triangle inequality over all the terms $|W_{j+1} - W_j|$ concludes the proof of the theorem. ∎

7 Applications of (Almost) Key Homomorphic PRFs

In this section, we construct one-round distributed PRFs, symmetric proxy re-encryption schemes, and updatable encryption from γ-almost key homomorphic PRFs, and PRFs secure against related-key attacks from (fully) key homomorphic PRFs.

7.1 Distributed PRFs

We show that key-homomorphic PRFs give rise to efficient one-round distributed PRFs.

Definition. To define distributed PRFs we follow the exposition of Naor, Pinkas, and Reingold [NPR99]. The model comprises of N servers $S_1, \ldots, S_N$ and a client C that is connected to at least t servers.

A distributed PRF is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Share}, \mathsf{F}, \mathsf{G}, \mathsf{f})$ with the following properties. Algorithm Setup takes the security parameter λ and outputs public parameters pp. The *key-sharing* algorithm $\mathsf{Share} : \mathcal{K} \to \mathcal{K}^N$ takes as input a random master secret key $k_0 \in \mathcal{K}$ and outputs a tuple

$(k_1, \ldots, k_N) \in \mathcal{K}^N$, where $k_1, \ldots, k_N$ represent the key-shares of k_0 from a (t, N)-threshold secret sharing scheme. The *partial evaluation* function $\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ computes a partial evaluation of the function f when given a key-share and an input point. The *combine* algorithm $G\colon 2^{[N]} \times \mathcal{Y}^t \to \mathcal{Y}$ takes as input a subset $W \subset [N]$ of size t and the t partial evaluations on key-shares in the set W and outputs a value in $\mathcal{Y}$. The *evaluation* function $\mathsf{f}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ maps a key and an input to the space of outputs.

The distributed PRF is initialized by a trusted third party that runs $\mathsf{Setup}(1^\lambda)$ to obtain the public parameters pp, samples a master secret key k_0 uniformly from $\mathcal{K}$, and runs $\mathsf{Share}(k_0)$ to obtain a tuple $(k_1, \ldots, k_n)$. The key-share k_i is distributed as the secret key for each server S_i along with public parameters pp[3]. A client C that wants to compute the evaluation function f under key k_0 on input x sends x to t servers $S_{i_1}, \ldots, S_{i_t}$. Each server S_i responds to the client with $\mathsf{F}(k_i, x)$. Then, the client *locally* computes $\mathsf{f}(k_0, x)$ by computing $\mathsf{G}\left(W, \mathsf{F}(k_{i_1}, x), \ldots, \mathsf{F}(k_{i_t}, x)\right)$.

Consistency. Let pp be the output of $\mathsf{Setup}(1^\lambda)$, k_0 be sampled uniformly from $\mathcal{K}$, and $(k_1, \ldots, k_N)$ be the key-shares output by $\mathsf{Share}(k_0)$. For every subset $W = \{i_1, \ldots, i_t\} \subset [N]$ of size t, and for every input x, a distributed PRF Π is *consistent* if $\mathsf{f}(k_0, x) = \mathsf{G}\left(W, \mathsf{F}(k_{i_1}, x), \ldots, \mathsf{F}(k_{i_t}, x)\right)$.

Pseudorandomness. Intuitively, the evaluation function f should remain pseudorandom even when the adversary is given $t-1$ key shares $k_{i_1}, \ldots, k_{i_{t-1}}$ for indices $\{i_1, \ldots, i_{t-1}\}$ of its choice. The adversary is also given an oracle $\mathcal{O}$ that performs arbitrary partial evaluations: it takes (i, x) as input and returns $\mathsf{F}(k_i, x)$. The adversary should be unable to distinguish the function from random at points x where it did not query the oracle $\mathcal{O}$. We formalize this intuition through an experiment between a challenger and an adversary $\mathcal{A}$. For $b \in \{0, 1\}$ the challenger in $\mathsf{Expt}_b^{\mathsf{dPRF}}$ operates as follows.

1. Given security parameter λ, the challenger runs $\mathsf{Setup}(1^\lambda)$ and publishes public parameters pp to the adversary. The challenger then samples a k_0 uniformly from $\mathcal{K}$ and runs $\mathsf{Share}(k_0)$ to obtain the tuple $(k_1, \ldots, k_N)$.

2. The adversary specifies a set $S^* = \{i_1, \ldots, i_{t-1}\}$, and the challenger responds with the corresponding secret keys $k_{i_1}, \ldots, k_{i_{t-1}}$.

3. The adversary (adaptively) sends secret-share queries $x_1, \ldots, x_Q \in \mathcal{X}$ to the challenger, and for each query x_j the challenger responds with $\mathsf{F}(k_i, x_j)$ for each $i \notin S^*$.

4. The adversary submits a challenge query $x^* \in \mathcal{X} \setminus \{x_1, \ldots, x_Q\}$ to the challenger. If $b = 0$, the challenger samples and returns a uniformly random $y \in \mathcal{Y}$ to the adversary. If $b = 1$, the challenger responds with $\mathsf{f}(k_0, x^*)$.

5. The adversary can adaptively issue more secret-share queries (step 3) and challenge queries (step 4), to which the challenger responds appropriately, so long as the set of all challenge queries and the set of all secret-share queries are disjoint.

6. The adversary outputs a bit $b' \in \{0, 1\}$.

Let W_b denote the probability that $\mathcal{A}$ outputs 1 in experiment $\mathsf{Expt}_b^{\mathsf{dPRF}}$.

[3]To avoid a trusted third party during setup time, Setup may be executed in a distributed manner such that no coalition $W \subset [N]$ of $t-1$ servers can learn any information about k_i for $i \notin W$.

Definition 7.1 (Distributed Pseudorandom Function)**.** A distributed PRF Π is secure if for all efficient adversaries $\mathcal{A}$ the quantity

$$\mathbf{Adv}^{\mathsf{dPRF}}[\Pi, \mathcal{A}] := |W_0 - W_1|$$

is negligible.

7.1.1 Distributed PRFs from key homomorphic PRFs

Let $F : \mathbb{F}_1 \times \mathcal{X} \to \mathbb{F}_2$ be a key homomorphic PRF where $\mathbb{F}_1, \mathbb{F}_2$ are fields. We use it to construct a one-round distributed PRF.

We consider a (t, N)-threshold secret sharing scheme [Sha79] over a secret k_0 in $\mathbb{F}_1$, which constructs key-shares by sampling a uniformly random polynomial $p(z) \in \mathbb{F}_1[\mathbb{Z}]$ of degree $t-1$ such that $p(0) = k_0$, and the remaining coefficients are sampled uniformly at random from $\mathbb{F}_1$. We then define share $k_i = p(i)$ for $i \in [1, N]$. For secret shares constructed in this manner, it holds that for any $W = \{i_1, \ldots, i_t\} \subset [N]$ of size t we have that $k_0 = p(0) = \sum_{i \in W} \Lambda_{i,W} \cdot k_i$, where $\Lambda_{i,W} \in \mathbb{F}_1$ are the Lagrange coefficients that depend only on W. Let $\Lambda'_{i,W} \in \mathbb{F}_2$ be such that for all $k \in \mathbb{F}_1$ and $x \in \mathcal{X}$, $\Lambda'_{i,W} \cdot F(k, x) = F(\Lambda_{i,W} \cdot k, x)$. Such $\Lambda'_{i,W}$ are ensured to be easily computable from $\Lambda_{i,W}$ by the key homomorphism property of F. We construct a distributed PRF scheme $\Pi_{\mathsf{dPRF}} = (\mathsf{Setup}, \mathsf{Share}, \mathsf{F}, \mathsf{G}, \mathsf{f})$ as follows.

- $\mathsf{Setup}(1^\lambda)$ outputs public parameters pp used by the key homomorphic PRF F.
- $\mathsf{Share}(k_0)$ samples a uniformly random polynomial $p(z)$ over $\mathbb{F}_1$ of degree $t-1$ such that $p(0) = k_0$ and outputs $(p(1), \ldots, p(N))$ in $\mathbb{F}_1^N$.
- $\mathsf{F}(k_i, x)$ returns the output of the key homomorphic PRF $F(k_i, x)$.
- $\mathsf{G}(W, y_1, \ldots, y_t)$ computes and returns $\sum_{i \in W} \Lambda'_{i,W} \cdot y_i$.
- $\mathsf{f}(k_0, x)$ returns the output of $F(k_0, x)$.

Let pp be the output of $\mathsf{Setup}(1^\lambda)$, k_0 be chosen uniformly from $\mathbb{F}_1$, $(k_1, \ldots, k_n)$ be the output of $\mathsf{Share}(k_0)$, $W = \{i_1, \ldots, i_t\} \subset [N]$, and $x \in \mathcal{X}$. The combine algorithm computes $\mathsf{G}\left(W, \mathsf{F}(k_{i_1}, x), \ldots, \mathsf{F}(k_{i_t}, x)\right) = \sum_{i \in W} \Lambda'_{i,W} \cdot F(k_i, x) = \sum_{i \in W} F\left(\Lambda_{i,W} \cdot k_i, x\right) = \mathsf{F}(k_0, x)$ as required. If F is a key homomorphic pseudorandom function, then the following theorem shows that Π_{dPRF} is a secure distributed PRF.

Theorem 7.2. *If F is a key homomorphic PRF, then Π_{dPRF} is a secure distributed PRF.*

Proof. In this security proof, we consider a modified version of the standard PRF security definition (see Definition A.1). Consider experiments $\mathsf{Expt}_b^{\mathsf{PRF}'}$ for $b \in \{0, 1\}$, where $\mathsf{Expt}_b^{\mathsf{PRF}'}$ differs from $\mathsf{Expt}_b^{\mathsf{PRF}}$ in the following way. An adversary interacts with a challenger that answers queries in a "PRF query phase" with *honest* outputs of the PRF on a randomly chosen key k, and behaves like either a PRF or a truly random function on "challenge queries." As long as the two sets of queries

are disjoint, we require adversaries have negligible advantage in distinguishing the experiments. Up to a constant factor (of 2) in advantage, this definition is equivalent to Definition A.1.[4]

Given an adversary $\mathcal{A}$ against the distributed PRF, we construct an algorithm $\mathcal{B}$ that breaks the pseudorandomness of the PRF F. Given access to a PRF challenger $\mathsf{Expt}_b^{\mathsf{PRF}'}$ for a bit $b \in \{0,1\}$, algorithm $\mathcal{B}$ simulates $\mathcal{A}$ in $\mathsf{Expt}_b^{\mathsf{proxy}}$. Algorithm $\mathcal{B}$ receives public parameters pp from the challenger and publishes them to the adversary. $\mathcal{A}$ specifies a set $S^* = \{i_1, \ldots, i_{t-1}\}$ of $t-1$ servers to $\mathcal{B}$ which then samples key-shares $k_{i_1}, \ldots, k_{i_{t-1}}$ uniformly from $\mathbb{F}_1$. It returns $(k_{i_1}, \ldots, k_{i_{t-1}})$ to $\mathcal{A}$. For each secret-share query $x_j \in \mathcal{X}$ made by the adversary for $j \in [1, Q]$, the simulator forwards x_j to the PRF challenger and receives a response $y_j = F(k, x_j)$ for some unknown key k. Let $W = S^* \cup \{0\}$. For all $\ell \in [1, N] \setminus S^*$, algorithm $\mathcal{B}$ computes

$$u_\ell = \sum_{i \in S^*} \mathsf{F}\left(\Lambda_{i,W}(\ell) \cdot k_i, x\right) + \Lambda'_{0,W}(\ell) \cdot y_j \tag{7.1}$$

and responds with u_ℓ as the purported value of $\mathsf{F}(k_\ell, x)$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ submits a challenge query $x^* \in \mathcal{X} \setminus \{x_1, \ldots, x_Q\}$ to $\mathcal{B}$. We consider a single challenge query and note that the proof generalizes to multiple adaptive queries in a straightforward manner. On input x^*, $\mathcal{B}$ forwards x^* to the PRF challenger as a challenge query and relays its response y^* back to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a bit b' which is also output by $\mathcal{B}$.

For each bit $b \in \{0,1\}$, the algorithm $\mathcal{B}$ simulates $\mathcal{A}$'s interaction with $\mathsf{Expt}_b^{\mathsf{dPRF}}$ identically to a real scheme. Let $k_0 = k$ be the key used by the PRF F. For each query x by $\mathcal{A}$, it holds that $u_\ell = \sum_{i \in S^*} \mathsf{F}\left(\Lambda_{i,W}(\ell) \cdot k_i, x\right) + \Lambda'_{0,W}(\ell) \cdot F(k_0, x) = F\left(\sum_{i \in W} \Lambda_{i,W}(\ell) \cdot k_i, x\right)$ by the key homomorphism property of F. Therefore, we have that $u_\ell = F(k_\ell, x) = \mathsf{F}(k_\ell, x)$ as desired.

If $\mathcal{B}$ is interacting with experiment $\mathsf{Expt}_0^{\mathsf{PRF}'}$ where it receives outputs of a random function from $\mathcal{X} \to \mathbb{F}_2$, then the value y^* received by $\mathcal{A}$ is uniform in $\mathbb{F}_2$ as in $\mathsf{Expt}_0^{\mathsf{dPRF}}$. If $\mathcal{B}$ is interacting with $\mathsf{Expt}_1^{\mathsf{PRF}'}$ where it forwards to $\mathcal{A}$ the value $y^* = F(k_0, x^*) = \mathsf{F}(k_0, x^*)$, then $\mathcal{A}$ receives a response to the challenge query as in $\mathsf{Expt}_1^{\mathsf{dPRF}}$. We therefore conclude that $\mathbf{Adv}^{\mathsf{PRF}}[F, \mathcal{B}] = \mathbf{Adv}^{\mathsf{dPRF}}[\Pi_{\mathsf{dPRF}}, \mathcal{A}]$. ∎

Constructing distributed PRFs from almost key homomorphic PRFs. The construction described above for a distributed PRF can be adapted to PRFs that are γ-almost key homomorphic with output space $\mathbb{Z}_p$ for some prime p (for example, our LWE-based PRF F_{LWE} in Section 5 restricted to a single coordinate). Recollect that for a scalar $T \in \mathbb{Z}$, primes $q > p$ in $\mathbb{N}$, and a γ-almost key homomorphic PRF $F\colon \mathbb{Z}_q \times \mathcal{X} \to \mathbb{Z}_p$, for every $x \in \mathcal{X}$ it holds that $T \cdot F(k_0, x) = F(T \cdot k_0, x) + e$ for an "error term" $e \in \{0, \ldots, \gamma T\}$. The main difficulty in using an almost key homomorphic PRF to instantiate Π_{dPRF} is to ensure that when multiplying with Lagrange coefficients $\Lambda_{i,W}$ (for $i \in [N]$ and $W \in 2^{[N]}$) in the combine phase, the "error term" $e \in \{0, \ldots, \gamma\Lambda_{i,W}\}$ does not become too large.

Unfortunately, when Lagrange coefficients are interpreted as elements in $\mathbb{Z}_p$, they are no longer low-norm and the error term becomes too large. To overcome this difficulty, we use the technique of "clearing the denominator" [Sho00, ABV+12]. We use the fact that for every Lagrange coefficient

[4] To see this, consider an intermediate hybrid between $\mathsf{Expt}_0^{\mathsf{PRF}'}$ and $\mathsf{Expt}_1^{\mathsf{PRF}'}$ where even PRF-queries are answered as outputs of a truly random function. This hybrid is indistinguishable from $\mathsf{Expt}_0^{\mathsf{PRF}'}$ in a straightforward manner from F's pseudorandomness (as per Definition A.1). It is indistinguishable from $\mathsf{Expt}_1^{\mathsf{PRF}'}$ because in each of the experiments, PRF-queries are identical to queries in $\mathsf{Expt}_1^{\mathsf{PRF}}$ or $\mathsf{Expt}_0^{\mathsf{PRF}}$ respectively and challenge queries are always answered with a truly random function.

$\Lambda_{i,W}$, it holds that the quantity $N! \cdot \Lambda_{i,W}$ is an integer. The combine algorithm multiplies all Lagrange coefficients by $N!$ to avoid interpreting them as elements in $\mathbb{Z}_p$. To show correctness, it suffices to choose a rounding parameter u such that for every Lagrange coefficient $\Lambda_{i,W}$, the error term introduced by the almost key homomorphic PRF $(\gamma \cdot N! \cdot \Lambda_{i,W})$ can be eliminated by suitably rounding down to elements in $[0, u-1]$. As the numerator of $\Lambda_{i,W}$ is upper-bounded by $N!$, it suffices to set $\lfloor p/u \rfloor > 2\gamma(N!)^2$ to ensure that the evaluation function is always correctly reconstructed, with an output space of size u. The scheme Π_{dPRF} is modified as follows.

- $\mathsf{Setup}(1^\lambda)$ samples and publishes public parameters pp used by the key homomorphic PRF F.
- $\mathsf{Share}(k_0)$ samples a uniformly random polynomial $p(z)$ of degree $t-1$ such that $p(0) = k_0$ and outputs $(p(1), \ldots, p(N))$.
- $\mathsf{F}(k_i, x)$ returns the output of $F(k_i, x)$.
- $\mathsf{G}(W, y_1, \ldots, y_t)$ computes and returns $\left\lfloor \sum_{i \in W} N! \cdot \Lambda'_{i,W} \cdot y_i \right\rceil_u$.
- $\mathsf{f}(k_0, x)$ returns the output of $\lfloor N! \cdot F(k_0, x) \rfloor_u$.

Pseudorandomness. The proof of pseudorandomness follows the outline of the proof of Theorem 7.2 closely. Queries in the adversary's challenge phase can be simulated given access to an oracle $f(\cdot)$ (that is either a truly random function or $F(k, \cdot)$) and returning to the adversary the outputs of $\lfloor N! \cdot f(\cdot) \rfloor_u$. Queries corresponding to partial evaluations $\mathsf{F}(k_\ell, x)$ for all $\ell \in [1, N] \setminus S^*$ are answered along the lines of Equation (7.1) taking into account the technique of clearing the denominator (by simply multiplying throughout by $N!$). The rest of the arguments in the proof follow identically to the ones presented in the proof of Theorem 7.2.

7.2 Symmetric Proxy Re-Encryption

Next we show that key homomorphic PRFs give rise to symmetric proxy re-encryption schemes. In a proxy re-encryption scheme, a proxy is given re-encryption information that enables the proxy to translate an encryption of any message from one key to an encryption of the same message under another key without revealing the underlying message.

Proxy re-encryption has been studied extensively. We consider the setting of achieving symmetric proxy re-encryption under semantic security. This notion has been considered before [SNS11], but we provide a formal treatment of the security model here. We consider a security model inspired by the work of Canetti and Hohenberger [CH07]. We adapt their definition to the symmetric-key setting by additionally providing access to an encryption oracle. We do not consider chosen-ciphertext security in this work. Thus, the extra restrictions placed on the re-encryption oracle from [CH07] are not required in this model. A symmetric proxy re-encryption scheme is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$ with the following properties.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. On input the security parameter λ, the setup algorithm Setup outputs the public parameters pp used for the scheme.
- $\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$. On input the security parameter λ, the key generation algorithm KeyGen outputs a secret key sk.

- $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2) \to \mathsf{rk}_{1,2}$. On input two secret keys sk_1 and sk_2, the re-encryption key generation algorithm $\mathsf{ReKeyGen}$ outputs a bidirectional re-encryption key $\mathsf{rk}_{1,2}$.
- $\mathsf{Enc}(\mathsf{sk}, m) \to C$. On input a secret key sk and message m, the encryption algorithm Enc outputs a ciphertext C.
- $\mathsf{ReEnc}(\mathsf{rk}_{1,2}, C_1) \to C_2$. On input a re-encryption key $\mathsf{rk}_{1,2}$ and a ciphertext C_1, the re-encryption algorithm ReEnc outputs a second ciphertext C_2 or $\perp$.
- $\mathsf{Dec}(\mathsf{sk}, C) \to m$. On input a secret key sk and a ciphertext C, the decryption algorithm Dec outputs a message m or the error symbol $\perp$.

Correctness. A symmetric proxy re-encryption scheme Π is *T-time correct* if, under public parameters $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and for all $m_i \in \mathcal{M}$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m$, and for any sequence of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$ output by $\mathsf{KeyGen}(1^\lambda)$ and re-encryption keys $\mathsf{rk}_{i,i+1}$ output by $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_{i+1})$ for $i \in [1, T-1]$, for all messages $m \in \mathcal{M}$ and all ciphertexts C output by $\mathsf{Enc}(\mathsf{sk}_1, m)$, it holds that $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{ReEnc}(\mathsf{rk}_{T-1,T}, \cdots \mathsf{ReEnc}(\mathsf{rk}_{1,2}, C) \cdots)) = m$.

Security. The experiment below is between a challenger and an adversary $\mathcal{A}$, which can make oracle queries of the following six different types: uncorrupted key generation, corrupted key generation, re-encryption key generation, challenge, re-encryption, and encryption. A public counter ctr (always visible $\mathcal{A}$) is maintained by the challenger and initialized to 0. This counter keeps track of the number of key generation queries (both uncorrupted and corrupted) made by $\mathcal{A}$. First, the challenger samples and publishes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ to $\mathcal{A}$. For $b \in \{0, 1\}$ the challenger in $\mathsf{Expt}_b^{\mathsf{proxy}}$ responds to each of the adversary's types of oracle queries as follows.

- Uncorrupted key generation: The challenger samples $\mathsf{KeyGen}(1^\lambda)$ to obtain a secret key which is set to $\mathsf{sk}_{\mathsf{ctr}}$ but not given to $\mathcal{A}$. The challenger then increments ctr.
- Corrupted key generation: The challenger samples $\mathsf{KeyGen}(1^\lambda)$ to obtain a secret key which is set to $\mathsf{sk}_{\mathsf{ctr}}$, which is given to $\mathcal{A}$. The challenger then increments ctr.
- Re-encryption key generation: On input (i, j) by $\mathcal{A}$, where $i, j < \mathsf{ctr}$, the challenger returns $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j)$ to $\mathcal{A}$. We require that either both sk_i and sk_j were generated from uncorrupted key generation, or both sk_i and sk_j were generated from corrupted key generation. We do not allow $\mathcal{A}$ to make a re-encryption key generation request when one secret key is corrupted and the other is uncorrupted.
- Challenge: This oracle can be queried only once. On input (i^*, m_0^*, m_1^*), if sk_{i^*} was generated from a call to uncorrupted key generation, then the challenger returns the challenge ciphertext $C^* := \mathsf{Enc}(sk_{i^*}, m_b^*)$ to $\mathcal{A}$. Otherwise, the challenger returns $\perp$.
- Re-encryption: On input (i, j, C), where $i, j < \mathsf{ctr}$, if sk_j was generated from a call to corrupted key generation, then return $\perp$. Otherwise, return the re-encrypted ciphertext $C' := \mathsf{ReEnc}(\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j), C)$.
- Encryption: On input (i, m), where $i < \mathsf{ctr}$, the challenger returns $\mathsf{Enc}(\mathsf{sk}_i, m)$ to $\mathcal{A}$.

Eventually, the adversary outputs a bit $b' \in \{0,1\}$. Let W_b denote the probability that $\mathcal{A}$ outputs 1 in experiment $\mathsf{Expt}_b^{\mathsf{proxy}}$.

Definition 7.3 (Symmetric Proxy Re-Encryption)**.** A symmetric proxy re-encryption scheme Π is secure if for all efficient adversaries $\mathcal{A}$ the quantity

$$\mathbf{Adv}^{\mathsf{proxy}}[\Pi, \mathcal{A}] := |W_0 - W_1|$$

is negligible.

For schemes which satisfy T-time correctness for some fixed T, we restrict the adversary to re-encryption queries which form chains (of re-encrypted ciphertexts) of length at most T. We show how to achieve a symmetric proxy re-encryption scheme with T-time correctness for all $T > 1$ under this security model using key homomorphic PRFs.

Symmetric Proxy Re-encryption from key homomorphic PRFs. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a key homomorphic PRF with superpolynomial input set size ($|\mathcal{X}| = \omega(\text{poly}(\lambda))$), and where $(\mathcal{X}, +)$ and $(\mathcal{Y}, +)$ are groups. Let $\Pi_{\mathsf{proxy}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$ be defined as follows.

- $\mathsf{Setup}(1^\lambda)$ samples and outputs the public parameters pp used by F.
- $\mathsf{KeyGen}(1^\lambda)$ outputs a uniform secret key sk from the key space $\mathcal{K}$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2)$ returns $\mathsf{rk}_{1,2} = \mathsf{sk}_2 - \mathsf{sk}_1$.
- $\mathsf{Enc}(\mathsf{sk}, m)$ chooses a random $r \leftarrow \mathcal{X}$ and outputs $(r, m + F(\mathsf{sk}, r))$.
- $\mathsf{ReEnc}(\mathsf{rk}_{1,2}, (r, C))$ outputs $(r, C + F(\mathsf{rk}_{1,2}, r))$.
- $\mathsf{Dec}(\mathsf{sk}, (r, C))$ outputs $C - F(\mathsf{sk}, r)$.

Correctness. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $m \in \mathcal{M}$. Then, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m + F(\mathsf{sk}, r) - F(\mathsf{sk}, r) = m$ as desired for all $r \in \mathcal{X}$. Also, for all $T \geq 1$, for any sequence of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$ output by $\mathsf{KeyGen}(1^\lambda)$ and re-encryption keys $\mathsf{rk}_{i,i+1} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_{i+1})$ along with ciphertext $C \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{ReEnc}(\mathsf{rk}_{T-1,T}, \cdots \mathsf{ReEnc}(\mathsf{rk}_{1,2}, C_1) \cdots)) = m + F(\mathsf{sk}_1, r) + F(\mathsf{sk}_2 - \mathsf{sk}_1, r) + \ldots + F(\mathsf{sk}_T - \mathsf{sk}_{T-1}, r) - F(\mathsf{sk}_T, r) = m$ as desired.

If F is a key homomorphic pseudorandom function with superpolynomial input set size, then the following theorem shows that Π_{proxy} is a secure symmetric proxy re-encryption scheme.

Theorem 7.4. *If F is a key homomorphic PRF, then Π_{proxy} is a secure symmetric proxy re-encryption scheme.*

Proof. We use a hybrid argument to prove the security of this theorem. Consider the experiments Expt_0 and Expt_2 interacting with an adversary $\mathcal{A}$ that are identical to $\mathsf{Expt}_0^{\mathsf{proxy}}$ and $\mathsf{Expt}_1^{\mathsf{proxy}}$ respectively. The intermediate experiment Expt_1 is identical to Expt_0 except in the response of the challenge oracle: instead of outputting the ciphertext $C^* := \mathsf{Enc}(\mathsf{sk}_{i^*}, m_0^*)$ (if sk_{i^*} was generated from a call to the uncorrupted key generation oracle), it samples a uniform $u \leftarrow \mathcal{C}$, the ciphertext space, and outputs $C^* := u$. Let W_n for $n \in \{0, 1, 2\}$ denote the probability that $\mathcal{A}$ outputs 1 interacting with Expt_n.

Recollect the modified version of the standard PRF security definition (see Definition A.1 and the proof of Theorem 7.2). In what follows, we show that for every adversary $\mathcal{A}$ making at most Q queries to the encryption oracle when interacting with Expt_0 and Expt_1, there is a PRF adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathsf{PRF}'}[F, \mathcal{B}] \geq (1 - Q/|\mathcal{X}|) \cdot |W_0 - W_1|$. Algorithm $\mathcal{B}$ receives public parameters pp from the PRF challenger and sends pp to $\mathcal{A}$. Algorithm $\mathcal{B}$ simulates the responses to each query type made by $\mathcal{A}$ in the following manner.

- Uncorrupted key generation: $\mathcal{B}$ samples δ_{ctr} uniformly at random in $\mathcal{K}$ without revealing it to $\mathcal{A}$, and then increments ctr.
- Corrupted key generation: $\mathcal{B}$ sets $\mathsf{sk}_{\mathsf{ctr}} \leftarrow \mathsf{KeyGen}(1^\lambda)$, returning $\mathsf{sk}_{\mathsf{ctr}}$ to $\mathcal{A}$, and then increments ctr.
- Re-encryption key generation: On input (i, j) where $i, j < \mathsf{ctr}$, if both sk_i and sk_j were generated from uncorrupted key generation, then $\mathcal{B}$ returns $\delta_j - \delta_i$. If both sk_i and sk_j were generated from corrupted key generation, then $\mathcal{B}$ returns $\mathsf{sk}_j - \mathsf{sk}_i$. Otherwise, $\mathcal{B}$ returns $\perp$.
- Challenge: On input (i^*, m_0, m_1), if sk_{i^*} was generated from corrupted key generation, then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ samples r uniformly at random in $\mathcal{X}$ and sends r as the challenge input to the PRF challenger, receiving challenge response $f(r)$. $\mathcal{B}$ outputs $C^* = (r, m_z + F(\delta_{i^*}, r) + f(r))$ to $\mathcal{A}$.
- Re-encryption: On input $(i, j, (r, C))$, if sk_j was generated from corrupted key generation, then $\mathcal{B}$ returns $\perp$. Otherwise, if sk_i was generated from corrupted key generation, $\mathcal{B}$ makes a query to the PRF challenger on input r and obtains response $f(r)$, and then returns $(r, C + F(\delta_j - \mathsf{sk}_i, r) + f(r))$ to $\mathcal{A}$. If both sk_i and sk_j were generated from uncorrupted key generation, then $\mathcal{B}$ returns $(r, C + F(\delta_j - \delta_i, r))$ to $\mathcal{A}$.
- Encryption: On input (i, m), $\mathcal{B}$ samples r uniformly at random from $\mathcal{X}$. $\mathcal{B}$ sends r as input to the PRF challenger and obtains response $f(r)$. $\mathcal{B}$ returns the ciphertext $(r, m + F(\delta_i, r) + f(r))$ to $\mathcal{A}$.

Eventually $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. Let r^* denote the randomness used by $\mathcal{B}$ to answer the adversary's challenge oracle query and $r_1, \ldots, r_Q$ the randomness used to answer the remaining Q queries. If $r \in \{r_1, \ldots, r_Q\}$, $\mathcal{B}$ aborts and outputs a uniform bit. Otherwise, it outputs b'.

The randomness strings $r^*, r_1, \ldots, r_Q$ are sampled uniformly at random in $\mathcal{X}$ and independent of the adversary's view. Thus, the probability that $\mathcal{B}$ aborts is $1 - Q/|\mathcal{X}|$ and is independent of whether it simulates Expt_0 or Expt_1. From here on, it suffices to only consider the case when $\mathcal{B}$ does not abort.

Consider the case $\mathcal{B}$ interacts with $\mathsf{Expt}_0^{\mathsf{PRF}'}$ (i.e., it interacts with a pseudorandom function during challenge queries). We let K^* denote the (implicit) key chosen by the PRF challenger. Uncorrupted keys sk_i simulated by $\mathcal{B}$ are implicitly set to $\mathsf{sk}_i := \delta_i + K^*$. As δ_i is chosen uniformly at random from $\mathcal{K}$, uncorrupted keys are correctly generated. Corrupted keys are always generated correctly. On input (i, j), for uncorrupted sk_i and sk_j, re-encryption keys returned by $\mathcal{B}$ are $\delta_i - \delta_j = (sk_i - K^*) - (sk_j - K^*) = \mathsf{sk}_i - \mathsf{sk}_j$, as required. The key homomorphism of F implies that on input m to the encryption oracle, $\mathcal{B}$ computes $F(\delta_i, r) + F(K^*, r) = F(\delta_i + K^*, r) = F(\mathsf{sk}_i, r)$. Thus, $C^* = \mathsf{Enc}(\mathsf{sk}_{i^*}, m_0^*)$ as required. Finally, through identical arguments, it holds that $\mathcal{B}$ honestly simulates re-encryption queries from $\mathcal{A}$. Thus, $\mathcal{B}$ simulates Expt_0 correctly.

Next, we consider the case $\mathcal{B}$ interacts with $\mathsf{Expt}_1^{\mathsf{PRF}'}$. On queries x to the PRF-challenger, $\mathcal{B}$ receives $F(K^*, x)$ for some implicit K^* and on input a challenge query x^*, receives a uniform $y^* \in \mathcal{Y}$. As $\mathcal{B}$ does not abort, the query r^* used to construct the challenge ciphertext C^* is not in the set of previously queried inputs. Observe that the output y^* is used *only* in answering the challenge ciphertext. The rest of the oracles simulated by $\mathcal{B}$ use the pseudorandom function F evaluated honestly on inputs. Therefore, $\mathcal{B}$ simulates all the oracles honestly (as shown in the case when $\mathcal{B}$ interacts with $\mathsf{Expt}_0^{\mathsf{PRF}'}$ above). The output of the challenge encryption query $C^* := \langle r, m_0^* + F(\delta_{i^*}, r) + y^* \rangle$ which is distributed uniformly in the ciphertext space. Thus, $\mathcal{B}$ simulates Expt_1.

Conditioned on $\mathcal{B}$ not aborting, it holds that it interacts with $\mathcal{A}$ in experiments Expt_b when interacting with $\mathsf{Expt}_b^{\mathsf{PRF}'}$ for both $b \in \{0, 1\}$. This implies that $\mathbf{Adv}^{\mathsf{PRF}'}[F, \mathcal{B}] \geq (1 - Q/|\mathcal{X}|) \cdot |W_0 - W_1|$ as required.

To complete the proof, we consider an almost identical argument starting with Expt_2 where m_1^* is used in the challenge phase and all other oracles are simulated identically. Then, it holds that $\mathbf{Adv}^{\mathsf{PRF}'}[F, \mathcal{B}] \geq (1 - Q/|\mathcal{Y}|) \cdot |W_1 - W_2|$. Applying a triangle inequality, we conclude that $|W_0 - W_2| \leq (1 - Q/|\mathcal{X}|)^{-1} \cdot \mathbf{Adv}^{\mathsf{PRF}'}[F, \mathcal{B}]$. As Q is polynomial in the security parameter, and $|\mathcal{X}|$ is at least superpolynomial, if F is a pseudorandom function, the adversary's advantage against the proxy re-encryption scheme is negligible. This completes the proof of Theorem 7.4. ∎

Using almost key homomorphic PRFs. Let $F : \mathcal{K} \times \mathcal{X} \to \mathbb{Z}_p$ be a γ-almost key homomorphic PRF. Let $T \geq 1$ be a fixed integer, and let u be an integer such that $\lfloor p/u \rfloor > 2\gamma T$. We show how to construct a symmetric proxy re-encryption scheme with T-time correctness, where $\mathbb{Z}_u$ is the message space. The scheme Π_{proxy} is defined as follows, where Setup takes an additional parameter T.

- $\mathsf{Setup}(1^\lambda, T)$ samples and outputs the public parameters pp used by F.
- $\mathsf{KeyGen}(1^\lambda)$ samples and outputs a secret key sk uniformly at random from the key space $\mathcal{K}$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2)$ returns $\mathsf{sk}_2 - \mathsf{sk}_1$.
- $\mathsf{Enc}(\mathsf{sk}, m)$ chooses a random $r \leftarrow \mathcal{X}$ and outputs $(r, \bar{m} \cdot \lfloor p/u \rfloor + F(\mathsf{sk}, r))$, where $\bar{m}$ represents the integer corresponding to m.
- $\mathsf{ReEnc}(\mathsf{rk}_{1,2}, (r, C))$ outputs $(r, C + F(\mathsf{rk}_{1,2}, r))$.
- $\mathsf{Dec}(\mathsf{sk}, (r, C))$ outputs $\lfloor C - F(\mathsf{sk}, r) \pmod p \rfloor_u$.

To verify that Π_{proxy} has T-time correctness, it suffices to observe that the error accumulated by re-encryptions does not exceed γT, and hence will be rounded away upon decryption.

Correctness. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $m \in \mathcal{M}$. Then, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = \lfloor \bar{m} \cdot \lfloor p/u \rfloor + F(\mathsf{sk}, r) - F(\mathsf{sk}, r) \rfloor_u = m$ as desired for all $r \in \mathcal{X}$. Also, for any sequence of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$ output by $\mathsf{KeyGen}(1^\lambda)$ and re-encryption keys $\mathsf{rk}_{i,i+1} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_{i+1})$ along with ciphertext $C \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$, let $e = F(\mathsf{sk}_1, r) + F(\mathsf{sk}_2 - \mathsf{sk}_1, r) + \ldots + F(\mathsf{sk}_T - \mathsf{sk}_{T-1}, r) - F(\mathsf{sk}_T, r)$. Then, $e \in \{0, \ldots, T\gamma\}$. Hence, we have that $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{ReEnc}(\mathsf{rk}_{T-1,T}, \cdots \mathsf{ReEnc}(\mathsf{rk}_{1,2}, C_1) \cdots)) = \lfloor \bar{m} \cdot \lfloor p/u \rfloor + e \pmod p \rfloor_u = m$ as desired.

Remark 7.5. For T-time correctness, the ciphertext size of the scheme Π_{proxy} grows logarithmically in γT. In other words, Π_{proxy} is efficiently implementable even if T is superpolynomial in λ.

The proof of semantic security remains unchanged, noting that responses to re-encryption queries can be simulated, so long as chains of re-encrypted ciphertexts do not exceed length T.

7.3 Updatable Encryption

We noted in the introduction that key homomorphic PRFs can be used to construct an efficient key rotation scheme for encrypted data stored in the cloud. We call such a scheme an *updatable encryption* scheme. They are closely related to proxy re-encryption systems, discussed in the previous section, with two important differences.

First, an updatable encryption scheme Π is the same as a symmetric proxy re-encryption scheme, except that the re-encryption key generation algorithm ReEnc takes as input two secret keys sk_1 and sk_2 along with a ciphertext C and outputs a short *uni-directional* re-encryption key $\mathsf{rk}_{1,2,C}$. The length of this re-encryption key $\mathsf{rk}_{1,2,C}$ must be *independent* of the size of the ciphertext to which it will be used.

Second, the result of re-encrypting a ciphertext C should result in a re-encrypted ciphertext C' that appears as a fresh encryption of the data that is independent from C. We formalize both requirements below.

Correctness. An updatable encryption scheme Π is *T-time correct* if, under public parameters $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$ and for all $m_i \in \mathcal{M}$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m$, and for all messages $m \in \mathcal{M}$, for any sequence of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$ output by $\mathsf{KeyGen}(1^\lambda)$, ciphertexts C_i and re-encryption keys $\mathsf{rk}_{i,i+1,C_i}$ given by $C_1 \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$ and $C_i \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i-1,i,C_{i-1}}, C_{i-1})$ and $\mathsf{rk}_{i,i+1,C_i} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_{i+1}, C_i)$ for $i \in [2, T-1]$, it holds that

$$\mathsf{Dec}(\mathsf{sk}_T, \mathsf{ReEnc}(\mathsf{rk}_{T-1,T,C_{T-1}}, \cdots \mathsf{ReEnc}(\mathsf{rk}_{1,2,C_1}, C_1) \cdots)) = m \ .$$

Security. The following experiment $\mathsf{Expt}_b^{\mathsf{update}}$ is identical to that of the experiment $\mathsf{Expt}_b^{\mathsf{proxy}}$ defined for a symmetric proxy re-encryption scheme, except for the following modifications to the re-encryption key generation and re-encryption oracle. All other oracles remain unchanged.

- Re-encryption key generation: On input (i, j, C) by $\mathcal{A}$, where $i, j < \mathsf{ctr}$, the challenger returns $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j, C)$ to $\mathcal{A}$. We require that either both sk_i and sk_j were generated from uncorrupted key generation, or both sk_i and sk_j were generated from corrupted key generation. We do not allow $\mathcal{A}$ to make a re-encryption key generation request when one secret key is corrupted and the other is uncorrupted.

- Re-encryption: On input (i, j, C), where $i, j < \mathsf{ctr}$, if sk_j was generated from a call to corrupted key generation, then return $\perp$. Otherwise, return the re-encrypted ciphertext $C' := \mathsf{ReEnc}(\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j, C), C)$.

Next, we formalize the requirement that a re-encrypted ciphertext C' should appear as a fresh encryption of the data that is independent from the input ciphertext C.

Definition 7.6 (Ciphertext Independence)**.** Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $m \in \mathcal{M}$. Let $C \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$ and $C' \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$, and let $\mathsf{rk}_{1,2,C} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2, C)$ and $\mathsf{rk}_{1,2,C'} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2, C')$. An updatable encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$ is ciphertext independent if the two joint distributions $(C, \mathsf{ReEnc}(\mathsf{rk}_{1,2,C}, C))$ and $(C, \mathsf{ReEnc}(\mathsf{rk}_{1,2,C'}, C'))$ are identical.

Let W_b denote the probability that an adversary $\mathcal{A}$ outputs 1 in experiment $\mathsf{Expt}_b^{\mathsf{update}}$.

Definition 7.7 (Updatable Encryption)**.** An updatable encryption scheme Π is secure if it is ciphertext independent and for all efficient adversaries $\mathcal{A}$ the quantity

$$\mathbf{Adv}^{\mathsf{update}}[\Pi, \mathcal{A}] := |W_0 - W_1|$$

is negligible.

7.3.1 Updatable Encryption from seed homomorphic PRGs.

Let $G : \mathcal{X} \to \mathcal{Y}$ be a seed homomorphic PRG, and let $(\mathsf{SymKeyGen}, \mathsf{E}, \mathsf{D})$ be a symmetric key CPA-secure encryption scheme with key space $\mathcal{K}$, where $(\mathcal{X}, +)$ and $(\mathcal{Y}, +)$ are groups. We show how to construct a secure updatable encryption scheme. The scheme Π_{update} is defined as follows.

- $\mathsf{Setup}(1^\lambda)$ samples and outputs the public parameters pp used by G.
- $\mathsf{KeyGen}(1^\lambda)$ samples and outputs a secret key $\mathsf{sk} \leftarrow \mathsf{SymKeyGen}(1^\lambda)$.
- $\mathsf{Enc}(\mathsf{sk}, m)$ chooses a random $x \in \mathcal{X}$ and outputs $C = (\mathsf{E}(\mathsf{sk}, x), m + G(x))$.
- $\mathsf{Dec}(\mathsf{sk}, C = (C_1, C_2))$ outputs $C_2 - G(\mathsf{D}(\mathsf{sk}, C_1))$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2, C = (C_1, C_2))$ chooses a random $x_2 \in \mathcal{X}$, sets $x_1 = \mathsf{D}(\mathsf{sk}_1, C_1)$, and returns the tuple $\mathsf{rk}_{1,2,C} \leftarrow \big(\mathsf{E}(\mathsf{sk}_2, x_2),\ x_2 - x_1\big)$.
- $\mathsf{ReEnc}(\mathsf{rk}_{1,2,C} = (r, s), C = (C_1, C_2))$ outputs $(r, C_2 + G(s))$.

Correctness. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $m \in \mathcal{M}$. Then, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m + G(x) - G(\mathsf{D}(\mathsf{sk}, \mathsf{E}(\mathsf{sk}, x))) = m$. as desired for all $x \in \mathcal{X}$. Also, for all messages $m \in \mathcal{M}$, for any sequence of secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_T$ output by $\mathsf{KeyGen}(1^\lambda)$, ciphertexts C_i and re-encryption keys $\mathsf{rk}_{i,i+1,C_i}$ given by $C_1 \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m)$ and $C_i \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i-1,i,C_{i-1}}, C_{i-1})$ and $\mathsf{rk}_{i,i+1,C_i} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_{i+1}, C_i)$ for $i \in [2, T-1]$, it holds that $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{ReEnc}(\mathsf{rk}_{T-1,T,C_{T-1}}, \cdots \mathsf{ReEnc}(\mathsf{rk}_{1,2,C_1}, C_1) \cdots)) = m + G(x_1) + G(x_2 - \mathsf{D}(\mathsf{sk}_1, \mathsf{E}(\mathsf{sk}_1, x_1))) + \cdots + G(x_T - \mathsf{D}(\mathsf{sk}_{T-1}, \mathsf{E}(\mathsf{sk}_{T-1}, x_{T-1}))) - G(\mathsf{D}(\mathsf{sk}_T, \mathsf{E}(\mathsf{sk}_T, x_T))) = m$ as desired, where $x_1, \ldots, x_T \in \mathcal{X}$ are the random elements chosen by Enc and $\mathsf{ReKeyGen}$.

Security. For semantic security, we note that for any given ciphertext $C = (C_1, C_2)$ output by $\mathsf{Enc}(\mathsf{sk}, m)$ for some message $m \in \mathcal{M}$ and $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, the first component C_1 hides the seed of the PRG used to construct C_2, since C_1 is simply an encryption of this seed under a CPA-secure symmetric encryption scheme. Then, since the seed is computationally hidden from an adversary, by the pseudorandomness of G, the second component C_2 reveals no information to a computationally bounded adversary about the message. For a re-encryption key $\mathsf{rk}_{i,j,C} = (r, s)$ of a

ciphertext C, we note that again by the CPA security of the symmetric encryption scheme that the component r reveals no information about the randomly chosen element $x_2 \in \mathcal{X}$, and since x_1 is not revealed by C, the component $s = x_2 - x_1$ also does not reveal x_2 nor x_1, and so the the generation of the re-encryption keys can be simulated without knowing any secret key information. The remainder of the security proof for the indistinguishability between $\mathsf{Expt}_0^{\mathsf{update}}$ and $\mathsf{Expt}_1^{\mathsf{update}}$ follows straightforwardly in a manner similar to the proof of security for the symmetric proxy re-encryption scheme from Section 7.2.

We now show that Π_{update} is ciphertext independent. For a fixed message $m \in \mathcal{M}$ and let $\mathsf{sk}_1, \mathsf{sk}_2$ be a pair of secret keys output by two calls to $\mathsf{KeyGen}(1^\lambda)$. Let $x, x' \in \mathcal{X}$ be such that $C = (\mathsf{E}(\mathsf{sk}, x), m + G(x))$ and $C' = (\mathsf{E}(\mathsf{sk}, x'), m + G(x'))$. Also, let $r, r' \in \mathcal{X}$ be such that $\mathsf{rk}_{1,2,C} = (\mathsf{E}(\mathsf{sk}_2, r), r - x)$ and $\mathsf{rk}_{1,2,C'} = (\mathsf{E}(\mathsf{sk}_2, r'), r' - x')$. Then, $\mathsf{ReEnc}(\mathsf{rk}_{1,2,C}, C) = (\mathsf{E}(\mathsf{sk}_2, r), m + G(x) + G(r - x)) = (\mathsf{E}(\mathsf{sk}_2, r), m + G(r))$, and is hence independent of x. Thus, $\mathsf{ReEnc}(\mathsf{rk}_{1,2,C}, C)$ is distributed independently of C and hence identical to $\mathsf{ReEnc}(\mathsf{rk}_{1,2,C'}, C') = (\mathsf{E}(\mathsf{sk}_2, r'), m + G(r')$, since r and r' were drawn from the same distribution.

Using almost seed homomorphic PRGs. With a γ-almost seed homomorphic PRG instead of a seed homomorphic PRG, the above construction Π_{update} can be modified to yield a symmetric proxy re-encryption scheme which satisfies T-time correctness for a pre-specified $T \geq 1$. This is done by padding the message in the encryption and then rounding away the accumulated error from re-encryptions during the decryption phase, similar to the modification made to the encryption and decryption algorithms of the symmetric proxy re-encryption scheme in Section 7.2. This new updatable encryption scheme achieves T-time correctness for a pre-specified T in the setup algorithm, and the ciphertext size of the scheme grows logarithmically in γT. We also note that the ciphertext independence property still holds when G is γ-almost seed homomorphic.

7.3.2 A more efficient construction from key homomorphic PRFs.

Using a key homomorphic PRF, we can achieve a more efficient solution which enjoys parallelism. Let $F : \mathcal{X} \to \mathcal{Y}$ be a key homomorphic PRF, and let $(\mathsf{SymKeyGen}, \mathsf{E}, \mathsf{D})$ be a symmetric key CPA-secure encryption scheme with key space $\mathcal{K}$, where $(\mathcal{X}, +)$, $(\mathcal{Y}, +)$, and $(\mathcal{K}, +)$ are groups. The following is a secure updatable encryption scheme which can be used to encrypt message blocks. Let m_i be the i^{th} message block of a message m.

- $\mathsf{Setup}(1^\lambda)$ samples and outputs the public parameters pp used by F.
- $\mathsf{KeyGen}(1^\lambda)$ samples and outputs a secret key $\mathsf{sk} \leftarrow \mathsf{SymKeyGen}(1^\lambda)$.
- $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2, C = (C_1, C_2, i))$ chooses a random $x_2 \in \mathcal{X}$, sets $x_1 = \mathsf{D}(\mathsf{sk}_1, C_1)$, and returns the tuple $(\mathsf{E}(\mathsf{sk}_2, x_2), x_2 - x_1)$.
- $\mathsf{Enc}(\mathsf{sk}, m_i)$ chooses a random $x \in \mathcal{X}$ and outputs $C = (\mathsf{E}(\mathsf{sk}, x), m_i + F(x, i), i)$.
- $\mathsf{ReEnc}(\mathsf{rk}_{1,2,C} = (r, s), C = (C_1, C_2, i))$ outputs $(r, C_2 + F(s, i))$.
- $\mathsf{Dec}(\mathsf{sk}, C = (C_1, C_2, i))$ outputs $C_2 - G(\mathsf{D}(\mathsf{sk}, C_1), i)$.

Correctness and security follow straightforwardly as in the seed homomorphic PRG construction. We also note that we can replace the instances of the key homomorphic PRF with a γ-almost key

homomorphic PRF, and by modifying the scheme to round away accumulated error appropriately, the construction achieves T-time correctness for a pre-specified T in the setup algorithm, and the ciphertext size of the scheme grows logarithmically in γT, as before.

7.4 PRFs Secure Against Related-Key Attacks

Our last application shows that key-homomorphic PRFs can be used to construct PRFs secure against related key attacks. The key homomorphism is primarily used in the proof of security.

A Φ-RKA-PRF is a pseudorandom function secure against related-key attacks from a class Φ of related-key deriving (RKD) functions. Bellare and Cash [BC10] show how to obtain a Φ-RKA-PRF from a key malleable PRF with respect to class of RKD functions Φ, a key fingerprint (defined below), and a collision-resistant hash function. We paraphrase the definitions and the main theorem from [BC10], using a simplified definition of key fingerprints which suffices for our construction.

Key Malleability. Let $(\mathcal{K}, \oplus)$ and $(\mathcal{Y}, \otimes)$ be groups. A PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is key malleable with respect to a class Φ if there exists an efficient algorithm T with black-box access to an oracle $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$ such that:

- If $\mathcal{O}(x) = F(k, x)$, $\mathsf{T}^{\mathcal{O}(\cdot)}(\phi, x) = F(\phi(k), x)$ for all $\phi \in \Phi$ and $x \in \mathcal{X}$, and
- If $\mathcal{O}(x) = f(x)$ where $f : \mathcal{X} \to \mathcal{Y}$ is a random function, for $\phi_1, \ldots, \phi_q \in \Phi$ and distinct $x_1, \ldots, x_q \in \mathcal{X}$, the set of values $\{\mathsf{T}^{\mathcal{O}(\cdot)}(\phi_1, x_1), \ldots, \mathsf{T}^{\mathcal{O}(\cdot)}(\phi_q, x_q)\}$ is indistinguishable from a set of q uniform and independently sampled elements of $\mathcal{Y}$.

Key Fingerprints. A key fingerprint $w \in \mathcal{X}$ is such that $F(\phi(k), w) \neq F(\phi'(k), w)$ for all $k \in \mathcal{K}$ and all distinct $\phi, \phi' \in \Phi$.

Compatible (Collision-Resistant) Hash Function. A collision-resistant hash function $H : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$ is *compatible* with F and a key fingerprint w if $w \notin \mathrm{Im}(H)$.

Theorem 7.8 (cf. [BC10, Theorem 3.1])**.** *Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of functions and Φ be a class of RKD functions, where F is a key malleable PRF with respect to Φ. Let $w \in \mathcal{X}$ be a key fingerprint for F and Φ and let H be a compatible collision resistant hash function. Define $F_{\mathrm{RKA}} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ by*

$$F_{\mathrm{RKA}}(k, x) = F(k, H(x, F(k, w))).$$

Then, F_{RKA} is a Φ-RKA-PRF.

A Φ-RKA-PRF from the DLIN assumption. Let $\Phi_{\oplus} = \{\phi : \phi(k) = k \oplus k'\}_{k' \in \mathcal{K}}$, where $\oplus$ represents the group action over $\mathcal{K}$. In the case of F_{DLIN}, $\oplus$ is vector addition.

Lemma 7.9. *If F is key homomorphic, then F is key malleable with respect to $\Phi_{\oplus}$.*

Proof. Define $\mathsf{T}^{\mathcal{O}(\cdot)}(\phi, x) = \mathcal{O}(x) \otimes F(k', x)$, where ϕ is such that $\phi(k) = k \oplus k'$. Note that when $\mathcal{O}(x) = f(x)$ for a truly random function f, the values $f(x_1), \ldots, f(x_q)$ are uniform and independent so long as $x_1, \ldots, x_q$ are distinct. Hence, the values $\mathsf{T}^{\mathcal{O}(\cdot)}(\phi_1, x_1), \ldots, \mathsf{T}^{\mathcal{O}(\cdot)}(\phi_q, x_q)$ are also uniform and independent. If F is key homomorphic, then $\mathsf{T}^{\mathcal{O}(\cdot)}(\phi, x) = F(\phi(k), x)$ when $\mathcal{O}(x) = F(k, x)$. ∎

Lemma 7.10. *For any $x \in \mathcal{X}$, x is a key fingerprint for F_{DLIN} and the class $\Phi_{\oplus}$.*

Proof. Fix some $\mathbf{k} \in \mathcal{K}$ and consider two distinct related-key deriving functions $\phi, \phi' \in \Phi$. It follows that $\phi(\mathbf{k}) \neq \phi'(\mathbf{k})$. Fix an input $x = x_1 \cdots x_\ell \in \mathcal{X}$, and let $\mathbf{M} = \prod_{i=1}^{\ell} \mathbf{A}_{x_i}$. Recall from the definition of F_{DLIN} that $\mathbf{A}_0$ and $\mathbf{A}_1$ are full rank, and so $\mathbf{M}$ is also full rank. Therefore, $\mathbf{M} \cdot \phi(\mathbf{k}) \neq \mathbf{M} \cdot \phi'(\mathbf{k})$, and so $F_{\text{DLIN}}(\phi(\mathbf{k}), x) = (g_\ell)^{\mathbf{M} \cdot \phi(\mathbf{k})} \neq (g_\ell)^{\mathbf{M} \cdot \phi'(\mathbf{k})} = F_{\text{DLIN}}(\phi'(\mathbf{k}), x)$. Thus, x is a key fingerprint for F_{DLIN} and $\Phi_{\oplus}$. ∎

We consider the fingerprint $w = \mathbf{0} \in \mathbb{Z}_p^\ell$. For any collision resistant hash function $H : \mathbb{Z}_p^\ell \times (\mathbb{G}_\ell)^\ell \to \mathbb{Z}_p^{\ell-1}$, the hash function $H' : \mathbb{Z}_p^\ell \times (\mathbb{G}_\ell)^\ell \to \mathbb{Z}_p^\ell$ defined as

$$H'(x, y) = \begin{bmatrix} H(x, y) \\ 1 \end{bmatrix}$$

is compatible with F_{DLIN} and the key fingerprint w. This allows us to apply Theorem 7.8 with Lemmas 7.9 and 7.10 to get the following theorem.

Theorem 7.11. *Let H be a collision resistant hash function. Under the DLIN assumption, the function $F(\mathbf{k}, x) = F_{\text{DLIN}}(\mathbf{k}, H'(x, F_{\text{DLIN}}(\mathbf{k}, \mathbf{0})))$ is a Φ-RKA-PRF.*

8 Conclusions and Open Problems

We explored the concept of key-homomorphic PRFs and discussed its application to key rotation, one-round distributed PRFs, and symmetric-key proxy re-encryption. Our construction of lattice-based key-homomorphic PRFs in the standard model relies on a non-uniform variant of the learning with errors assumption that we show is equivalent to the standard LWE assumption.

We leave as an open problem the question of constructing key-homomorphic PRFs from other standard assumptions and in particular constructions using bilinear maps. Another interesting area of research is to construct key-homomorphic PRFs whose performance is comparable to real-world block ciphers such as AES. It would also be interesting to improve the tightness of the analysis in our lattice-based PRF, perhaps using techniques from [AKP+13].

Acknowledgments. We thank Zvika Brakerski, Daniele Miccancio, and Chris Peikert for helpful comments about this work. We also thank Gregory Roth for suggesting the application to re-keying in the cloud. This work was supported by NSF, the DARPA PROCEED program, an AFO SR MURI award, a grant from ONR, and by Samsung. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT '10*, volume 6110 of *LNCS*, pages 553–572, 2010.

[ABH09] G. Ateniese, K. Benson, and S. Hohenberger. Key-private proxy re-encryption. In *CT-RSA*, pages 279–294, 2009.

[ABV+12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, pages 280–297, 2012.

[ACP+09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *CRYPTO*, pages 417–432, 2002.

[AFG+06] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40, 2011.

[AKP+13] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *CRYPTO*, pages 57–74, 2013. Available at `http://eprint.iacr.org/2013/098`.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BC10] M. Bellare and D. Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO*, pages 666–684, 2010.

[BCF00] E. F. Brickell, G. D. Crescenzo, and Y. Frankel. Sharing block ciphers. In *ACISP*, volume 1841 of *LNCS*, page 457470, 2000.

[BF97] D. Boneh and M. Franklin. Efficient generation of shared rsa keys (extended abstract). In *CRYPTO*, pages 425–439, 1997.

[BHH+08] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology – CRYPTO '08*, pages 108–125, 2008.

[BK03] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT*, pages 491–506, 2003.

[BLM+13] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO'13*, pages 410–428, 2013.

[BLP+13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013.

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Eurocrypt*, pages 719–737, 2012.

[BPT12] M. Bellare, K. G. Paterson, and S. Thomson. Rka security beyond the linear barrier: Ibe, encryption and signatures. In *ASIACRYPT*, pages 331–348, 2012.

[BS03] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[CG99] R. Canetti and S. Goldwasser. An efficient *threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, pages 90–106, 1999.

[CH07] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *ACM Conference on Computer and Communications Security*, pages 185–194, 2007.

[CK05] D. L. Cook and A. D. Keromytis. Conversion and proxy functions for symmetric key ciphers. In *ITCC*, pages 662–667, 2005.

[CLT13] J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, pages 476–493, 2013.

[Cov12] A. Coviello. Open letter to rsa customers, 2012. `www.rsa.com/node.aspx?id=3872`.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[DF91] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (extended abstract). In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 457–469. Springer, 1991.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[Dod03] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, pages 1–17, 2003.

[DS02] P. D'Arco and D. Stinson. On unconditionally secure robust distributed key distribution centers. *ASIACRYPT 2002*, pages 181–189, 2002.

[DYY06] Y. Dodis, A. Yampolskiy, and M. Yung. Threshold and proactive pseudo-random permutations. In *TCC*, pages 542–560, 2006.

[FGM+97] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *FOCS'97*, pages 384–393, 1997.

[GGH13] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 34(4):792–807, 1986.

[GJK+99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, pages 295–310, 1999.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08*, pages 197–206. ACM, 2008.

[GRJ+07] R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of rsa functions. *J. Cryptology*, 20(3):393, 2007.

[HK07a] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[HK07b] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO'07*, pages 553–571, 2007.

[Kra01] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *CRYPTO*, pages 310–331, 2001.

[KSS12] B. Kreuter, A. Shelat, and C.-H. Shen. Billion-gate secure computation with malicious adversaries. In *Proceedings of the 21st USENIX conference on Security symposium*. USENIX Association, 2012.

[LV11] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Transactions on Information Theory*, 57(3):1786–1802, 2011.

[LW09] A. Lewko and B. Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *ACM CCS*, pages 112–120, 2009.

[MMP+11] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. TOTP: Time-based one-time password algorithm. RFC 6238, May 2011.

[MP13] D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. *IACR Cryptology ePrint Archive*, 2013.

[MS95] S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions, with applications to clipper-like escrow systems. In *CRYPTO'95*, pages 185–196, 1995.

[MSNW+05] K. M. Martin, R. Safavi-Naini, H. Wang, and P. R. Wild. Distributing the encryption and decryption of a block cipher. *Des. Codes Cryptography*, 36(3):263–287, 2005.

[Nie02] J. Nielsen. A threshold pseudorandom function construction and its applications. *CRYPTO '02*, pages 43–59, 2002.

[NPR99] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and kdcs. In *EUROCRYPT 99*, pages 327–346. Springer, 1999.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS'97*, pages 458–67, 1997.

[NS12] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. *SIAM Journal on Computing*, 41(4):772–814, 2012.

[Ped91] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526. Springer, 1991.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *41st Annual ACM Symposium on Theory of Computing — STOC '09*, pages 333–342. ACM, 2009.

[Pie12] K. Pietrzak. Subspace lwe. In *TCC*, pages 548–563, 2012.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05*, pages 84–93. ACM, 2005.

[SDF+94] A. D. Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *STOC'94*, page 522533, 1994.

[SG02] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha07a] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[Sha07b] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007:74, 2007.

[Sho00] V. Shoup. Practical threshold signatures. In *EUROCRYPT*, pages 207–220, 2000.

[SNS11] A. Syalim, T. Nishide, and K. Sakurai. Realizing proxy re-encryption in the symmetric world. In *ICIES '13*, volume 251, pages 259–274, 2011.

[SW13] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. `http://eprint.iacr.org/`.

A Additional Preliminaries

A.1 Pseudorandom Functions

We briefly review the definition of pseudorandom functions [GGM86]. 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 allow the PRF to take additionally public parameters pp and use $F_{\mathsf{pp}} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ to denote such a PRF.

Security for a PRF is defined using two experiments between a challenger and an adversary $\mathcal{A}$. For $b \in \{0, 1\}$ define the following experiment $\mathsf{Expt}_b^{\mathsf{PRF}}$:

1. Given security parameter λ, the challenger samples and publishes public parameters pp to the adversary. Next, if $b = 0$ the challenger chooses a random key $k \in \mathcal{K}$ and sets $f(\cdot) := F_{\mathsf{pp}}(k, \cdot)$. If $b = 1$ the challenger chooses a random function $f : \mathcal{X} \to \mathcal{Y}$.
2. The adversary (adaptively) sends input queries $x_1, \ldots, x_Q$ in $\mathcal{X}$ and receives back $f(x_1), \ldots, f(x_Q)$.
3. Eventually the adversary outputs a bit $b' \in \{0, 1\}$.

Let W_b denote the probability that $\mathcal{A}$ outputs 1 in experiment $\mathsf{Expt}_b^{\mathsf{PRF}}$.

Definition A.1 (Pseudorandom Function)**.** A PRF $F_{\mathsf{pp}} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is secure if for all efficient adversaries $\mathcal{A}$ the quantity

$$\mathbf{Adv}^{\mathsf{PRF}}[F, \mathcal{A}] := |W_0 - W_1|$$

is negligible.

A.2 Some Lemmas from Linear Algebra

Matrix norm. The norm of a matrix $\mathbf{R} \in \mathbb{R}^{k \times m}$ is defined as $\|\mathbf{R}\|_{\max} := \sup_{\|\mathbf{u}\|=1} \|\mathbf{R} \cdot \mathbf{u}\|$. We derive the following lemma along the lines of [ABB10, Lemma 16] which shows that a random matrix over $\{0, 1\}^{m \times m}$ has a matrix norm roughly m.

Lemma A.2. *Let $\mathbf{u} \in \mathbb{Z}^m$ be a vector and let $\mathbf{R}$ be a uniformly chosen matrix over $\{0, 1\}^{m \times m}$. Then,*

$$\Pr\left[\|\mathbf{Ru}\| > \frac{1}{2}\|\mathbf{u}\| \cdot \left(m + \sqrt{m} \cdot \omega(\sqrt{\log m})\right)\right] < \mathrm{negl}(m).$$

Proof. Observe that a uniformly chosen matrix $\mathbf{R}$ can be written as $\mathbf{R} = \frac{1}{2} \cdot \mathbf{1}^{m \times m} + \frac{1}{2} \cdot \mathbf{R}'$ where $\mathbf{1}$ is an all-ones matrix and $\mathbf{R}'$ is a matrix chosen uniformly from $\{-1, 1\}^{m \times m}$. Therefore, $\mathbf{Ru} = \frac{1}{2} \cdot \mathbf{1u} + \frac{1}{2} \cdot \mathbf{R}'\mathbf{u}$. A standard application of Hölder's inequality for power means implies that $\|\mathbf{1u}\| = \sqrt{m} \cdot \sum_{i=1}^{m} u_i \leq \sqrt{m} \cdot (\sqrt{m}\|\mathbf{u}\|)$. From [ABB10, Lemma 16], $\|\mathbf{R}'\mathbf{u}\|$ is within $\sqrt{m} \cdot \omega(\sqrt{\log m})$ with overwhelming probability (in m). Thus, from the triangle inequality, $\|\mathbf{Ru}\| \leq \frac{1}{2} \cdot \left(m + \sqrt{m} \cdot \omega(\sqrt{\log m})\right) \|\mathbf{u}\|$, as required. ∎

Lemma A.3. *Let m, and q be integers such that q is prime. Then the probability that a uniformly chosen matrix $\mathbf{A} \leftarrow \{0, 1\}^{m \times m}$ has $\mathbb{Z}_q$-rank m is at least* 0.288*.*

Proof. We start off with the following observation. For integer $k \in [m]$, let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ denote k linearly independent vectors in $\mathbb{Z}_q^m$. Let $V = \mathrm{span}_{\mathbb{Z}_q}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$. We claim that

$$|V \cap \{0, 1\}^m| \leq 2^k. \tag{A.1}$$

To see why, denote by $\mathbf{V}^*$ the $k \times m$ matrix over $\mathbb{Z}_q$ where the i-th row is the vector $\mathbf{v}_i$. As $\mathrm{Rk}(\mathbf{V}^*) = k$, without loss of generality, we may assume that columns 1 through k are linearly independent. This implies that for any vector $\mathbf{u} \in \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, $u_{k+1}, \ldots, u_m$ are *uniquely* determined by the values $u_1, \ldots, u_k$. This is also true of all vectors $\mathbf{u} \in \mathrm{span}_{\mathbb{Z}_q}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$. The proof of Equation (A.1)

follows in a straightforward manner now, by noting that there are at most 2^k possible values assigned to $u_1, \ldots, u_k$ for any $\mathbf{u} \in V \cap \{0,1\}^m$.

To prove the lemma, we look at $\mathbf{A}$ as m rows $\mathbf{a}_1, \ldots, \mathbf{a}_m$ sampled independently and uniformly from $\{0,1\}^m$. For $k \in [m-1]$, having chosen vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k$, the probability that $\mathbf{a}_{k+1}$ is independent of $\mathbf{a}_1, \ldots, \mathbf{a}_k$ is at least $(2^n - 2^k)/2^n$ from Equation (A.1). Therefore, the probability that $\mathbf{A}$ has rank m is:

$$\sum_{i=0}^{m-1} \Pr[\mathbf{a}_{i+1} \notin \mathrm{span}(\mathbf{a}_1, \ldots, \mathbf{a}_i)] \geq \sum_{i=0}^{m-1} \left(1 - \frac{1}{2^{m-i}}\right) > 0.288.$$

∎

Lemma A.4. *Let m, n, k, and q be integers such that $m \geq n$ and q is prime and let $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ be a full-rank matrix. Then, for uniform $\mathbf{S} \leftarrow \mathbb{Z}_q^{m \times k}$, $\mathbf{BS}$ is distributed uniformly over $\mathbb{Z}_q^{n \times k}$.*

Proof. Let $\mathbf{B}$ be viewed as $\mathbf{B} = [\mathbf{b}_1 | \cdots | \mathbf{b}_m]$ for column vectors $\mathbf{b}_i \in \mathbb{Z}_q^n$. As $\mathrm{Rk}(\mathbf{B}) = n$, there are n columns that are linearly independent. Let $\mathbf{B}^* \in \mathbb{Z}_q^{n \times n}$ denote the submatrix of these n linearly independent columns. Consider fixing $m - n$ rows of $\mathbf{S}$ corresponding to the remaining $m - n$ columns and only consider a $n \times m$ submatrix $\mathbf{S}^*$ that correspond to $\mathbf{B}^*$. The matrix $\mathbf{S}^*$ has column vectors $\mathbf{k}_1^*, \ldots, \mathbf{k}_k^* \in \mathbb{Z}_q^n$.

Then $\mathbf{BS} = [\mathbf{B}^*\mathbf{s}_1^* + \mathbf{u}_1 | \cdots | \mathbf{B}^*\mathbf{s}_k^* + \mathbf{u}_k]$ where $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are arbitrary vectors that depend on the values of the fixed rows. For any $i \in [k]$, as $\mathbf{B}^*$ is full-rank, there is a bijection between vectors $\mathbf{s}_i^*$ and $\mathbf{B}^*\mathbf{s}_i^*$. As $\mathbf{s}_i^*$ is distributed uniformly over $\mathbb{Z}_q^n$, so is $\mathbf{B}^*\mathbf{s}_i^*$ and $\mathbf{B}^*\mathbf{s}_i^* + \mathbf{u}_i$. This in turn implies that $\mathbf{BS}$ is distributed uniformly over all possible $n \times k$ matrices.

The above result holds true for every possible fixing of the $m - n$ rows corresponding to the columns not in $\mathbf{B}^*$ and therefore holds true for the uniform distribution over these values as well. ∎