Key Homomorphic PRFs and Their Applications

Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan

Computer Science Department,
Stanford University, Stanford, CA 94305

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. We also give a construction 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, 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 [28] 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.

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

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

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 410–428, 2013.

where $\lceil x \rfloor_p$ 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 [8] holds. Because rounding is not linear (i.e. it can happen that $\lceil a+b \rfloor_p \neq \lceil a \rfloor_p + \lceil b \rfloor_p$) the function F_{LWR} is not key homomorphic. However, it comes very close and is sufficiently homomorphic for most of our applications. In particular, F_{LWR} is "almost" key homomorphic in the sense that

$$F_{\text{LWR}}(\mathbf{k}_1 + \mathbf{k}_2, x) = F_{\text{LWR}}(\mathbf{k}_1, x) + F_{\text{LWR}}(\mathbf{k}_2, x) + e \tag{1.1}$$

where e is short; namely, $e \in [-1, 1]$.

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 [33]. 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

$$F_{\text{LWE}}(\mathbf{k}, x) = \left\lceil \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rfloor_p \in \mathbb{Z}_p^m \,. \tag{1.2}$$

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 $\mathbf{e} \in [-1, 1]^m$. Therefore this function is almost key homomorphic in the same sense as F_{LWR}, which is sufficient for most of our applications. We prove that F_{LWE} is a secure PRF based on the LWE assumption in the standard model.

The construction in Eq. (1.2) is closely related to an elegant non-key homomorphic PRF due to Banerjee, Peikert, and Rosen [8], but is technically quite different from it. The secret key in [8] 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 [8] 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 the full version, 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 [34, 23] 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 $x = x_1 \ldots x_\ell \in \{0,1\}^\ell$ is defined as

$$F_{\text{DLIN}}(\mathbf{k}, x) = (g_\ell)^{\mathbf{w}} \quad \in (\mathbb{G}_\ell)^\ell \qquad \text{where} \qquad \mathbf{w} = \mathbf{A}_{x_1} \cdots \mathbf{A}_{x_\ell} \cdot \mathbf{k} \quad \in \mathbb{Z}_p^\ell \quad (1.3)$$

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 the full version. 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 [29], but since the DDH assumption is false in groups with an ℓ-linear map, the relation is closer to the Lewko-Waters [25] 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 (1.3) the secret key is only a single vector in $\mathbf{k} \in \mathbb{Z}_p^\ell$ and this enables the key-homomorphic property. However, our construction inherently requires an ℓ-linear map in $\mathbb{G}$ whereas [25] did not. Unfortunately, we cannot use the ℓ-linear map candidate of Garg, Gentry, and Halevi [20] to instantiate the construction because the ℓ-decision linear problem is easy for that proposal. We therefore view our lattice-based construction as our primary key homomorphic PRF and await other ℓ-linear map candidates to instantiate our second scheme.

1.2 Key Homomorphic PRFs: Applications

Our interest in key homomorphic PRFs stems from a number of real-world applications for such functions. We describe them briefly here and discuss some of these applications in more detail in Section 6. Due to space constraints the remaining applications have been deferred to the full version.

Distributed PRFs. In a one-time password system 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 computation of the PRF value using the same PRF secret key. Since the server knows the secret PRF keys for all users, these authentication servers have become a prime target for attacks [17]. In response, RSA introduced *Distributed Credential Protection* where PRF keys are shared among two or more key servers and all 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.

Key homomorphic PRFs give a clean solution to distributing PRF keys using a simple communication pattern: 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 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 "multiplies" the responses from the key servers by the appropriate Lagrange coefficients and then "adds" the results using the key homomorphism property. We give the details in Section 6.1. 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 only the high order bits of the computed value $F(k, x)$ so as to eliminate the homomorphism error.

Symmetric-Key Proxy Re-encryption. Key homomorphic PRFs provide the symmetric-key analogue of public-key proxy re-encryption [12, 7, 15, 6, 26]. 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 6.

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.

We note that basic symmetric-key proxy re-encryption can also be done using a seed-homomorphic pseudorandom generator (PRG)—that is, 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 3.1. However, encrypting with a PRG is only one-time secure, whereas randomized counter-mode using a PRF provides security against chosen-plaintext attacks, thereby enabling a single key to encrypt multiple messages. We also

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., [24]).

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).

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 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.

We note that key-rotation in the cloud can potentially be done using ad-hoc solutions such as nested encryption, but these solutions 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 would provide 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 [9] 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 $\Phi_\oplus = \{\phi(k) = k \oplus k'\}_{k' \in \mathcal{K}}$, 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 the full version 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 [9].

1.3 Related Work

Distributed PRFs were first considered by Naor and Pinkas [28] for the purpose of distributing the Kerberos key distribution center. They proposed the simple random oracle construction F_{DDH} which is key homomorphic. Nielsen [30] and D'Arco and Stinson [18] constructed robust distributed PRFs, but evaluation requires interaction between the key servers and multiple rounds of communication. Dodis [19] constructs distributed PRFs and VRFs under a strong assumption and also requires multiple rounds of communication to evaluate the PRF.

Much recent work has focused on preventing related key attacks [10, 9, 11]. Key homomorphic PRFs are on the other end of the spectrum where key homomorphism is encouraged in support of specific applications.

Syalim, Nishide, and Sakurai [37] describe a symmetric proxy re-encryption scheme based on the all or nothing transform (AONT) in the random oracle model. Cook and Keromytis [16] propose to use double encryption to provide one-hop symmetric proxy re-encryption.

2 Preliminaries

Rounding. We define $\lceil \cdot \rfloor$ to round a real number to the nearest integer. For integers q and p where $q \geq p \geq 2$, we define the function $\lceil \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lceil x \rfloor_p = \lceil (p/q) \cdot \bar{x} \rfloor \pmod p$ where $\bar{x} \in \{0, 1, \ldots, q-1\}$ is the integer congruent to $x \pmod q$. For a vector $\mathbf{v} \in \mathbb{Z}_q^m$, we define $\lceil \mathbf{v} \rfloor_p$ as the vector in $\mathbb{Z}_p^m$ obtained by rounding each coordinate of the vector individually. A probability distribution χ over $\mathbb{R}$ is said to be *B-bounded* if it holds that $\Pr_{x \leftarrow \chi}[|x| > B]$ is negligible in the security parameter.

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) [22] 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 $f(\cdot)$. In this paper, we allow our PRFs and PRGs to be further paramaterized by an additional public parameter pp, which will be clear from context. We let $\mathbf{Adv}^{\mathsf{PRF}}[F, \mathcal{A}]$ denote the advantage of adversary $\mathcal{A}$ in distinguishing the PRF F (along with its public parameters) from a random function $f : \mathcal{X} \to \mathcal{Y}$.

Learning With Errors (LWE) Assumption. The LWE problem was introduced by Regev [33] 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:*

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

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 [33] shows that for a certain noise distribution $\chi = \overline{\Psi}_\alpha$[1], for n polynomial in λ and $q > 2\sqrt{n}/\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction (see also [31, 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 [5]. Similarly, the noise distribution $\boldsymbol{\chi}$ can be a simple low-norm distribution [27].

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.

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 [28]. 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* $\mathbf{v} \in [-\gamma, \gamma]^m$ *such that*
$$F(k_1, x) + F(k_2, x) = F(k_1 \oplus k_2, x) + \mathbf{v} \pmod{p}.$$

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.

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

3.1 Seed Homomorphic Pseudorandom Generators

To explain our PRF construction 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 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. Let $\mathbb{G}$ be a group of order p in which the DDH assumption holds. Consider the following PRG $G_{\text{DDH}} : \mathbb{Z}_p \to \mathbb{G} \times \mathbb{G}$ and public parameters pp being a pair $g, h \xleftarrow{R} \mathbb{G}$:

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

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.

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 $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, is secure assuming the Learning With Rounding (LWR) problem is hard for the given parameters p, q, n, m:

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

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 $\mathbf{e} \in [-1, 1]^m$. An almost seed homomorphic PRG can be used for updatable encryption by appending a few zeros to each plaintext block before encryption so that low-order bits can be dropped during decryption thereby eliminating any errors resulting from the *almost* homomorphic property.

Key Homomorphic PRFs from the GGM Construction. 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 [22] with key space $\mathcal{X}$ and input space $\{0,1\}^\ell$ as follows:

$$F_{\text{GGM}}(k,\ x = x_1 \cdots x_\ell) = G_{x_\ell}\big(\, G_{x_{\ell-1}}\big(\cdots G_{x_2}(\, G_{x_1}(k)\,)\cdots\big)\big) \tag{3.1}$$

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 above cannot be used directly in construction (3.1). The problem is that these generators do not compose as needed for construction (3.1). 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.

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 [33] 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:*

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

where $m = poly(k)$, $\mathbf{A} \leftarrow \boldsymbol{\eta}^m$ (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 $(\mathbf{B},\ \mathbf{v}')$. It works 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 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 [14] and also in [2, 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 [4, 21]:

- 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 $q = q(k)$ be an integer 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 $(\mathbb{Z}_q, \lfloor k/(6 \log_2 q) \rfloor, \chi)$-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 $\mathbf{T} = (\mathbf{M}, \mathbf{T}_{\mathbf{M}})$.
- $\mathsf{SamplePre}(\mathbf{z} \in \mathbb{Z}_q^n,\ \mathbf{T})$ outputs $\mathsf{SampleD}(\mathbf{M}, \mathbf{T}_{\mathbf{M}}, \mathbf{z}, \sigma)$.

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 [32] 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 $q = q(k)$ be an integer and let V be a linear subspace of $\mathbb{Z}_q^k$ of dimension at least n. 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 Sketch. By Theorem 4.3 it suffices to show that $\boldsymbol{\eta}_V$ is coset sampleable and this follows by elementary linear algebra. □

5 An LWE-Based almost Key Homomorphic PRF

In this section, we construct a 1-almost key homomorphic PRF in the standard model based on the LWE assumption. Our new (almost key homomorphic) PRF has parameters comparable to those of [8]. Despite being *almost* key homomorphic, our PRF can still be used for the applications discussed in the introduction.

Construction. Let q, n, and m be integers such that $m = n\lceil \log q \rceil$. Recall the definition of the rounding function $\lceil \cdot \rfloor_p$ and the noise distribution $\overline{\Psi}_\alpha$ from Section 2, and the definition of $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ from Lemma 4.5. Let public parameters 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. The secret key $\mathbf{k}$ is a vector in $\mathbb{Z}_q^m$. Define $F_{\mathrm{LWE}} : \mathbb{Z}_q^m \times \{0,1\}^\ell \to \mathbb{Z}_p^m$ as follows:

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

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

The parameters α, m, ℓ, and p must be chosen carefully, as α determines the choice of the underlying lattice hardness assumption used for security. In the interest of space, we provide a proof overview of the pseudorandomness of F_{LWE} and leave the rest of the details to the full version.

Proof Overview of Theorem 5.1. In proving the pseudorandomness of F_{LWE}, we follow the outline of the standard GGM construction [22]. 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 all bits of x are ignored, and instead of the product of $\mathbf{A}_{x_i}$'s, the experiment returns random values. 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 $j \in [\ell+1]$, which is shown as follows.

Let $\mathbf{P} = \prod_{i=1}^{j-1} \mathbf{A}_{x_i}$. When the adversary queries its PRF oracle at a point $x \in \{0,1\}^\ell$, the resulting PRF evaluation given to the adversary in Expt_j and Expt_{j+1} looks like $\lceil \mathbf{P} \cdot \mathbf{r} \rfloor_p$ and $\left\lceil \mathbf{P}\mathbf{A}_{x_j} \cdot \mathbf{r} \right\rfloor_p$, where $\mathbf{r}$ is chosen uniformly in $\mathbb{Z}_q^m$ and is kept consistent across the adversary's queries using a lookup table indexed by the 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 where the evaluations given to the adversary look like $\left\lceil \mathbf{P}\mathbf{A}_{x_j}\mathbf{s} + \mathbf{P}\boldsymbol{\delta} \right\rfloor_p = \left\lceil \mathbf{P}(\mathbf{A}_{x_j}\mathbf{s} + \boldsymbol{\delta}) \right\rfloor_p$, where $\mathbf{s}$ and $\boldsymbol{\delta}$ are kept consistent across the adversary's queries as was done previously. Now, it remains to show that

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

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. Although $\boldsymbol{\delta}$ is low-norm, if $\mathbf{P}$ were distributed uniformly, as is the usual case, this argument would fail. This explains why we must sample $\mathbf{A}_0$ and $\mathbf{A}_1$ using $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$—it ensures that $\mathbf{P}$ 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 such a challenge, the matrix $\mathbf{A}$ is low-norm, which is precisely modeled by the non-uniform learning with errors problem, introduced in Section 4. From Theorem 4.3, we can show computational indistinguishability in (b) under the *standard* LWE assumption. □

With a simple argument about the rounding of values from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ (for which the details are left to the full version), for every $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{Z}_q^m$ and every $x \in \{0,1\}^\ell$, there exists an $\mathbf{e} \in [-1, 1]^m$ such that $F_{\text{LWE}}(\mathbf{k}_1, x) + F_{\text{LWE}}(\mathbf{k}_2, x) = F_{\text{LWE}}(\mathbf{k}_1 + \mathbf{k}_2, x) + \mathbf{e}$. Thus, we can state the following theorem.

Theorem 5.2. *The tuple $(F_{\text{LWE}}, +)$ is a 1-almost key homomorphic PRF, where $+$ is addition over $\mathbb{Z}_q^m$.*

6 Applications of (Almost) Key Homomorphic PRFs

In this section, we construct one-round distributed PRFs [28] and symmetric proxy re-encryption schemes from γ-almost key homomorphic PRFs.

6.1 Distributed PRFs

Definition. To define distributed PRFs we follow the exposition of Naor, Pinkas, and Reingold [28]. 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 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. 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 by providing a concrete experiment-based security definition in the full version.

Distributed PRFs from Key Homomorphic PRFs. Let $F : \mathbb{F} \times \mathcal{X} \to \mathbb{F}$ be a key homomorphic PRF where $\mathbb{F}$ is a field. We consider a (t, N)-threshold secret sharing scheme [35] over a secret k_0 in $\mathbb{F}$, which constructs key-shares by sampling a uniformly random polynomial $p(z) \in \mathbb{F}[Z]$ of degree $t-1$ such that $p(0) = k_0$, and the remaining coefficients are sampled uniformly at random from $\mathbb{F}$. 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 $\Lambda_{i,W} \in \mathbb{F}$ are the Lagrange coefficients that depend only on W. 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 $p(z)$ of degree $t-1$ such that $p(0) = k_0$ and outputs $(p(1), \ldots, p(N))$.

- $\diamond$ $\mathsf{F}(k_i, x)$ returns the output of the key homomorphic PRF $F(k_i, x)$.
- $\diamond$ $\mathsf{G}(W, y_1, \ldots, y_t)$ computes and returns $\sum_{i \in W} \Lambda_{i,W} \cdot y_i$.
- $\diamond$ $\mathsf{f}(k_0, x)$ returns the output of $F(k_0, x)$.

Let pp be the output of $\mathsf{Setup}(1^\lambda)$, k_0 be chosen uniformly from $\mathbb{F}$, $(k_1, \ldots, k_n)$ 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 $\mathsf{G}\left(W, \mathsf{F}(k_{i_1}, x), \ldots, \mathsf{F}(k_{i_t}, x)\right) = \sum_{i \in W} \Lambda_{i,W} \cdot F(k_i, x)$. By the key homomorphism property of F, this quantity is equal to $\sum_{i \in W} F\left(\Lambda_{i,W} \cdot k_i, x\right) = \mathsf{F}(k_0, x)$ as required. If F is a key homomorphic pseudorandom function, then the following theorem shows that Π_{dPRF} is a secure distributed PRF. The complete proof is given in the full version.

Theorem 6.1. *If F is a key homomorphic PRF, then Π_{dPRF} is a secure distributed PRF.*

Constructing dPRFs 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. Unfortunately, when Lagrange coefficients are interpreted as elements in $\mathbb{Z}_p$, they can be arbitrarily large which breaks correctness of the combine algorithm. To overcome this difficulty, we use the technique of "clearing the denominator" [36, 1]. Note that for every Lagrange coefficient $\Lambda_{i,W}$, it holds that $N! \cdot \Lambda_{i,W} \in \mathbb{Z}$, and so the combine algorithm now uses $N! \cdot \Lambda_{i,W} \in \mathbb{Z}$ to reconstruct the output of the PRF. The complete details of the construction and its security proof are included in the full version.

6.2 Symmetric Proxy Re-encryption

Another natural application of key homomorphic PRFs is in constructing 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. 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.

- $\diamond$ $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$. On input the security parameter λ, the setup algorithm Setup outputs the public parameters pp used for the scheme.
- $\diamond$ $\mathsf{KeyGen}(1^\lambda) \rightarrow \mathsf{sk}$. On input the security parameter λ, the key generation algorithm KeyGen outputs a secret key sk.
- $\diamond$ $\mathsf{ReKeyGen}(\mathsf{sk}_1, \mathsf{sk}_2) \rightarrow \mathsf{rk}_{1,2}$. On input two secret keys sk_1 and sk_2, the re-encryption key generation algorithm ReKeyGen outputs a bidirectional re-encryption key $\mathsf{rk}_{1,2}$.
- $\diamond$ $\mathsf{Enc}(\mathsf{sk}, m) \rightarrow C$. On input a secret key sk and message m, the encryption algorithm Enc outputs a ciphertext C.
- $\diamond$ $\mathsf{ReEnc}(\mathsf{rk}_{1,2}, C_1) \rightarrow C_2$. 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 $\perp$.
- $\diamond$ $\mathsf{Dec}(\mathsf{sk}, C) \rightarrow m$. 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 security model defines the notion of semantic security for *symmetric* proxy re-encryption. We adapt the public-key model of Canetti and Hohenberger [15] to the symmetric-key settings by providing access to an additional encryption oracle. The model is described in detail in the full version.

Symmetric Proxy Re-encryption from Key Homomorphic PRFs. We show how to achieve a symmetric proxy re-encryption scheme with T-time correctness for all $T > 1$ under our security model using key homomorphic PRFs. Let $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ be a key homomorphic PRF with superpolynomial input set size ($|\mathcal{X}| = \omega(\mathrm{poly}(\lambda))$). 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 $C - F(\mathsf{sk}, r)$.

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. The complete proof is given in the full version.

Theorem 6.2. *If F is a key homomorphic PRF, then Π_{proxy} is a secure symmetric proxy re-encryption scheme.*

Using almost Key Homomorphic PRFs. In the full version, we show how to construct a symmetric proxy re-encryption scheme with T-time correctness (where T must be an additional input to Setup) from almost key homomorphic PRFs. The proof of correctness is straightforward and the proof of security remains the same.

7 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 be interesting to improve the tightness of the analysis in our lattice-based PRF, perhaps using techniques from [3]. Another useful improvement would be to reduce the hardness of worst-case lattice problems directly to our non-uniform LWE variant to improve its hardness parameters.

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. Distrib. Statement "A": Approved for Public Release, Distribution Unlimited.

References

[1] Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H.: Functional encryption for threshold functions (or fuzzy IBE) from lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 280–297. Springer, Heidelberg (2012)

[2] Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011)

[3] Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited: New reduction, properties and applications. IACR Cryptology ePrint Archive, 2013:98 (2013)

[4] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS, pp. 75–86 (2009)

[5] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)

[6] Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009)

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

[8] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)

[9] Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010)

[10] Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)

[11] Bellare, M., Paterson, K.G., Thomson, S.: RKA security beyond the linear barrier: IBE, encryption and signatures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 331–348. Springer, Heidelberg (2012)
[12] Blaze, M., Bleumer, G., Strauss, M.J.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
[13] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: STOC, pp. 575–584 (2013)
[14] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
[15] Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: ACM Conference on Computer and Communications Security, pp. 185–194 (2007)
[16] Cook, D.L., Keromytis, A.D.: Conversion and proxy functions for symmetric key ciphers. In: ITCC (1), pp. 662–667 (2005)
[17] Coviello, A.: Open letter to rsa customers (2012), `http://www.rsa.com/node.aspx?id=3872`
[18] D'Arco, P., Stinson, D.R.: On unconditionally secure robust distributed key distribution centers. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 346–363. Springer, Heidelberg (2002)
[19] Dodis, Y.: Efficient construction of (Distributed) verifiable random functions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 1–17. Springer, Heidelberg (2002)
[20] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610 (2012)
[21] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008, pp. 197–206. ACM (2008)
[22] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 34(4), 792–807 (1986)
[23] Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
[24] Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
[25] Lewko, A., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: ACM CCS, pp. 112–120 (2009)
[26] Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. IEEE Transactions on Information Theory 57(3), 1786–1802 (2011)
[27] Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. IACR Cryptology ePrint Archive (2013)
[28] Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and kDCs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 327–346. Springer, Heidelberg (1999)
[29] Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: FOCS 1997, pp. 458–467 (1997)
[30] Nielsen, J.B.: A threshold pseudorandom function construction and its applications. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 401–416. Springer, Heidelberg (2002)
[31] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: 41st Annual ACM Symposium on Theory of Computing (STOC 2009), pp. 333–342. ACM (2009)

[32] Pietrzak, K.: Subspace LWE. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 548–563. Springer, Heidelberg (2012)
[33] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005, pp. 84–93. ACM (2005)
[34] Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. IACR Cryptology ePrint Archive, 2007:74 (2007)
[35] Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
[36] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
[37] Syalim, A., Nishide, T., Sakurai, K.: Realizing proxy re-encryption in the symmetric world. In: Abd Manaf, A., Zeki, A., Zamani, M., Chuprat, S., El-Qawasmeh, E. (eds.) ICIEIS 2011, Part I. CCIS, vol. 251, pp. 259–274. Springer, Heidelberg (2011)