A Low-Round Distributed PRF from Lattices and its Application to Distributed Key Management

Matthias Geihs[1] and Hart Montgomery[2]

[1]Torus Labs
[2]Linux Foundation

Abstract

We initiate the study of lattice-based pseudo-random functions (PRFs) for use in multi-party computation protocols, motivated by their application to distributed key management. We show that the LWE-based PRF of Boneh et al. (CRYPTO'13) can be turned into a distributed PRF protocol that runs in only 8 online rounds, improving over the state-of-the-art by an order of magnitude. The resulting protocol can be used as a method for distributed key derivation and reduces the amount of managed key material in distributed key management systems from linear in the number of users to constant. Finally, we support our findings by implementing and evaluating our protocol using the MP-SPDZ framework (CCS'20).

1 Introduction

Distributed pseudo-random functions (PRFs) are an important building block in multi-party computation (MPC) protocols. We refer to [GRR+16, GØS+23] for a list of applications. Here we focus on their application to distributed key management and, in particular, distributed key derivation, which we will explain shortly.

1.1 Distributed key management

A distributed key management system [Web23, Dfn23, Lit23] is a cryptographic key management system run by a set of servers such that no individual server has access to any of the managed keys, but some subset of k servers can run a protocol to determine any key distributedly. This is typically achieved through the use of multi-party computation (MPC) protocols, whose security relies on the assumption that a certain fraction of the involved parties is honest and follows the protocol.

At the core of distributed key management typically is a distributed key generation (DKG) protocol that is used by the servers to generate new user secret keys. Each key is afterwards stored in *secret-shared form* [Sha79], which means that each server holds only a share of each key. Individual key shares reveal absolutely nothing about the user keys, unless they are combined with sufficiently many other correlated key shares.

The user can access its key by authenticating against the servers, downloading the key shares, and reconstructing the key locally. Alternatively, it can instruct the servers to run a multi-party computation protocol with input the secret-shared key, for example, in order to sign a message without ever reconstructing the key in the clear.

1.2 Key refresh and challenges to scalability

The security of a distributed key management system as described above relies on the assumption that a certain fraction of the servers are honest and keep their stored key shares hidden from everyone else.

To maintain security against gradual corruption, the servers are required to regularly run a share refresh protocol [HJK$^+$95] that updates all key shares such that old shares become useless and cannot be combined with new shares to reconstruct the key. A similar protocol is run whenever the set of key management servers changes. However, the workload for refreshing the key share database *grows linearly with the number of keys managed by the system.*

One solution to reducing the workload for share refresh is to use a *distributed key derivation* (DKD) protocol instead of a DKG for generating the user key. In contrast to a DKG, which produces a fresh and uncorrelated key on every run of the protocol, a DKD allows the servers to deterministically derive a user key from a secret-shared master key and the user's identity on demand. This means that user key shares need not be stored by the servers and the cost for share refresh is reduced from linear in the number of user keys to constant, because only the secret-shared master key needs to be maintained and refreshed. Such a DKD protocol is essentially realized by a suitable distributed PRF protocol, as we will see in the next section.

1.3 Suitable distributed PRFs

A pseudo-random function (PRF) is a keyed function $F_k : A \to B$ that is indistinguishable from a randomly chosen function with the same domain and range, with respect to a computationally-bounded observer who does not know the secret key. If the range of the PRF is equal to the key space of the targeted cryptographic scheme (e.g., an integer prime field in the case of the ECDSA signature scheme), then the PRF can be used as a key derivation function to derive a large number of user keys from a single master key and the corresponding user identities. That is, given a master PRF key k and a user identity u, the derived user key is $F_k(i)$. Note that while this user key is completely determined by k and u, yet, it appears completely random to anyone who does not know the master key k.

By a *distributed PRF* we mean an interactive protocol that is run between a set of parties and allows them to evaluate a PRF without knowledge of the secret key. For our distributed key derivation application, we are looking for a distributed PRF that satisfies the following properties:

User Identifiers $\subseteq$ Input(PRF). The PRF input space must contain all user identifiers, because the user keys will be derived from them.

Output(PRF) = KeySpace: The PRF output distribution should be indistinguishable from the key distribution of the targeted cryptographic scheme. Specifically, in the case of ECDSA, we are looking for a uniform distribution over a prime field.

Secret-shared output: The output of the distributed PRF should be in secret-shared form so that it remains hidden from the servers and can be fed into subsequent multi-party computation protocols such as a multi-party signature generation protocol [DKL+19, KG21].

Robust to faulty or malicious servers: The protocol must be robust and secure against a subset of faulty or malicious servers.

Well-founded assumptions: The security of the protocol should rely on well-founded security assumptions that are reliable in real world applications.

Low round complexity: The protocol should require only a small number of interactions between the servers in order to be deployable to settings where the servers reside in different locations and the network latency between them is relatively high.

Why do we focus on round complexity? We choose to focus on measuring performance in terms of round complexity because we found this to be a major bottleneck with respect to existing candidate protocols and our application. Moreover, while communication and computational complexity can be mitigated by upgrading the server infrastructure, round complexity can only be mitigated by moving the servers closer to each other, which is far more costly, or may not be possible at all (e.g., for regulatory reasons).

1.4 State of the art

[GØS$^+$23] give an overview of the state of the art of MPC-friendly PRF constructions and we mostly summarize their observations here.

Traditional PRFs (e.g., AES [DBN$^+$01], SHA-3 [Dwo15]) are not designed for use in MPC protocols. They typically have a higher multiplicative depth than specialized constructions and work over bytes instead of prime fields.

The Naor-Reingold construction [NR97] yields an efficient distributed PRF protocol but has a public output and is therefore not suitable for our use case. [GRR$^+$16] describe an efficient distributed PRF protocol based on the hardness of the Shifted Legendre Symbol Problem, which, however, is a rather unconvential and seemingly less reliable assumption [BBU$^+$19].

LowMC [ARS$^+$15], MiMC [AGR$^+$16], GMiMC [AGP$^+$19], HadesMiMC [GLR$^+$20], and Rescue [AABS$^+$20] are invertible block ciphers. However, invertibility is not required for our application, and a lower multiplicative complexity may be achieved by working with non-invertible functions [DGG$^+$21].

Farfalle [BDH$^+$17] is an efficiently parallelizable permutation-based PRF construction with arbitrary input and output length. Ciminion [DGG$^+$21] is a modified version of Farfalle based on a Feistel scheme. However, both require the expensive computation of a key schedule. Finally, Hydra [GØS$^+$23] is a recent MPC-friendly PRF construction with comparable performance to Ciminion but without the need for a key schedule. Both compare favorably to the state of the art in terms of multiplicative complexity, but still require on the order of 100 online communication rounds per PRF evaluation, which incurs a significant latency overhead in our application.

Another interesting candidate for MPC-friendly PRFs is the "dark matter" PRF construction family [BIP$^+$18]. We note that this has been used to construct oblivious PRFs [ADD$^+$23] but has the drawback that it uses a non-standard assumption and has only been analyzed in the semi-honest 3-party setting.

1.5 Contribution

We show that the lattice-based PRF construction of Boneh et al. [BLM$^+$13] can be turned into a distributed PRF protocol that fulfills all our requirements (cf. Section 1.3) and runs in only $2 + \log_2^2(q) + \log_2^2(q/p)$ online rounds, where q and p are lattice parameters. For concrete parameter choices, this results in a distributed PRF protocol running in only 8 online rounds, improving over the state-of-the-art by an order of magnitude. We support our findings by implementing and evaluating our protocol using the MP-SPDZ framework of Keller [Kel20].

1.6 Related work

[MK23] also addresses the problem of scalable distributed key management via on-demand distributed key derivation. In comparison to our approach, they use the lattice-based PRF construction from [BLM$^+$13] as an almost key-homomorphic PRF, which leads to a non-interactive approximate distributed PRF protocol. Due to the PRF not being exactly key homomorphic, they have to account for a small error in the key derivation. This unfortunately makes the protocol rather complex to implement as it requires the use of particular secret sharing schemes and zero-knowledge proofs to maintain protocol efficiency, correctness, and security. In contrast, our approach does not impose any additional requirements on the used secret sharing scheme and can be directly paired with subsequent MPC protocols without the need for costly post-processing of the generated key shares.

1.7 Organization

Section 2 introduces notation, basic primitives, and the lattice-based PRF construction of [BLM$^+$13]. Section 3 describes how we optimize the construction for the MPC setting. Section 4 describes how we implement the construction as an MPC protocol and evaluates its performance.

2 Preliminaries

2.1 Notation

We first introduce some basic notation. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \dots, n\}$. 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 $(x_1, \dots, x_{|\mathbf{x}|})$. 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 in some (security) parameter λ. 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.

Distributions. By $\boldsymbol{\eta}_{\mathsf{Bin}(k)}$ we denote the uniform distribution on $\{0,1\}^k$. 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 $\alpha/\sqrt{2\pi}$. For two probability distributions X and Y over a finite domain D, we define their statistical distance as

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in D} |\Pr[X = a] - \Pr[Y = a]| \ .$$

We denote the uniform distribution over a finite domain D by $U(D)$. The following theorem describes a bound on the statistical distance between $U(\mathbb{Z}_m)$ and $U(\mathbb{Z}_n) \bmod m$, for $m, n \in \mathbb{N}$.

Theorem 2.1. *For $m, n \in N$, $\Delta(U(\mathbb{Z}_m), U(\mathbb{Z}_n) \bmod m) \leq m/n$.*

A similar bound is also used in [GRR+16], but left without a proof. For completeness, we include a proof of the theorem in Appendix A.

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.

2.2 Pseudorandom Functions

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 $f(\cdot)$. In this paper, we allow our PRFs to be further parameterized by a public parameter pp. When needed, this pp is generated by a Setup algorithm.

Security for a PRF is defined using an experiment 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 2.2 (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.

2.3 Lattice Preliminaries

Learning with errors. The *learning with errors* (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 2.3 (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:

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

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].

Learning with rounding. Banerjee, Peikert, and Rosen [BPR12] consider a related problem, denoted the "learning with *rounding*" (LWR) problem. (Recollect the notation $\lfloor \cdot \rfloor_p$ from Section 2.1.)

Definition 2.4 (Learning With Rounding)**.** For integers $q = q(\lambda)$ and $p = p(\lambda)$ such that $q > p \geq 2$ the learning with rounding problem $(\mathbb{Z}_q, n, p)$-LWR is to distinguish between the following pairs of distributions:

$$\{\mathbf{A}\mathbf{s}, \lfloor \mathbf{A}\mathbf{s} \rfloor_p\} \qquad \text{and} \qquad \{\mathbf{A}, \lfloor \mathbf{u} \rfloor_p\},$$

where $m = \text{poly}(n)$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.

Banerjee et al. show that for any B-bounded distribution χ over $\mathbb{Z}$ and $q \geq pBn^{\omega(1)}$, the $(\mathbb{Z}_q, n, p)$-LWR problem is at least as hard as solving the $(\mathbb{Z}_q, n, \chi)$-LWE problem.

Definition 2.5. 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 $\lfloor q\, X \rceil \mod q$ where X is a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\,\pi}$.

Theorem 2.6 (cf. [Reg05])**.** *If there exists an efficient, possibly quantum, algorithm for deciding the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)$-LWE problem for $q > 2\sqrt{n}/\alpha$ then there is an efficient quantum algorithm for approximating the SIVP and GapSVP problems, to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2 norm, in the worst case.*

2.4 Lattice-Based PRF from [BLM+13]

We restate the LWE-based PRF from [BLM+13], Section 5. It will serve as the basis for our MPC-friendly PRF construction.

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 $\lfloor \cdot \rceil_p$ and the definition of $\boldsymbol{\eta}_{\mathsf{Bin}(m)}$ from Section 2.1, and the standard LWE noise distribution $\overline{\Psi}_\alpha$.

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. 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:

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

Security. [BLM+13], Theorem 5.1 establishes the security of F_{LWE} under the LWE assumption. We restate the theorem here for reference.

Theorem 2.7. *The function F_{LWE} is pseudorandom under the $(\mathbb{Z}_q, n, \overline{\Psi}_\alpha)$-*LWE *assumption for parameter choices satisfying $\alpha \cdot m^\ell \cdot p \leq 2^{-\omega(\log n)}$.*

3 Optimizing for MPC

In the following, we describe how we optimize the lattice-based PRF construction of [BLM+13] (cf. Section 2.4) for the MPC setting, with a particular focus on the application to distributed key derivation.

3.1 Using A Random Oracle

For practical purposes, it is generally accepted to rely on the random oracle model (ROM) [BR93]. Relying on this model, we can replace the product $\prod_{i=1}^{\ell} \mathbf{A}_{x_i}$ by a call to a hash function $H(x)$ modeled as a random oracle. This leads to the following PRF construction, which we essentially consider a "folklore" construction.

Construction. Let q, p, l, m, and n be integers such that $m = n\lceil \log q \rceil$ and p divides q. Let $H : \{0,1\}^* \to \mathbb{Z}_q^{l\times m}$ be a cryptographic hash function that gets as input a binary string of arbitrary length and outputs a matrix in $\mathbb{Z}_q^{l\times m}$. The secret key $\mathbf{k}$ is a vector in $\mathbb{Z}_q^m$. Define $F_{\text{RO}} : \mathbb{Z}_q^m \times \{0,1\}^* \to \mathbb{Z}_p^l$ as follows:

$$F_{\text{RO}}(\mathbf{k}, x) = \lfloor H(x) \cdot \mathbf{k} \rfloor_p .$$

Security. The following theorem establishes the security of F_{RO} under the LWE assumption in the random oracle model. We note that this follows almost immediately from the proof of hardness of learning with rounding in [BPR12], Theorem 3.2.

Theorem 3.1. *Let n, p, and q be integers, and let χ be any B-bounded distribution over $\mathbb{Z}$ such that, for security parameter λ, $q = q(\lambda)$, $p = p(\lambda)$, $q > p \geq 2$, and $q \geq pBn^{\omega(1)}$. Any adversary that can win the PRF security game as defined in Definition 2.2 with non-negligible advantage ε can be used to solve the $(\mathbb{Z}_q, n, p)$-LWR problem with advantage ε.*

Proof. Suppose we are given access to a $(\mathbb{Z}_q, n, p)$-LWR oracle which either outputs "real" samples of the form $\left(\mathbf{A}_i \in \mathbb{Z}_q^{l\times m}, \lfloor \mathbf{Ak} \rfloor_p \in \mathbb{Z}_p^l\right)$ for a fixed (but uniformly random) key $\mathbf{k} \in \mathbb{Z}_q^m$ and "fresh" uniformly random samples $\mathbf{A}_i$ or "random" samples $\left(\mathbf{A}_i \in \mathbb{Z}_q^{l\times m}, \mathbf{r} \in \mathbb{Z}_p^l\right)$. We show how to build a PRF simulation such that any adversary that wins the PRF security game can be used to solve the associated LWR problem.

Consider the following challenger $\mathcal{C}$ in the PRF security game. $\mathcal{C}$ will keep a database consisting of entries of the form $\left(\{0,1\}^*, \mathbb{Z}_q^{l\times m}, \mathbb{Z}_p^l\right)$. Recall that $\mathcal{C}$ has access to a $(\mathbb{Z}_q, n, p)$-LWR oracle $\mathcal{O}$ which is either a "real" or "random" LWR oracle. $\mathcal{C}$ does the following on respective queries:

- On query $x_i \in \{0,1\}^*$ to H, $\mathcal{C}$ checks if x_i exists in the database. If so, it responds with the *second* entry in the database (some $\mathbf{A}_i \in Z_q^{l\times m}$, as we will see soon). If not, it queries the LWR oracle, getting a query of the form $\left(\mathbf{A}_i \in \mathbb{Z}_q^{l\times m}, \mathbf{t}_i \in \mathbb{Z}_p^l\right)$, where $\mathbf{t}$ could be either "real" or "random", depending on the LWR oracle. $\mathcal{C}$ then adds the tuple $(x_i, \mathbf{A}_i, \mathbf{t}_i)$ to the database.

- On query $x_i \in \{0,1\}^*$ to F, $\mathcal{C}$ checks if x_i exists in the database. If so, it responds with the *third* entry in the database (some $\mathbf{t}_i \in Z_q^l$). If not, it queries the LWR oracle, getting a query of the form $\left(\mathbf{A}_i \in \mathbb{Z}_q^{l\times m}, \mathbf{t}_i \in \mathbb{Z}_p^l\right)$, where $\mathbf{t}$ could be either "real" or "random", depending on the LWR oracle. $\mathcal{C}$ then adds the tuple $(x_i, \mathbf{A}_i, \mathbf{t}_i)$ to the database.

Note that if the LWR oracle is "real", then $\mathcal{C}$ simulates F_{RO} perfectly. On the other hand, if the LWR oracle is "random", then $\mathcal{C}$ simulates a truly random function. Thus, any adversary that can win the PRF game with advantage ε can be used to solve the LWR problem with identical advantage, completing the proof. $\square$

3.2 Optimizing lattice parameters

Our goal is to choose the PRF parameters so that the PRF can be evaluated efficiently as an MPC protocol. In particular, we focus on minimizing the online round count, which is important for achieving low latency in our distributed key derivation application.

We observe that if we choose q and p as powers of 2, then both the modulo operation $x \mod q$ as well as the rounding operation $\lfloor x \rceil_p$ can be expressed efficiently by bitwise operations. For an integer x with binary representation $x_1, \ldots, x_n$, we have

$$\lfloor x \mod q \rceil_p = \mathsf{int}(x_{\log_2(q/p)}, \ldots, x_{\log_2(q)}) ,$$

where $\mathsf{int}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$.

As we will see later, this adaptation enables us to evaluate the modulo and rounding operation efficiently in the MPC setting.

3.3 Adapt message space and compose outputs

For our application, we need the PRF to output random values in a prime field $\mathbb{Z}_{p'}$, where p' is determined by the application. However, so far the lattice PRF outputs a vector in $\mathbb{Z}_p^l$, where p is a power of 2.

Adapt message space. As we would like to obtain PRF outputs in $\mathbb{Z}_{p'}$, for some prime p', we will choose $\mathbb{Z}_{p'}$ as the native message space of computation, which is also compatible with many MPC protocols. State-of-the-art MPC protocols allow for non-interactive addition and one-round multiplication over $\mathbb{Z}_{p'}$.

Compose outputs. To convert a vector of random elements over $\mathbb{Z}_p$ into a single random value over $\mathbb{Z}_{p'}$, we use the following map M_{COMP},

$$M_{\text{COMP}} : \mathbb{Z}_p^l \to \mathbb{Z}_{p'}; \mathbf{x} \mapsto \sum_{i=1}^{l} x_i \cdot p^{i-1} .$$

We denote the composed PRF by

$$F_{\text{COMP}} : \mathbb{Z}_q^m \times \{0,1\}^* \to \mathbb{Z}'_p; (\mathbf{k}, x) \mapsto M_{\text{COMP}}(F_{\text{RO}}(\mathbf{k}, x)) \ .$$

Security. The security of the PRF F_{COMP} follows from the security of F_{RO} (Theorem 3.1) and a bound on the statistical distance between the distribution of M_{COMP} over uniform-random inputs and the uniform distribution over $\mathbb{Z}_{p'}$ (Theorem 2.1).

Theorem 3.2. F_{COMP} *satisfies Definition 2.2.*

Proof. By a corollary of Theorem 2.1, we have that

$$\Delta\left(M_{\text{COMP}}(U(\mathbb{Z}_p^l)), U(\mathbb{Z}_{p'})\right) \leq p'/p^l \ .$$

Moreover, by Theorem 3.1 we know that F_{RO} is computationally indistinguishable from $U(\mathbb{Z}_p^l)$. It follows from hybridizing the above statements that F_{COMP} is computationally indistinguishable from $U\left(\mathbb{Z}_{p'}\right)$, and that the advantage $\mathbf{Adv}^{\mathsf{PRF}}[F_{\text{COMP}}, \mathcal{A}]$ is negligible for all polynomially-bounded $\mathcal{A}$. □

4 Implementation & Evaluation

4.1 Implementation

In order to better understand the practical performance of our lattice-based PRF F_{COMP} as an MPC protocol, we implemented and evaluated it using the MP-SPDZ framework [Kel20], which is a framework for evaluating MPC protocols that comes with its own high-level programming language and supports a variety of base MPC protocols. The source code can be found at `https://github.com/torusresearch/MP-SPDZ/blob/lattice-prf/Programs/Source/lattice_prf.mpc`.

MPC setup. We configure MP-SPDZ to use `mal-shamir` as the base MPC protocol, which is a maliciously secure MPC protocol based on Shamir secret sharing that requires an honest majority, and thereby satisfies our robustness requirement. Moreover, it supports prime field message spaces, which suits the message space of our PRF construction. We run our experiments between 3 parties and with security against 1 corrupted party, and we use a prime field $\mathbb{Z}_{p'}$ as the native message space, where p' is a 256-bit prime.

PRF parameters. We choose the lattice PRF parameters q and p as powers of 2, with varying concrete values during the experiments (e.g., $q = 2^{12}$ and $p = 2^8$). Furthermore, we set $l = \lceil(\log_2(p') + \Delta_s)/\log_2(p)\rceil$, where $\Delta_s = 40$ controls the statistical distance between F_{COMP} and F_{RO}.

Key generation. For generating the master PRF key, the parties need to sample secret shares of a uniformly random $\mathbf{k} \in \mathbb{Z}_q^m$. As there is no method for directly sampling secret-shared values modulo q over the message space $\mathbb{Z}_{p'}$, we rely on random bit sampling [RW19]. Concretely, for each entry of $\mathbf{k}$, we sample $\log_2 q$ shared random bits and then accumulate them to obtain a shared random integer in $\mathbb{Z}_q$.

Computation modulo q. Evaluating the PRF in MPC requires computing the matrix-vector product $H(x) \cdot \mathbf{k}$ modulo q, where $H(x)$ is a public matrix and $\mathbf{k}$ is a secret-shared vector.

We first compute the matrix-vector product $H(x) \cdot \mathbf{k}$ over the prime field $\mathbb{Z}_{p'}$. Note that p' must be large enough so that the computation does not wrap. Per row of $H(x)$, this involves m cleartext-ciphertext multiplications and $m - 1$ ciphertext-ciphertext additions, which can all be done locally. Let $\mathbf{y} \in \mathbb{Z}_{p'}^l$ denote the resulting vector.

Next, we compute each entry of $\mathbf{y}$ modulo q. Since q is a power of 2, this can be done using algorithm `sint.mod2m` of MP-SPDZ, which runs in $1 + \lceil\log_2^2(q)\rceil$ online rounds and makes use of pre-shared random bits.

Rounding operation. Given the vector $\mathbf{y}' = H(x) \cdot \mathbf{k} \mod q$, we next compute the rounding operation $\lfloor \mathbf{y}' \rfloor_p$. Concretely, this means we need to map each element y_i' of $\mathbf{y}'$ to the largest integer j such that $j * q/p \leq y_i'$. Since q and p are powers of 2, this is equivalent to cutting off the $\log_2(q/p)$ lowest order bits of y_i'. In MP-SPDZ this can be done using algorithm `sint.right_shift`, which runs in $1 + \lceil\log_2^2(q/p)\rceil$ online rounds and makes use of pre-shared random bits.

Composition. Finally, we compose the entries of the rounded output vector $\lfloor \mathbf{y}' \rfloor_p$ into a single uniform random value $y'' \in \mathbb{Z}_{p'}$ by evaluating M_{COMP}. This involves plaintext-ciphertext multiplications and ciphertext-ciphertext additions, which can all be done locally.

Total round count. The total round count for computing $F_{\text{COMP}}(x)$ in MPC is $2 + \log_2^2(q) + \log_2^2(q/p)$. For $q = 2^{12}$ and $p = 2^8$, this results in a MPC PRF protocol running in 8 online rounds, assuming sufficiently many shared random bits have been generated during preprocessing.

4.2 Parameter selection

In the following, we describe the lattice parameters that we use for our evaluation. Our parameter selection is inspired by the parameters of Frodo [BCD+16] with 128 bit classical security, as this scheme relies on a similar assumption.

As the lattice dimension we choose a fixed value of $m = 512$. We note that the lattice dimension does not affect the complexity of the multi-party computation. This is because only the size of the secret key $\mathbf{k}$ and the hash output $H(x)$ depend on m. However, as the vector product $H(x) \cdot \mathbf{k}$ collapses these vector elements onto a single element, and this is computed locally at the start of the protocol, the interactive part of the protocol does not depend on m at all.

For the modulo parameter q and the rounding parameter p, we use the two parameter sets $(q, p) = (2^{12}, 2^8)$ and $(q, p) = (2^{32}, 2^{24})$. Here, smaller values optimize for round complexity, while larger values optimize for preprocessing complexity due to the lower number of pre-shared random bits required.

4.3 Evaluation

We measure the performance of our distributed PRF protocol for the parameters described in Section 4.2. We run our experiments on a single machine with an M1 Pro CPU, 32GB of RAM, and without network latency. The results are shown in Table 1.

Parameter set	**Time (ms)**	**Data (MB)**	**Rounds**	**Bits**
$(q, p) = (2^{12}, 2^8)$	11.07	0.41	8	5328
$(q, p) = (2^{32}, 2^{24})$	7.65	0.40	10	2688

Table 1: Measurements for running one evaluation of our distributed lattice-based PRF protocol for different lattice parameters. *Time* is the time required to run the protocol on a single machine without network latency. *Data* is the total amount of data sent between all parties. *Rounds* is the number of online communication rounds. *Bits* is the number of pre-shared random bits consumed.

Comparison with other protocols. We compare our distributed PRF protocol with existing constructions in Table 2, relying on the implementations from [GØS+23]. Most of the other protocols are designed for producing multiple field elements per protocol evaluation, and for some protocols producing 2 elements is the minimum. Therefore, we compare the different protocols for the case of producing 2 field elements, which results in a doubling of the communication data size of our protocol.

Overall, our protocol compares favorably in terms of running time and round complexity. However, we note that due to the large number of random bits consumed by our protocol, it requires siginificantly more communication, especially during the preprocessing phase.

Protocol	**Time (ms)**	**Data (MB)**	**Rounds**
Ciminion [DGG+21]	31.93	0.28	283
GMiMC [AGP+19]	59.04	0.28	670
HadesMiMC [GLR+20]	47.81	0.19	466
Hydra [GØS+23]	21.42	0.07	140
MiMC [AGR+16]	33.18	0.25	331
Rescue [AABS+20]	26.35	0.28	124
Our protocol $(2^{12}, 2^{8})$	18.76	0.82	8
Our protocol $(2^{32}, 2^{24})$	11.34	0.80	10

Table 2: Comparison of existing distributed PRF protocols with our protocol for generating 2 field elements, measured using MP-SPDZ.

References

[AABS+20] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Transactions on Symmetric Cryptology*, 2020(3):1–45, Sep. 2020.

[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.

[ADD+23] M. R. Albrecht, A. Davidson, A. Deo, and D. Gardham. Crypto dark matter on the torus: Oblivious prfs from shallow prfs and fhe. Cryptology ePrint Archive, Paper 2023/232, 2023. `https://eprint.iacr.org/2023/232`.

[AGP$^+$19] M. R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger. Feistel structures for mpc, and more. In K. Sako, S. Schneider, and P. Y. A. Ryan, editors, *Computer Security – ESORICS 2019*, pages 151–171, Cham, 2019. Springer International Publishing.

[AGR$^+$16] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 191–219, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[ARS$^+$15] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for mpc and fhe. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BBU$^+$19] W. Beullens, T. Beyne, A. Udovenko, and G. Vitto. Cryptanalysis of the legendre PRF and generalizations. *IACR Cryptol. ePrint Arch.*, page 1357, 2019.

[BCD$^+$16] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. Cryptology ePrint Archive, Paper 2016/659, 2016. `https://eprint.iacr.org/2016/659`.

[BDH$^+$17] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. V. Assche, and R. V. Keer. Farfalle: parallel permutation-based cryptography. *IACR Trans. Symmetric Cryptol.*, 2017(4):1–38, 2017.

[BIP$^+$18] D. Boneh, Y. Ishai, A. Passelègue, A. Sahai, and D. J. Wu. Exploring crypto dark matter:. In A. Beimel and S. Dziembowski, editors, *Theory of Cryptography*, pages 699–729, Cham, 2018. Springer International Publishing.

[BLM$^+$13] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. Key homomorphic prfs and their applications. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

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

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012.

[BR93] M. Bellare and P. Rogaway. Random oracle are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[DBN$^+$01] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[Dfn23] Dfns. Dfns web page. `https://www.dfns.co`, July 2023.

[DGG$^+$21] C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. Ciminion: Symmetric encryption based on toffoli-gates over large finite fields. In A. Canteaut and F. Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2021.

[DKL$^+$19] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ecdsa from ecdsa assumptions: The multiparty case. In *2019 IEEE*

Symposium on Security and Privacy (SP), pages 1051–1066, 2019.

[Dwo15] M. Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions, 2015-08-04 2015.

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

[GLR$^+$20] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. On a generalization of substitution-permutation networks: The hades design strategy. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 674–704, Cham, 2020. Springer International Publishing.

[GØS$^+$23] L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. From farfalle to megafono via ciminion: The PRF hydra for MPC applications. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 255–286. Springer, 2023.

[GRR$^+$16] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. Mpc-friendly symmetric key primitives. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 430–443, New York, NY, USA, 2016. Association for Computing Machinery.

[HJK$^+$95] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 339–352, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[Kel20] M. Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[KG21] C. Komlo and I. Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. In O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography*, pages 34–65, Cham, 2021. Springer International Publishing.

[Lit23] Lit Protocol. Lit protocol web page. `https://www.dfns.co`, July 2023.

[MK23] E. V. Mangipudi and A. P. Kate. D-kode: Distributed mechanism to manage a billion discrete-log keys. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, AFT '22, page 308–325, New York, NY, USA, 2023. Association for Computing Machinery.

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

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 458–467. IEEE Computer Society, 1997.

[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.

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

[RW19] D. Rotaru and T. Wood. Marbled circuits: Mixing arithmetic and boolean circuits with active security. In F. Hao, S. Ruj, and S. Sen Gupta, editors, *Progress in Cryptology – INDOCRYPT 2019*, pages 227–249, Cham, 2019. Springer International Publishing.

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

[Web23] Web3Auth. Web3Auth web page. `https://web3auth.io`, July 2023.

A Proof of Theorem 2.1

Proof. Let $m, n \in \mathbb{N}$, $X = U(\mathbb{Z}_m)$, and $Y = U(\mathbb{Z}_n) \bmod m$. By the definition of the statistical distance, we have

$$\begin{aligned} \Delta(X,Y) &= \frac{1}{2} \sum_{a \in \mathbb{Z}_m} |\Pr[X = a] - \Pr[Y = a]| \\ &= \frac{1}{2} \sum_{a \in \mathbb{Z}_m} |1/m - \Pr[Y = a]| \ . \end{aligned} \tag{1}$$

For $a \in \mathbb{Z}_m$, define $D_a = \{x \in \mathbb{Z}_n : x \bmod m = a\}$. We have

$$\Pr[Y = a] = \sum_{b \in D_a} \Pr[U(\mathbb{Z}_n) = b] = |D_a|/n \ , \tag{2}$$

where

$$|D_a| = \begin{cases} \lceil n/m \rceil & \text{if } a < n \bmod m, \\ \lfloor n/m \rfloor & \text{else.} \end{cases} \tag{3}$$

Next, we determine an upper bound on $|1/m - \Pr[Y = a]|$ by writing out the equation using (2) and (3). For $a < n \bmod m$, we have

$$\left| \frac{1}{m} - \Pr[Y = a] \right| = \left| \frac{1}{m} - \frac{\lceil n/m \rceil}{n} \right| = \left| \frac{1}{m} - \frac{n + m - n \bmod m}{mn} \right| \leq \frac{1}{n} \ , \tag{4}$$

and for $a \geq n \bmod m$,

$$\left| \frac{1}{m} - \Pr[Y = a] \right| = \left| \frac{1}{m} - \frac{\lfloor n/m \rfloor}{n} \right| = \left| \frac{1}{m} - \frac{n - n \bmod m}{mn} \right| \leq \frac{1}{n} \ . \tag{5}$$

Finally, by combining (1), (4), and (5), we obtain an upper bound on the statistical distance of X and Y,

$$\Delta(X,Y) \leq \frac{1}{2} \sum_{a \in \mathbb{Z}_m} 1/n \leq m/n \ .$$

$\square$