Efficient Quantum-safe Distributed PRF and Applications: Playing DiSE in a Quantum World

Sayani Sinha[1], Sikhar Patranabis[2], and Debdeep Mukhopadhyay[1]

[1]IIT Kharagpur
[2]IBM Research India

January 31, 2025

Abstract

We propose the first *distributed* version of a simple, efficient, and provably quantum-safe pseudorandom function (PRF). The distributed PRF (DPRF) supports arbitrary threshold access structures based on the hardness of the well-studied Learning with Rounding (LWR) problem. Our construction (abbreviated as PQDPRF) practically outperforms not only existing constructions of DPRF based on lattice-based assumptions, but also outperforms (in terms of evaluation time) existing constructions of: (i) classically secure DPRFs based on discrete-log hard groups, and (ii) quantum-safe DPRFs based on any generic quantum-safe PRF (e.g. AES). The efficiency of PQDPRF stems from the extreme simplicity of its construction, consisting of a simple inner product computation over $\mathbb{Z}_q$, followed by a rounding to a smaller modulus $p < q$. The key technical novelty of our proposal lies in our proof technique, where we prove the correctness and post-quantum security of PQDPRF (against semi-honest corruptions of any less than threshold number of parties) for a polynomial q/p (equivalently, "modulus to modulus")-ratio.

Our proposed DPRF construction immediately enables efficient yet quantum-safe instantiations of several practical applications, including key distribution centers, distributed coin tossing, long-term encryption of information, etc. We showcase a particular application of PQDPRF in realizing an efficient yet quantum-safe version of distributed symmetric-key encryption (DiSE – originally proposed by Agrawal et al. in CCS 2018), which we call PQ – DiSE. For semi-honest adversarial corruptions across a wide variety of corruption thresholds, PQ – DiSE substantially outperforms existing instantiations of DiSE based on discrete-log hard groups and generic PRFs (e.g. AES). We illustrate the practical efficiency of our PQDPRF via prototype implementation of PQ – DiSE.

Contents

1 Introduction

Threshold Cryptography. The privacy guarantees of any (computationally secure) cryptosystem fundamentally rely on the secure storage of a secret key. If the secret key is stored on a single server, this server becomes the single point of vulnerability, i.e., if an adversary successfully manages to corrupt the server, the secret key is retrieved and the security of the whole system is compromised. Threshold cryptography provides a solution to this problem by allowing the secret key to remain distributed among multiple (say, T) servers in the form of several key shares. Among them, if t servers (for $1 < t \leq T$) can collaborate with their respective key shares to successfully perform the cryptographic computation without any knowledge of the actual secret key, we call it (t, T)-threshold cryptography. An underlying threshold secret sharing algorithm makes sure that collaboration of at least t servers is necessary to reconstruct the secret, or in other words, less than t servers together can not reconstruct the secret. Hence, if an adversary manages to corrupt $(t-1)$ number of servers (at most) in a (t, T)-threshold cryptosystem, the system still continues to remain secure, as the adversary can not retrieve the actual secret from secret shares of $(t-1)$ servers.

In this paper, we focus on threshold cryptographic systems [BGG+18, AJLA+12] where the secret key is shared once during the initial setup phase (either by a trusted dealer or in a decentralized manner) and is *never* explicitly reconstructed in the clear. Subsequently, any cryptographic computation is performed in two phases: (a) first, each of the participating t servers does the some *partial computation* with its own key share, and then (b) these partial computations are combined together either by one of the participating servers or a separate evaluating entity to get the final result. Crucially, the combination process should leak no additional information about the secret key beyond what is revealed by the final output.

Threshold PRF and Applications. The concept of shared evaluation of a PRF was initially proposed in [MS95], albeit for restricted threshold access structures. This was generalized to arbitrary threshold access structures in follow-up works [NPR99, Nie02, NR04], with new applications in [AMMR18, CGMS21]. In a threshold or distributed PRF, the PRF key is distributed across multiple (say, T) parties, and evaluations can be performed on any given input in a distributed manner by a threshold $t \in [2, T]$ number of parties. Informally, the primitive retains its pseudorandomness guarantees against any adversary that corrupts $t' < t$ parties. Some applications of a distributed PRF are as follows.

- **Distributed KDC [NPR99]:** Key Distribution Center (KDC) provides keys to the users in a network that shares sensitive data. Usually, there is a dedicated key between the KDC and each user in the network. Whenever two users have to communicate securely, one of them requests a key to the KDC. KDC chooses a random key and sends it to each of the two parties, keeping it encrypted with their respective dedicated keys. The users can then decrypt it and retrieve the key for the secure communication session between them. This approach was introduced by Needham and Schroeder in [NS78], and KDC has been widely implemented in Kerberos System[1]. However, KDC is a single point of vulnerability as it stores the dedicated keys of all the users. KDC, being a single point of contact, also suffers from the availability problem whenever there is a need for communication between multiple pairs of users or communication is needed among a set of more than two parties. To avoid these scenarios, distributed KDC is considered, which consists of multiple (say, T) servers to service the key requests, and a user can contact any available subset of t servers out of them and receive a key irrespective of which particular subset it contacted. Distributed PRF is a

[1] `https://web.mit.edu/kerberos/`

building block of distributed KDC [NPR99].

- **DiSE [AMMR18, CGMS21]:** A formal construction of threshold distributed symmetric-key encryption (DiSE) was proposed in [AMMR18], where a user has to contact t-out-of-T servers for both encryption and decryption. The construction is discussed in detail later in Section 4.1 as an application context of threshold PRF.

Post-quantum Security. Once physical quantum computer comes to existence [AAB+19], various quantum algorithms [Sho94, Sho99, Gro96] can be used to break classical cryptosystems built on the hardness assumption of mathematical problems like integer factorization, discrete logarithm. Lattice-based cryptography offers security against cryptanalytic attack by quantum computers. Although NIST[1] launched standardization process of quantum-safe asymmetric-key cryptography, need for standard post-quantum symmetric-key cryptography [BUK19] still persists. As PRF is a building block of various symmetric-key primitives, we take a step towards this goal by constructing a simple but efficient quantum-safe threshold PRF.

1.1 Our Contributions

Our contributions are as summarized below.

First Practically Efficient Quantum-safe Distributed PRF. We provide the first non-interactive *distributed* version of a simple but efficient quantum-safe PRF (PQDPRF) in random oracle model based on lattice-based Learning with Rounding (LWR) assumption. Such efficient straight-forward construction of quantum-safe distributed PRF with polynomial ratio between input and output modulus is the first of its kind to the best of our knowledge. We claim novelty of our contribution with respect to existing works as follows.

- Efficient DPRFs with their possible application areas have been proposed [NPR99, Nie02], but they are not quantum-safe.
- LWR assumption was introduced in [BPR12] along with a proposed PRF in standard model, but their construction was inefficient as it required superpolynomial modulus-to-modulus ratio. Also, the aspect of thresholdization was not captured there.
- Some other PRF contructions based on variants of LWE assumption [BLMR13, BP14, BV15] can further be used in constructing threshold PRF. "Universal thresholdizer" tool by [BGG+18] can be used to construct threshold PRF from an underlying threshold FHE protocol. However, none of them is a straightforward and efficient approach to designing quantum-safe threshold PRF.
- A robust non-interactive lattice-based DPRF construction with theoretically efficient parameters is proposed in [LST21] for adaptive corruption settings. Although our construction is in the semi-honest setting against static corruptions, its main advantage lies in its simplicity, superior practical efficiency, and ease of implementation as compared to the scheme in [LST21]. In particular, our proposed construction provides a practically efficient quantum-safe drop-in replacement for AES/DDH-based DPRFs for applications (e.g., DiSE [AMMR18], distributed KDC [NPR99]) where robustness can be achieved more efficiently and directly at the application level instead of trying to achieve the same at the DPRF level, requiring costlier and more mathematically involved techniques.

[1] National Institute of Standards and Technology

We prove the correctness, consistency, and security of our proposed PQDPRF, described in Section 3.2 and Section 3.3.

A Practical Use-case of Proposed Distributed PRF. We validate the efficacy of proposed PQDPRF by plugging it into existing DiSE (distributed symmetric-key encryption) protocol [AMMR18], to get an improved quantum-safe version of DiSE, which we call PQ − DiSE. We also show that our proposed LWR-based DPRF, apart from being quantum-safe, is more efficient than other DPRFs (i.e., DDH-based DPRF and AES-based DPRF) previously used in DiSE, and consequently, PQ − DiSE outperforms DiSE in terms of throughput (number of encryptions per second). We emphasize that, to the best of our knowledge, no prior work has actually explored practical implementations and prototype realizations of applications such as in [AMMR18, CGMS21] based on quantum-safe distributed PRFs from lattice-based assumptions.

2 Preliminaries and Background

This section presents notations and background material.

2.1 Notation

The notation $x \leftarrow \mathcal{X}$ signifies that x is sampled according to distribution $\mathcal{X}$, whereas $x \xleftarrow{R} X$ means that, x is uniform random choice over set X. Upper case (e.g., $\mathbf{A}$) and lower case (e.g., $\mathbf{a}$) variables in bold denote a matrix and a vector, respectively. With two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$, $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i b_i \pmod q$ represents their vector dot product modulo q. The cardinality of a set S is denoted by $|S|$. The notation $[n]$ for some $n \in \mathbb{N}$ denotes the set $\{1, \ldots, n\}$. For any $y \in \mathbb{Z}_q$, the round-off operation, denoted by $\lfloor y \rceil_p$ gives the nearest integral value of $(y \cdot \frac{p}{q})$ in $\mathbb{Z}_p$; in particular, if $(y \cdot \frac{p}{q})$ has a fractional part exactly equal to 0.5, we choose to always round it down to $\lfloor y \cdot \frac{p}{q} \rfloor$ to avoid ambiguity. We can apply the round-off operator to vectors and matrices as well to denote element-wise round-off operation. A negligible function of λ is denoted by $\mathsf{negl}(\lambda)$; $\mathsf{poly}(\lambda)$ denotes a polynomial function of λ. Terms "threshold PRF" and "distributed PRF" are used alternatively throughout the paper.

2.2 Some Terminologies and Definitions

Here, we provide definitions of some terminologies that have been used frequently in the paper.

Threshold Access Structure. Let $\mathcal{P} = \{P_1, \ldots, P_T\}$ be a set of T parties, and suppose that some secret k is distributed among them in form of secret shares. Access structure is a set consisting of all "valid" subsets $\overline{\mathcal{P}} \subseteq \mathcal{P}$ of parties that can recover the secret k by combining their key shares together. For any $t, T \in \mathbb{N}$ $(t \leq T)$, a minimal (t, T)-threshold access structure over $\mathcal{P}$ is defined as a collection of valid subsets of the form $\mathbb{A}_{t,T} = \{\overline{\mathcal{P}} \subseteq \mathcal{P} : \left|\overline{\mathcal{P}}\right| = t\}$, such that we have $|\mathbb{A}_{t,T}| = \binom{T}{t}$ as the number of valid subsets.

Monotone Boolean Formula (MBF). A Boolean formula is monotone if it has a single output and it consists of only AND and OR combination of Boolean variables. Note that any (t, T)-threshold access structure $\mathbb{A}_{t,T}$ can be represented by a MBF. The fact that a particular collaboration of t parties is able to reconstruct the secret, is captured by Boolean formula of the

form $(x_1 \wedge \ldots \wedge x_t)$ and that any such t-collaboration is a way of reconstruction, is captured by ORing $\binom{T}{t}$ such terms. For e.g., $\mathbb{A}_{3,4}$ is represented by $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_4)$.

Pseudo Random Function (PRF). We recall the formal definition of a pseudorandom function (PRF). Let $\mathcal{F} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of pseudo random functions and $\mathcal{F}' = \{f' | f' : \mathcal{X} \to \mathcal{Y}\}$ be the set of all possible functions with the same domain and range. Let us assume that, $f_k \in \mathcal{F}$ uses a uniform random secret $k \xleftarrow{R} \mathcal{K}$ and, on input $x \in \mathcal{X}$, outputs $f_k(x)$, using both k and x. Then, the advantage of any PPT distinguisher $\mathcal{D}$ is negligible, i.e.,

$$\left| \mathsf{Pr}[\mathcal{D}^{f_k(\cdot)}(1^\lambda) = 1] - \mathsf{Pr}[\mathcal{D}^{f'(\cdot)}(1^\lambda) = 1] \right| \leq \mathrm{negl}(\lambda),$$

where λ is a security parameter. The first probability is taken over uniform choice of k and randomness of $\mathcal{D}$, and the second probability is taken over uniform choice of f' and randomness of $\mathcal{D}$.

Weak Pseudo Random Function. A PRF is weak if its output is pseudorandom, only when the inputs are uniformly random over the input space. This is in contrast to the case of (strong) PRF, where indistinguishability holds for any input from the input space. However a weak PRF can be converted to a PRF by relying on existence of a random oracle. If $f_k(\cdot) : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a weak PRF and $\mathcal{H} : \{0,1\}^\star \to \mathcal{X}$ is a hash function modeled as a random oracle, then $g_k(\cdot) = f_k(\mathcal{H}(\cdot))$ is a PRF [NPR99].

Learning with Rounding (LWR) Problem. LWR problem is a "derandomized" version of Learning with Errors (LWE) problem, first introduced in [BPR12]. Given a parameter $n \in \mathbb{N}$, two moduli $q, p \in \mathbb{N}$ such that $q > p \geq 2$, the LWR distribution L_s for a secret $\mathbf{s} \in \mathbb{Z}_q^n$ is defined over $\mathbb{Z}_q^n \times \mathbb{Z}_p$ of the form $(\mathbf{a}, b)$, where we choose $\mathbf{a} \xleftarrow{R} \mathbb{Z}_q^n$ and then calculate $b = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p$. The decision LWR problem is to distinguish samples of $L_{\mathbf{s}}$ from uniformly random samples of $\mathbb{Z}_q^n \times \mathbb{Z}_p$.

2.3 Distributed PRF (DPRF)

If the evaluation of a PRF is performed in a distributed way, we call it a distributed PRF. In this case, the secret k of the PRF always remains distributed as shares among multiple parties. Here, we define distributed PRF formally.

Definition 1 (Distributed Pseudo Random Function (DPRF))**.** *Let $\mathcal{P} = \{P_1, \ldots, P_T\}$ be a set of T parties, and let $\mathbb{S}$ be a class of threshold access structures on $\mathcal{P}$. A threshold PRF scheme for $\mathbb{S}$ over an input space $\mathcal{X}$ and key space $\mathcal{K}$ is a tuple of probabilistic polynomial-time algorithms as follows,*

$$\mathsf{DPRF} = (\mathsf{DPRF.Setup}, \mathsf{DPRF.PartialEval}, \mathsf{DPRF.FinalEval}).$$

$\underline{\mathsf{DPRF.Setup}(1^\lambda, \mathbb{A})}$*: On input the security parameter λ and an access structure $\mathbb{A} \in \mathbb{S}$, this algorithm generates a key $k \xleftarrow{R} \mathcal{K}$, and then generates multiple key shares of k corresponding to $\mathbb{A}$. At the end of key sharing among T parties, the actual key k is not stored anywhere. Each party has to store one or more than one key share depending on the particular threshold secret sharing scheme used.*

$\underline{\mathsf{DPRF.PartialEval}(x, P_i, A)}$*: On input a valid subset $A \in \mathbb{A}$, an input $x \in \mathcal{X}$ and a party $P_i \in A$, the appropriate key share (say, k_i) of P_i corresponding to A is chosen and a partial evaluation $f_{k_i}(x)$ is returned.*

$\underline{\mathsf{DPRF.FinalEval}}(A, \{f_{k_i}(x)\}_{P_i \in A})$*: On input a valid subset* $A \in \mathbb{A}$ *and all the partial evaluations by parties* $P_i \in A$*, this algorithm combines them to get the final PRF evaluation. The actual combination procedure depends upon the reconstruction property of the underlying threshold secret sharing scheme.*

Correctness and Consistency. A (t, T)-distributed PRF with $f_k(\cdot)$ as its underlying PRF is *correct* if given an input, its distributed evaluation by any valid subset $A \in \mathbb{A}$ outputs the same value as would be obtained by directly evaluating $f_k(\cdot)$ on the same input with high probability.

$$\mathsf{Pr}[\mathsf{DPRF.FinalEval}(A, \{\mathsf{DPRF.PartialEval}(x, P_i, A)\}_{P_i \in A}) = f_k(x)] \geq 1 - \epsilon.$$

Here, $0 < \epsilon < 1$ is a very small fraction whose value can be adjusted based on the practical instantiation of the DPRF in an application. A (t, T)-distributed PRF is *consistent* if distributed evaluation on a given input by any two distinct valid subsets $S_1, S_2 \in \mathbb{A}$ outputs the same value with high probability, i.e.,

$$\Big|\mathsf{Pr}[\mathsf{DPRF.FinalEval}(S_1, \{\mathsf{DPRF.PartialEval}(x, P_i, S_1)\}_{P_i \in S_1}) \\ \neq \mathsf{DPRF.FinalEval}(S_2, \{\mathsf{DPRF.PartialEval}(x, P_j, S_2)\}_{P_j \in S_2})]\Big| \leq \epsilon'.$$

Here, ϵ' is some small fraction depending on the application. Note that the correctness of (t, T)-distributed PRF implies its consistency, but not the other way around.

Security. We borrow the notion of DPRF security from [NPR99].

The adversarial model. We assume a probabilistic polynomial time (PPT) adversary that can statically corrupt (i.e., announces the set of corrupt parties before the partial evaluation query phase starts) at most $(t-1)$ number of parties and each party if corrupted, is honest but curious.

The security notion. Let $\mathcal{P} = \{P_1, \ldots, P_T\}$ be the set of T parties and $\mathcal{A}$ be a PPT adversary as described above. Let $\mathcal{P}'$ be a statically corrupted set such that, $|\mathcal{P}'| = (t-1)$. Hence, after $\mathsf{DPRF.Setup}(1^\lambda, \mathbb{A}_{t,T})$ is run, $\mathcal{A}$ has access to key shares of each $P_i \in \mathcal{P}'$. We say that DPRF is secure if the winning probability of $\mathcal{A}$ against a challenger $\mathcal{C}$ in the following game is negligible.

Game:

1. $\mathcal{A}$ sends a query input $x \in \mathcal{X}$ to $\mathcal{C}$. $\mathcal{C}$ sends $(f_k(x), \{f_{k_i}(x)\}_{P_i \in \mathcal{P} \setminus \mathcal{P}'})$ to $\mathcal{A}$, where $f_{k_i}(x) = \mathsf{DPRF.PartialEval}(x, P_i, \mathcal{P}' \bigcup P_i)$.
2. The above step is repeated at most a priori bounded number of times for adaptive choice of query input $x \in \mathcal{X}$.
3. $\mathcal{A}$ sends a new challenge query $x^\star$ (different from query phase inputs) to $\mathcal{C}$.
4. $\mathcal{C}$ chooses a random bit $b \xleftarrow{R} \{0, 1\}$. If $b = 0$, it sends $f_k(x^\star)$ to $\mathcal{A}$, otherwise, it sends some $y \xleftarrow{R} \mathcal{Y}$ to $\mathcal{A}$, where $\mathcal{Y}$ is the range of underlying PRF.
5. $\mathcal{A}$ has to output a distinguishing bit b'.

$\mathcal{A}$ wins the game, if $b = b'$.

2.4 (t, T)-Threshold Secret Sharing

A threshold secret sharing scheme is an essential underlying primitive to build a distributed PRF protocol. A (t, T)-threshold secret sharing scheme shares a key k among these T parties in

such a way that any t or more parties are able to reconstruct it from their respective shares, though collaboration of less than t parties does not suffice. We choose to use Benaloh-Leichter Linear Integer Secret Sharing Scheme (LISSS) as described in [DT06]. The secret sharing scheme is "linear integer" because key shares can be linearly combined to get the actual secret back in a way that the coefficients of the linear combination are integers. These coefficients used during the reconstruction of the secret are called *recovery coefficients*. Though the original Benaloh-Leichter LISSS shares a scalar secret, it can naturally be extended to share a secret in vector form. As we deal with secrets belonging to $\mathbb{Z}_q^n$ in later sections, we describe the LISSS scheme in the context of sharing a secret vector $\mathbf{k} \in \mathbb{Z}_q^n$ here.

2.4.1 Preprocessing.

Here, we discuss some necessary preprocessing steps for threshold secret sharing.

Formation of distribution matrix M: Formation of distribution matrix $\mathbf{M}$ depends upon the MBF, representing a (t, T)-threshold access structure. As any MBF is a combination of AND and OR of Boolean variables, we need to focus on the three following cases.
Each Boolean variable x_i corresponds to a singleton matrix with 1 as its only element.
AND-ing of $\mathbf{M_{f_a}}$ *and* $\mathbf{M_{f_b}}$*:* Let $\mathbf{M_{f_a}}$ with dimension $d_a \times e_a$ and $\mathbf{M_{f_b}}$ with dimension $d_b \times e_b$ be the distribution matrices for Boolean formulae f_a and f_b respectively. Then we form $\mathbf{M_{f_a \wedge f_b}}$ as follows:

c_a	c_a	C_a	0
0	c_b	0	C_b

Here, c_a and c_b denote the first column of $\mathbf{M_{f_a}}$ and $\mathbf{M_{f_b}}$ respectively. C_a and C_b denote the rest of the columns of $\mathbf{M_{f_a}}$ and $\mathbf{M_{f_b}}$ respectively. $\mathbf{M_{f_a \wedge f_b}}$ has dimension $(d_a + d_b) \times (e_a + e_b)$.
OR-ing of $\mathbf{M_{f_a}}$ *and* $\mathbf{M_{f_b}}$*:* Let $\mathbf{M_{f_a}}$ with dimension $d_a \times e_a$ and $\mathbf{M_{f_b}}$ with dimension $d_b \times e_b$ be the distribution matrices for Boolean formulae f_a and f_b respectively. Then we form $\mathbf{M_{f_a \vee f_b}}$ as follows:

c_a	C_a	0
c_b	0	C_b

Here, c_a and c_b denote the first column of $\mathbf{M_{f_a}}$ and $\mathbf{M_{f_b}}$ respectively. C_a and C_b denote the rest of the columns of $\mathbf{M_{f_a}}$ and $\mathbf{M_{f_b}}$ respectively. $\mathbf{M_{f_a \vee f_b}}$ has dimension $(d_a + d_b) \times (e_a + e_b - 1)$.
It can be easily verified that, the distribution matrix $\mathbf{M}$ for (t, T)-threshold secret sharing has dimension $d \times e$, where $d = \binom{T}{t} t$ and $e = (1 + \binom{T}{t}(t-1))$.

Formation of share matrix $\boldsymbol{\rho}$: $\boldsymbol{\rho}$ is a matrix with dimension $e \times n$. Its first row is populated from the n elements of the actual secret vector $\mathbf{k} \in \mathbb{Z}_q^n$. The rest of the elements of the matrix are filled uniformly randomly from $\mathbb{Z}_q$.

2.4.2 Sharing.

First we compute the matrix $\mathbf{M}\boldsymbol{\rho}$ that has $d = \binom{T}{t} t$ rows. Each of the rows is a unique key share. Note that the number of t-sized subset of $\mathcal{P}$ is $\binom{T}{t}$ and each of the t parties in a t-sized subset will hold a keyshare corresponding to that specific group, which justifies $d = \binom{T}{t} t$ to be the total number of unique keyshares. For ease of explanation, we identify each keyshare with

the following two attributes: (1) party_id (which party the key share belongs to), (2) group_id (which t-sized group the key share is used for). By enumerating over all t-sized subsets and tagging them with corresponding enumerating serial numbers, we get group_id's of all t-sized subsets.

Now sharing of d rows among T parties happens in the following manner: we consider $d = \binom{T}{t}t$ rows as $\binom{T}{t}$ chunks of rows of size t. Now for $i \in [\binom{T}{t}]$, we pick i^{th} such chunk at a time and assign each of the t rows to parties belonging to the subset with group_id i. For example, in a $(3, 5)$-threshold secret sharing, subset $\{P_1, P_2, P_3\}$ has group_id 1, so first three rows of $\mathbf{M}\boldsymbol{\rho}$ are assigned to P_1, P_2, P_3 respectively. $\{P_1, P_2, P_4\}$ has group_id 2, so next three rows of $\mathbf{M}\boldsymbol{\rho}$ are assigned to P_1, P_2, P_4 respectively and so on.

2.4.3 Reconstruction.

Any t-sized group of parties, with their key shares, should be able to reconstruct $\mathbf{k}$. Given $\mathcal{P}' = \{P_{i_1}, P_{i_2}, \ldots, P_{i_t}\} \subset \mathcal{P}$ with $i_1 < \cdots < i_t$, each of the t parties will have one key share with group_id corresponding to $\mathcal{P}'$. Let us denote these t key shares as $\{\mathbf{k}_{i_1}, \ldots, \mathbf{k}_{i_t}\}$. In any t-sized group, the party with minimum value of party_id is called the group_leader. Hence, P_{i_1} is the group_leader here. In this LISSS the recovery coefficient is 1 for the group_leader and -1 for the rest of the $(t-1)$ parties. The key $\mathbf{k}$ can be reconstructed as $\mathbf{k} = \mathbf{k}_{i_1} - \sum_{j=2}^{t} \mathbf{k}_{i_j}$. We exploit this reconstruction property in final evaluation of (t, T)-threshold PRF.

Size of Secret Shares. After applying (t, T)-threshold secret sharing on $\mathbf{k} \in \mathbb{Z}_q^n$, each party gets $\binom{T-1}{t-1}$ key shares to store. So each party has to store $\binom{T-1}{t-1} \cdot n \cdot \lceil \log_2 q \rceil$ bits in total.

3 Our Contribution: Proposed Distributed PRF

In this section, we first describe a post-quantum secure PRF in the random oracle model. Next, in Section 3.2, we construct a distributed version of the same PRF such that, if the key is distributed among T parties, participation of all T parties is necessary to evaluate the PRF on a given input. We call it (T, T)-distributed PRF, denoted with $\mathsf{PQDPRF}_{\mathsf{T,T}}$. In Section 3.3, we provide a generalized construction of quantum-safe (t, T)-distributed PRF, denoted with $\mathsf{PQDPRF}_{\mathsf{t,T}}$, where participation of all T parties is no longer a necessity, but the collaboration of at least t ($t \leq T$) parties is required to evaluate the PRF on any given input. In general, PQDPRF refers to both of these schemes in subsequent sections. We elaborate on the choice of parameters for PQDPRF in Section 3.4. Section 3.5 compares our work with existing lattice-based DPRF [LST21].

3.1 Underlying Quantum-safe PRF

We discuss the straightforward construction of underlying quantum-safe PRF from the Learning with Rounding (LWR) assumption in the following.

The PRF Construction
Fixed parameters:

- Key: $\mathbf{k} \xleftarrow{R} \mathcal{K}$, where $\mathcal{K} = \mathbb{Z}_q^n$. [Secret]
- $q \in \mathbb{N}$, modulus of input space. [Public]
- $p \in \mathbb{N}$, modulus of output space. [Public]

Input: $\mathbf{x} \in \mathcal{X} = \mathbb{Z}_q^n$
Evaluation: $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k} \rangle \rceil_p$, where $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q^n$ is a hash function, modeled as a random oracle.

Choice of LWR over LWE. To realize a distributed PRF, we typically need an algebraically structured PRF with some kind of "deterministic homomorphism" between key space and output space. Unfortunately, it is hard to achieve such an algebraically structured PRF from standard LWE. For example, the natural LWE-based (weak) PRF would be: $f_{\mathbf{k}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{k} \rangle + e$, where the error e needs to be deterministic, and thus needs to be generated using some (weak) PRF as $e = g_{\mathbf{k}}(\mathbf{x})$. Now, unless g is thresholdizable, f can not be thresholdized. Hence, in order to avoid this circular requirement, we resort to LWR, where the rounding operation enables deterministic homomorphism. Several advantages of choosing LWR over LWE in the construction of PRF have been discussed in [Mon18].

Post-quantum Security of the PRF. We discuss the security of underlying PRF here.

Theorem 1. *The above construction of PRF is secure in the random oracle model if the LWR assumption holds.*

Proof. We assume a distinguisher $\mathcal{D}$ which distinguishes PRF outputs on a polynomial number of inputs of its choice from outputs of a truly random function on the same set of inputs. Assuming that an LWR challenger $\mathcal{C}$ chooses to always generate samples either from an LWR distribution with fixed secret $\mathbf{k} \in \mathcal{K}$ or from uniform random distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_p$, we build another distinguisher $\mathcal{D}'$ to distinguish LWR samples from uniformly random samples generated by $\mathcal{C}$ in the following manner:

- $\mathcal{D}$ sends an input $\mathbf{x}_i$ of its choice to $\mathcal{D}'$.
- $\mathcal{D}'$ requests for a sample of the form $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ from $\mathcal{C}$.
- Upon receiving $(\mathbf{a}_i, b_i)$, $\mathcal{D}'$ now programs the random oracle such that $\mathcal{H}(\mathbf{x}_i) = \mathbf{a}_i$. It returns b_i to $\mathcal{D}$ as the output for input $\mathbf{x}_i$.
- After polynomial repetitions of above three steps, $\mathcal{D}$ returns a distinguishing bit b.
- $\mathcal{D}'$ forwards the same bit b as distinguishing bit to $\mathcal{C}$.

If $\mathcal{C}$ chooses to generate all $(\mathbf{a}_i, b_i)$ samples from LWR distribution, b_i is indeed PRF output for an input $\mathbf{x}_i$, since $b_i = \lfloor \langle \mathbf{a}_i, \mathbf{k} \rangle \rceil_p = \lfloor \langle \mathcal{H}(\mathbf{x}_i), \mathbf{k} \rangle \rceil_p = f_{\mathbf{k}}(\mathbf{x}_i)$. On the other hand if $\mathcal{C}$ chooses to generate samples from uniform distribution, b_i is a TRF (truly random function) output for input $\mathbf{x}_i$. Hence, if $\mathcal{D}$ guesses b correctly with non-negligible probability, $\mathcal{D}'$ wins the game against $\mathcal{C}$ with non-negligible probability, thus breaking the LWR assumption. Therefore, by contradiction, we conclude that the above PRF construction is secure due to LWR assumption with the same moduli p, q.

Furthermore, the proposed PRF is a post-quantum secure construction in random oracle model [1], since it relies upon quantum-safe LWR assumption. □

Polynomial Modulus-to-modulus Ratio of LWR Parameters. During the introduction of the LWR assumption [BPR12], the hardness of decision-LWR problem, when derived from the hardness of the well-established decision-LWE problem, required a superpolynomial (in security parameter) "$\frac{q}{p}$" ratio while keeping the dimension (n) and modulus (q) same and allowing unbounded number (m) of adversarial queries. Later, several works [BGM+16, AKPW13] focused on new reduction techniques from LWE problem to LWR problem, which would require only polynomial "$\frac{q}{p}$" ratio, but allow a priori bounded number of adversarial queries and a multiplicative decrease in dimension. Another work [Mon18] proposed a (non-practical) variant of LWR problem where reduction from LWE allows an unbounded number of adversarial queries while achieving a polynomial "$\frac{q}{p}$" ratio. However [ASA16] proposes a dimension-preserving reduction from LWE to LWR problem requiring only polynomial "$\frac{q}{p}$" ratio allowing a priori bounded number of queries. Formally, we summarize the following theorem from Theorem 1.1 of [ASA16].

Theorem 2 (Theorem 1.1 of [ASA16])**.** *Let λ be the security parameter. Let Ψ be a B-bounded LWE noise distribution over $\mathbb{Z}$ and $p, q = \text{poly}(\lambda)$, $m = \text{poly}(\lambda)$, $n \in \mathbb{N}$ with $\frac{q}{p} \geq mB\lambda$. Suppose a PPT adversary $\mathcal{A}$ can distinguish LWR samples with parameter (n, q, p, m) from uniform random samples with advantage $\epsilon \geq \lambda^{-c}$ for some constant $c \geq 1$. In that case, there must exist another adversary $\mathcal{A}'$ which can distinguish LWE samples with parameters (n, q, m, Ψ) from uniform random samples with advantage $\epsilon' = \epsilon(mB)^{-c}$.*

We conclude from the above theorem that it is possible to obtain a hard instance of decision-LWR problem from a hard instance of LWE problem with the same set of parameters (n, q) and polynomially large "$\frac{q}{p}$" ratio.

Concrete Choice of LWR Parameters. Although the theoretical analysis above implies a technical gap between the hardness of LWE problem and LWR problem, no practical attack on LWR exploits this gap to perform better than an attack on LWE with the same set of parameters. Hence, several LWR-based constructions [DKSRV18, CKLS18], including NIST candidates (e.g., SABER) make a more aggressive choice of LWR parameters than what is suggested by the theoretical analysis, to build practically efficient cryptosystems. We follow the same approach while providing a concrete choice of LWR parameters for our construction in Section 3.4. Since the existing attacks on LWR do not capture the loss in security while traversing from an LWE-based construction to an LWR-based construction, we first find a set of LWE parameters for which LWE problem is hard to solve. Then we use them as LWR parameters while maintaining polynomial "$\frac{q}{p}$" ratio.

3.2 Proposed (T, T)-Distributed PRF

Here, we propose the formal construction of (T, T)-distributed PRF, based on the PRF described in 3.1, and discuss its proof of correctness, consistency, and security.

Construction 1 (Post-quantum Secure (T, T)-distributed Pseudo Random Function ($\mathsf{PQDPRF}_{\mathsf{T},\mathsf{T}}$))**.** $\mathsf{PQDPRF}_{\mathsf{T},\mathsf{T}}$ *is a post-quantum secure (T, T)-distributed PRF over an input space (or, domain)*

[1] We assume quantum adversary with classical access to random oracle here [BDF+11].

$\mathcal{X} = \mathbb{Z}_q^n$ and key space $\mathcal{K} = \mathbb{Z}_q^n$. The range of PRF is $\mathcal{Y} = \mathbb{Z}_p$. Here $q, p \in \mathbb{N}$ are publicly known moduli of input and output space respectively; $q_1 \in \mathbb{N}$ $(p < q_1 < q)$ is another public modulus to be used during partial evaluation. Assume, $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q^n$ is a hash function modeled as a random oracle. Also, let us assume $\mathcal{P} = \{P_1, \ldots, P_T\}$ to be the set of T parties. The access structure $\mathbb{A}_{T,T}$ here is a singleton set, such that, $\mathbb{A}_{T,T} = \{\mathcal{P}\}$. The protocol consists of the following three PPT algorithms,

$$\mathsf{PQDPRF}_{\mathsf{T,T}} = (\mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{Setup}, \mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{PartialEval}, \mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{FinalEval}).$$

$\underline{\mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{Setup}(1^\lambda, \mathbb{A}_{T,T})}$*: First, on input security parameter λ, a key $\mathbf{k} \xleftarrow{R} \mathcal{K}$ is generated. Next, it is distributed among T parties using additive secret sharing such that each party $P_i \in \mathcal{P}$ gets a key share $\mathbf{k_i}$ and $\sum_{i=1}^{T} \mathbf{k_i} = \mathbf{k}$.*

$\underline{\mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{PartialEval}(\mathbf{x}, P_i)}$*: For a given input $\mathbf{x}$, each party P_i partially evaluates the PRF with its own share $\mathbf{k_i}$ as follows: $f_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k_i} \rangle \rceil_{q_1}$, and broadcasts it to other $(T-1)$ parties.*

$\underline{\mathsf{PQDPRF}_{\mathsf{T,T}}.\mathsf{FinalEval}(\{f_{\mathbf{k_i}}(\mathbf{x})\}_{i \in [T]})}$*: Each party having its own partial evaluation and partial evaluations of rest $(T-1)$ parties, computes the final evaluation of the PRF on the given input $\mathbf{x}$ as follows: $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \sum_{i=1}^{T} f_{\mathbf{k_i}}(\mathbf{x}) \rceil_p$.*

Remark. The construction $\mathsf{PQDPRF}_{\mathsf{T,T}}$ requires a *two-layered rounding*; first from modulo q to q_1 during partial evaluation, and then from modulo q_1 to p during final evaluation.

3.2.1 Proof of Correctness and Consistency.

Here, we formally prove the correctness of our proposed $\mathsf{PQDPRF}_{\mathsf{T,T}}$. Let us express direct and distributed PRF evaluation as

$$f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) = \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k} \rangle \rceil_p, \quad f_{\{\mathbf{k_i}\}_{i \in [T]}}^{\mathsf{dist}}(\mathbf{x}) = \lfloor \sum_{i=1}^{T} \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k_i} \rangle \rceil_{q_1} \rceil_p.$$

Claim 1. *The difference between direct PRF evaluation and distributed PRF evaluation on some input x is strictly upper bounded by 1 with high probability, i.e.,*

$$\left| f_{\{\mathbf{k_i}\}_{i \in [T]}}^{\mathsf{dist}}(\mathbf{x}) - f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) \right| < 1.$$

Proof. Note that by definition of round-off operation (Section 2.1), for any $y \in \mathbb{Z}_q$, we can express $\lfloor y \rceil_p$ as $\frac{p}{q} y + e$, where $-0.5 \leq e < 0.5$.

Now, assuming $\mathbf{r} = \mathcal{H}(\mathbf{x})$, the direct evaluation can be written as,

$$f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p = \frac{p}{q} \langle \mathbf{r}, \mathbf{k} \rangle + e',$$

where $-0.5 \leq e' < 0.5$. The distributed evaluation can be expressed as,

$$f^{\mathsf{dist}}_{\{\mathbf{k_i}\}_{i\in[T]}}(\mathbf{x}) = \lfloor \sum_{i=1}^{T} \lfloor \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1} \rceil_p = \lfloor \sum_{i=1}^{T} (\langle \mathbf{r}, \mathbf{k_i} \rangle \cdot \frac{q_1}{q} + e_i) \rceil_p$$
$$= \lfloor \frac{q_1}{q} \sum_{i=1}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle + \sum_{i=1}^{T} e_i \rceil_p \leq \lfloor \frac{q_1}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{T}{2} \rceil_p$$
$$= \frac{p}{q_1} (\frac{q_1}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{T}{2}) + e = \frac{p}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{p}{q_1} \cdot \frac{T}{2} + e,$$

where, each $-0.5 \leq e_i < 0.5$ and $-0.5 \leq e < 0.5$. Thus,

$$f^{\mathsf{dist}}_{\{\mathbf{k_i}\}_{i\in[T]}}(\mathbf{x}) - f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x}) \leq \frac{p}{q_1} \cdot \frac{T}{2} + e - e'$$
$$\implies \left| f^{\mathsf{dist}}_{\{\mathbf{k_i}\}_{i\in[T]}}(\mathbf{x}) - f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x}) \right| \leq \left| \frac{p}{q_1} \cdot \frac{T}{2} \right| + |e - e'|$$

As $e, e' \in [-0.5, 0.5)$, $|e - e'| < 1$ holds true. Subsequently in Section 3.4 we discuss choice of values p, q_1 such that $\epsilon = \frac{p}{q_1} \cdot \frac{T}{2}$ is small enough and $|e - e'| + \epsilon$ does not exceed 1. □

Now, as both $f^{\mathsf{dist}}_{\{\mathbf{k_i}\}_{i\in[T]}}(\mathbf{x})$ and $f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x})$ are integers, we can conclude that their values are same except with a very small probability. Thus, the correctness of our proposed distributed PRF is satisfied.

Please note that in the case of (T, T)-distributed PRF, $\mathbb{A}_{T,T}$ is a singleton set, and hence, the consistency of $\mathsf{PQDPRF}_{\mathsf{T,T}}$ is trivially satisfied.

3.2.2 Proof of Security.

We recall the definition of security for a DPRF in the random oracle model in Section 2.3. We provide the formal statement on the security of the proposed DPRF in the following.

Theorem 3. *Our proposed* $\mathsf{PQDPRF}_{\mathsf{T,T}}$ *is secure if the underlying PRF is secure.*

Proof Overview. The underlying PRF, described in Section 3.1, is secure based on the LWR assumption (see Theorem 1). The hardness of the LWR problem is argued in Theorem 2 from the hardness of LWE problem. In the proposed DPRF, as described in Construction 1, we rely on the hardness of the same LWR instance, on which the underlying PRF relies. As the construction is in a (T, T)-threshold scenario, we follow the security notion of Section 2.3 and assume maximal corruption by the adversary. In other words, we assume a PPT adversary that has corrupted $(T - 1)$ parties and gained access to their key shares. Apart from the actual PRF evaluation for each query input, it is also allowed to see the partial evaluations of the honest parties for each queried input. The crux of the proof of security for the proposed DPRF (over and above the security of the underlying PRF) is to prove that the partial evaluation of an honest party does not leak any meaningful information about the honest party's key share. To prove this, we propose a strategy to simulate the partial evaluation of the honest party without using its actual key share, and then show that the distributions of the real and simulated partial evaluations are statistically indistinguishable.

Proof of Theorem 3. We define four hybrids Hybrid_0, Hybrid_1, Hybrid_2 and Hybrid_3 (see the hybrid diagrams in the following page), such that in each hybrid, the game is between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We assume $\mathcal{P} = \{P_1, \ldots, P_T\}$ to be a set of T parties. The adversary $\mathcal{A}$ has corrupted $(T-1)$ parties among them (say, $P_2, \ldots, P_T$ without loss of generality). So, P_1 is the only honest party, and its key share $\mathbf{k_1}$ is unknown to the adversary. Each hybrid consists of a query phase and a challenge phase. Only a priori bounded number of queries are allowed in the query phase, whereas the challenge phase consists of a single challenge. In the first hybrid (Hybrid_0), the adversary sees the actual partial evaluations in the query phase,

Query Phase	Challenge Phase
The adversary $\mathcal{A}$ sends a query input $\mathbf{x}$ to the challenger $\mathcal{C}$. In response, $\mathcal{C}$ sends $f_{\mathbf{k}}(\mathbf{x})$ and $f_{\mathbf{k_1}}(\mathbf{x})$ to $\mathcal{A}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \rceil_{q_1}}$.	$\mathcal{A}$ sends a challenge input $\mathbf{x}^\star$. In response, it receives $\boxed{f_{\mathbf{k}}(\mathbf{x}^\star)}$ from $\mathcal{C}$.

Hybrid_0

Query Phase	Challenge Phase
$\mathcal{A}$ sends a query input $\mathbf{x}$ to $\mathcal{C}$. In response, $\mathcal{C}$ sends $f_{\mathbf{k}}(\mathbf{x})$ and $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x})$ to $\mathcal{A}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x}) = \lfloor f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} \rceil - \sum_{i=2}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1}}$.	The adversary sends a challenge input $\mathbf{x}^\star$. In response, it receives $\boxed{f_{\mathbf{k}}(\mathbf{x}^\star)}$ from the challenger.

Hybrid_1

Query Phase	Challenge Phase
$\mathcal{A}$ sends a query input $\mathbf{x}$ to $\mathcal{C}$. In response, $\mathcal{C}$ sends $f_{\mathbf{k}}(\mathbf{x})$ and $f_{\mathbf{k_1}}(\mathbf{x})$ to $\mathcal{A}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \rceil_{q_1}}$.	$\mathcal{A}$ sends a challenge input $\mathbf{x}^\star$. In response, it receives from $\mathcal{C}$ $\boxed{y^\star \xleftarrow{R} \mathbb{Z}_p}$.

Hybrid_3

Query Phase	Challenge Phase
The adversary sends a query input $\mathbf{x}$ to the challenger. In response, the challenger sends $f_{\mathbf{k}}(\mathbf{x})$ and $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x})$ to the adversary, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x}) = \lfloor f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} \rceil - \sum_{i=2}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1}}$.	$\mathcal{A}$ sends a challenge input $\mathbf{x}^\star$. In response, it receives from $\mathcal{C}$ $\boxed{y^\star \xleftarrow{R} \mathbb{Z}_p}$.

Hybrid_2

and in the challenge phase, it sees the actual DPRF evaluation on challenge input. In the last hybrid (Hybrid_3), the adversary still sees the actual partial evaluations in the query phase, but in the challenge phase, it sees a truly random value. We aim to prove indistinguishability of Hybrid_3 from Hybrid_0, such that the adversary can not distinguish the output of the proposed DPRF and the output of a truly random function for the challenge input $\mathbf{x}^\star$ even in the presence of direct PRF evaluation and partial evaluation by the honest party (P_1) on a bounded number

of uniform random query inputs. For the sake of argument, we introduce two intermediate hybrids. In Hybrid_1, the adversary sees simulated partial evaluations in the query phase, but in the challenge phase, it still sees the actual DPRF evaluation on the challenge input. In the next hybrid (Hybrid_2), the adversary keeps on seeing simulated partial evaluations in the query phase, whereas, in the challenge phase, it sees a truly random value.

We prove the indistinguishability between the chain of hybrids in the form of some lemmas, which, in turn, proves the theorem. □

Indistinguishability between chain of hybrids.

Lemma 1. Hybrid_1 *is statistically indistinguishable from* Hybrid_0.

Proof. These two hybrids differ in the query phase, as in the first case the adversary $\mathcal{A}$ sees the actual partial evaluation by the honest party, while in the second case, $\mathcal{A}$ sees the simulated partial evaluation.

Actual partial evaluation in Hybrid_0: $f_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \rceil_{q_1}$.

Simulated partial evaluation in Hybrid_1:

$$f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x}) = \lfloor f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} - \sum_{i=2}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1} = \lfloor \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p \cdot \frac{q}{p} - \sum_{i=2}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1}$$
$$= \lfloor \langle \mathbf{r}, \mathbf{k} \rangle - e - \sum_{i=2}^{T} \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1} = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle - e \rceil_{q_1}.$$

Now we will prove that, the error term e in the expression of $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x})$ can be rewritten as $\langle \mathbf{r}, \mathbf{k}' \rangle$ for some $\mathbf{k}' \in \{0,1\}^n$, such that,

$$f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle - e \rceil_{q_1} = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle - \langle \mathbf{r}, \mathbf{k}' \rangle \rceil_{q_1} = \lfloor \langle \mathbf{r}, \mathbf{k_1} - \mathbf{k}' \rangle \rceil_{q_1} = \lfloor \langle \mathbf{r}, \mathbf{k'_1} \rangle \rceil_{q_1},$$

which essentially has the same distribution as of $f_{\mathbf{k_1}}(\mathbf{x})$. Note that $e = \langle \mathbf{r}, \mathbf{k} \rangle - \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p \cdot \frac{q}{p}$ is non-zero with high probability and it is independent of $\mathbf{k_1}$. So, $\mathbf{k}'$ satisfying $e = \langle \mathbf{r}, \mathbf{k}' \rangle$ is independent of $\mathbf{k_1}$. Hence although, $\mathbf{k'_1} = \mathbf{k_1} - \mathbf{k}'$, $\mathbf{k'_1}$ and $\mathbf{k_1}$ are independent of each other.

Since actual partial evaluation $f_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \rceil_{q_1}$ in Hybrid_0 and simulated partial evaluation $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k'_1} \rangle \rceil_{q_1}$ in Hybrid_1 come from the same distribution, Hybrid_1 is statistically indistinguishable from Hybrid_0. □

What remains to be proved is the following claim.

Claim 2. *Given* $\mathbf{r} \in \mathbb{Z}_q^n$*, and an error term* $e \in \mathbb{Z}_q$*, e can be represented as* $\langle \mathbf{r}, \mathbf{k}' \rangle$ *for some* $\mathbf{k}' \in \{0,1\}^n$.

Proof. We recall a simple application of leftover hash lemma [AMP19, IN96, IZ89], which states that given an additive group G, and n elements of that group $g_1, \ldots, g_n$, an arbitrary subset sum of those group elements is statistically indistinguishable from a random group element $g \xleftarrow{R} G$. In other words, for a random $\mathbf{k} \xleftarrow{R} \{0,1\}^n$ with high entropy, $\sum_{i=1}^{n} k_i g_i$ is uniformly random over G, provided $n \geq 3 \log |G|$.

Analogously, $\mathbb{Z}_q$ is an additive group with q elements. For any $\mathbf{r} \in \mathbb{Z}_q^n$ and $\mathbf{k}' \in \{0,1\}^n$, $\langle \mathbf{r}, \mathbf{k}' \rangle$ represents a subset sum of group elements, which is uniformly random over $\mathbb{Z}_q$ by leftover hash lemma, for unknown $\mathbf{k}'$ and $n \geq 3 \log q$.

Now let us assume that e can not be represented as $\langle \mathbf{r}, \mathbf{k}' \rangle$, which implies that there exists at least one $e \in \mathbb{Z}_q$, that can not be produced from the subset sum $\sum_{i=1}^{n} k'_i r_i$, thus leading to the violation of leftover hash lemma.

Hence by contradiction we can say that for any $e \in \mathbb{Z}_q$, there exists a $\mathbf{k}'$, such that $\langle \mathbf{r}, \mathbf{k}' \rangle = e$. □

Lemma 2. Hybrid_2 *is computationally indistinguishable from* Hybrid_1.

Proof. The challenger $\mathcal{C}$ in both these hybrids, responds to a query input $\mathbf{x}$ with $f_{\mathbf{k}}(\mathbf{x})$ and $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x})$. As $f^{\mathsf{sim}}_{\mathbf{k_1}}(\mathbf{x})$ is not a function of $\mathbf{k_1}$, it is never able to leak any meaningful information about $\mathbf{k_1}$ to the adversary $\mathcal{A}$. Hence the problem of distinguishing Hybrid_2 from Hybrid_1 reduces to the problem of distinguishing the PRF output of the form $\lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k} \rangle \rceil_p$ from the output of a truly random function in the challenge phase, which is hard since the underlying PRF has already been discussed to be secure due to hardness of LWR problem (Theorem 1). Thus, we conclude that Hybrid_2 is indistinguishable from Hybrid_1. Also, since our DPRF relies on the hardness of the same LWR instance, on which the underlying PRF relies, no loss in security is incurred owing to the choice of LWR parameters for PQDPRF. □

Lemma 3. Hybrid_3 *is statistically indistinguishable from* Hybrid_2.

Proof. Lemma 2 implies the proof of this lemma. □

Finally, Lemma 1, Lemma 2, and Lemma 3 together establish the fact that Hybrid_0 is indistinguishable from Hybrid_3, which implies that, $\mathcal{A}$ can not distinguish PRF output from the output of a truly random function even after seeing actual partial evaluations of the honest party for bounded number of query inputs, thus completing the proof of Theorem 3. Hence, the proposed (T, T)-distributed PRF is secure.

3.3 Generalised (t, T)-Threshold PRF

Now we extend the protocol described in the previous section for a more general setting of (t, T)-threshold PRF, where $2 \leq t \leq T$.

Construction 2 (Post-quantum Secure (t, T)-distributed Pseudo Random Function ($\mathsf{PQDPRF}_{\mathsf{t,T}}$))**.** $\mathsf{PQDPRF}_{\mathsf{t,T}}$ *is a post-quantum secure (t, T)-threshold PRF, whose input space $\mathcal{X}$, key space $\mathcal{K}$ and range $\mathcal{Y}$ are same as of* $\mathsf{PQDPRF}_{\mathsf{T,T}}$. *Here $q, p \in \mathbb{N}$ are publicly known moduli of input and output space respectively, while $q_1 \in \mathbb{N}$ $(p < q_1 < q)$ is the publicly known modulus of partial evaluation. Let $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q^n$ be a hash function modeled as a random oracle. With the key $\mathbf{k} \in \mathcal{K}$, distributed beforehand among T parties of the set $\mathcal{P} = \{P_1, \ldots, P_T\}$ by some threshold secret sharing procedure, we expect any subset of $\mathcal{P}$ having size at least t to be able to collaboratively evaluate the PRF on a given input $\mathbf{x} \in \mathcal{X}$. $\mathbb{A}_{t,T} = \{A \subset \mathcal{P} : |A| = t\}$ is the threshold access structure with $|\mathbb{A}_{t,T}| = \binom{T}{t}$. Any set with cardinality more than t is not explicitly considered as a member of $\mathbb{A}_{t,T}$, because one can pick any t number of parties from that set and perform the threshold evaluation on a given input. Hence, it is redundant to keep $A \subset \mathcal{P}$ with $|A| > t$ as a member of $\mathbb{A}_{t,T}$.* $\mathsf{PQDPRF}_{\mathsf{t,T}}$ *consists of three PPT algorithms,*

$$\mathsf{PQDPRF}_{\mathsf{t,T}} = (\mathsf{PQDPRF}_{\mathsf{t,T}}.\mathsf{Setup}, \mathsf{PQDPRF}_{\mathsf{t,T}}.\mathsf{PartialEval}, \mathsf{PQDPRF}_{\mathsf{t,T}}.\mathsf{FinalEval}).$$

$\underline{\mathsf{PQDPRF}_{\mathsf{t,T}}.\mathsf{Setup}(1^\lambda, \mathbb{A}_{t,T})}$*: On input security parameter λ and the threshold access structure $\mathbb{A}_{t,T}$, a key $\mathbf{k} \xleftarrow{R} \mathcal{K}$ is generated first. Next, it is distributed among T parties with a threshold*

secret sharing scheme, as described in Section 2.4, after which, each $P_i \in \mathcal{P}$ has to store $\binom{T-1}{t-1}$ number of shares, each corresponding to one of the $\binom{T-1}{t-1}$ number of t-sized subset of $\mathcal{P}$, that P_i belongs to. Notice that the original key $\mathbf{k}$ is destroyed and stored nowhere once the threshold secret sharing is done.

$\underline{\mathsf{PQDPRF}_{\mathsf{t},\mathsf{T}}.\mathsf{PartialEval}(\mathbf{x}, P_i, \mathcal{P}')}$*: For a given input $\mathbf{x}$ and a valid t-sized subset $\mathcal{P}' \in \mathbb{A}_{t,T}$, party $P_i \in \mathcal{P}'$ partially evaluates the PRF with its own share $\mathbf{k_i}$ corresponding to the subset $\mathcal{P}'$ as $f_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k_i} \rangle \rceil_{q_1}$, and broadcasts it to other collaborating parties in $\mathcal{P}' \setminus P_i$.*

$\underline{\mathsf{PQDPRF}_{\mathsf{t},\mathsf{T}}.\mathsf{FinalEval}(\mathcal{P}', \{f_{\mathbf{k_i}}(\mathbf{x})\}_{i\in[T]})}$*: Each party $P_i \in \mathcal{P}'$ having its own partial evaluation and partial evaluations of rest of the $(t-1)$ parties, computes the final evaluation of the PRF on the given input $\mathbf{x}$ as $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \sum_{P_i \in \mathcal{P}'} c_i f_{\mathbf{k_i}}(\mathbf{x}) \rceil_p$. Here, c_i's are the recovery coefficients and according to the threshold secret sharing scheme, described in Section 2.4, recovery coefficient of the* group_leader *(the party in $\mathcal{P}'$ with minimum* party_id*) is 1 and recovery coefficient of each of the other parties in $\mathcal{P}' \setminus P_i$ is -1.*

3.3.1 Proof of Correctness and Consistency.

Correctness of $\mathsf{PQDPRF}_{\mathsf{t},\mathsf{T}}$ can be proved essentially in the same way as of $\mathsf{PQDPRF}_{\mathsf{T},\mathsf{T}}$, which in turn implies its consistency. Let $\mathcal{P} = \{P_i\}_{i\in[T]}$ be a set of T parties, and $\mathbb{A}_{t,T}$ be a threshold access structure defined on it. Without loss of generality let us consider $\mathcal{P}' = \{P_1, \ldots, P_t\} \in \mathbb{A}_{t,T}$ to be a valid t-sized subset. Clearly P_1 is the group_leader of $\mathcal{P}'$. Let $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q^n$ be a hash function modeled as a random oracle. Given an input $\mathbf{x}$, let us denote the direct PRF evaluation with $f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x})$ and distributed PRF evaluation by t number of parties in $\mathcal{P}'$ using $\mathsf{PQDPRF}_{\mathsf{t},\mathsf{T}}$ as $f^{\mathsf{dist}}_{\mathbf{k}}(\mathbf{x})$. They are computed as follows.

$$f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k} \rangle \rceil_p, \quad f^{\mathsf{dist}}_{\mathbf{k}}(\mathbf{x}) = \lfloor \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k_1} \rangle \rceil_{q_1} - \sum_{i=2}^{t} \lfloor \langle \mathcal{H}(\mathbf{x}), \mathbf{k_i} \rangle \rceil_{q_1} \rceil_p.$$

Claim 3. *Difference between direct PRF evaluation ($f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x})$) and distributed PRF evaluation ($f^{\mathsf{dist}}_{\mathbf{k}}(\mathbf{x})$) on some input $\mathbf{x}$ is strictly upper bounded by 1, i.e.,*

$$\left| f^{\mathsf{dist}}_{\mathbf{k}}(\mathbf{x}) - f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x}) \right| < 1.$$

Proof. Note that by definition of round-off operation (Section 2.1), for any $y \in \mathbb{Z}_q$, we can express $\lfloor y \rceil_p$ as $\frac{p}{q}y + e$, where $-0.5 \leq e < 0.5$.

We assume $\mathbf{r} = \mathcal{H}(\mathbf{x})$. Now the direct evaluation can be written as,

$$f^{\mathsf{dir}}_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p = \frac{p}{q} \langle \mathbf{r}, \mathbf{k} \rangle + e',$$

where $|e'| \leq 0.5$. The distributed evaluation can be expressed as,

$$\begin{aligned}
f_{\mathbf{k}}^{\mathsf{dist}}(\mathbf{x}) &= \lfloor \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \rceil_{q_1} - \sum_{i=2}^{t} \lfloor \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1} \rceil_p \\
&= \lfloor \langle \mathbf{r}, \mathbf{k_1} \rangle \cdot \frac{q_1}{q} + e_1 - \sum_{i=2}^{t} (\langle \mathbf{r}, \mathbf{k_i} \rangle \cdot \frac{q_1}{q} + e_i) \rceil_p \\
&= \lfloor \frac{q_1}{q} (\langle \mathbf{r}, \mathbf{k_1} \rangle - \sum_{i=2}^{t} \langle \mathbf{r}, \mathbf{k_i} \rangle) + (e_1 - \sum_{i=2}^{t} e_i) \rceil_p \\
&\leq \lfloor \frac{q_1}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{t}{2} \rceil_p \\
&= \frac{p}{q_1} (\frac{q_1}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{t}{2}) + e \\
&= \frac{p}{q} \langle \mathbf{r}, \mathbf{k} \rangle + \frac{p}{q_1} \cdot \frac{t}{2} + e,
\end{aligned}$$

where, each $-0.5 \leq e_i < 0.5$ and $-0.5 \leq e < 0.5$ due to the definition of the round-off operation. Thus,

$$f_{\mathbf{k}}^{\mathsf{dist}}(\mathbf{x}) - f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) \leq \frac{p}{q_1} \cdot \frac{t}{2} + e - e' \implies \left| f_{\mathbf{k}}^{\mathsf{dist}}(\mathbf{x}) - f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) \right| \leq \left| \frac{p}{q_1} \cdot \frac{t}{2} \right| + |e - e'| .$$

We choose values of p, t and q_1 such that, the quantity $\epsilon = \frac{p}{q_1} \cdot \frac{t}{2}$ becomes sufficiently small. As $-0.5 \leq e < 0.5$ and $-0.5 \leq e' < 0.5$, $|e - e'| < 1$ always holds true. Thus, the quantity $\epsilon + |e - e'|$ is highly unlikely to exceed 1. Hence, the difference between direct PRF evaluation and distributed PRF evaluation is strictly upper bounded by 1, i.e.,

$$\left| f_{\mathbf{k}}^{\mathsf{dist}}(\mathbf{x}) - f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x}) \right| < 1.$$

□

As both $f_{\mathbf{k}}^{\mathsf{dist}}(\mathbf{x})$ and $f_{\mathbf{k}}^{\mathsf{dir}}(\mathbf{x})$ are integers, we conclude that their values are same except with a very small probability. With our choice of parameters, described in Section 3.4, a simple calculation shows that the probability of incorrect DPRF evaluation is upper bounded by 2^{-27}. Thus, the correctness of our proposed distributed PRF $\mathsf{PQDPRF}_{\mathsf{t,T}}$ is satisfied. Since the correctness of distributed PRF implies its consistency, the proposed $\mathsf{PQDPRF}_{\mathsf{t,T}}$ is also consistent with very high probability.

3.3.2 Proof of Security.

The idea of the proof remains essentially the same as the proof of security of (T, T)-distributed PRF (Section 3.2.2). We provide the detailed proof of security of $\mathsf{PQDPRF}_{\mathsf{t,T}}$ in the following.

The idea of the proof remains essentially the same as of proof of security of (T, T)-distributed PRF (Section 3.2.2). However among T parties of the set $\mathcal{P} = \{P_1, \ldots, P_T\}$, only t parties are required to collaborate to evaluate the PRF on a given input. We assume a PPT adversary $\mathcal{A}$ which has corrupted a subset $\mathcal{P}_\mathcal{C} \subset \mathcal{P}$ of size $(t-1)$ and thus acquired all their key shares. We show that, even if $\mathcal{A}$ is able to see the PRF evaluation and all the partial evaluations of the honest parties in $\mathcal{P} \setminus \mathcal{P}_\mathcal{C}$ for a priori bounded number of query inputs, it will not be able to

Query Phase	Challenge Phase
$\mathcal{A}$ sends a query input $\mathbf{x}$ to $\mathcal{C}$. Then, $\mathcal{C}$ responds with $f_{\mathbf{k}}(\mathbf{x})$ and $\{f_{\mathbf{k_i}}(\mathbf{x})\}_{P_i \in \mathcal{P} \setminus \mathcal{P}_\mathcal{C}}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1}.}$	$\mathcal{C}$, on receiving a challenge input $\mathbf{x}^\star$ from $\mathcal{A}$, responds with $\boxed{f_{\mathbf{k}}(\mathbf{x}^\star).}$

$\mathsf{Hybrid}_\mathbf{0}$

Query Phase	Challenge Phase
On receiving query input $\mathbf{x}$ from $\mathcal{A}$, $\mathcal{C}$ responds with PRF evaluation $f_{\mathbf{k}}(\mathbf{x})$ and simulated partial evaluations for the honest parties $\{f^{\mathbf{sim}}_{\mathbf{k_i}}(\mathbf{x})\}_{P_i \in \mathcal{P} \setminus \mathcal{P}_\mathcal{C}}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, if $i == \mathsf{gl}$, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_i}}(\mathbf{x}) = \lfloor f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} + \sum_{P_j \in \mathcal{P}_\mathcal{C}} \langle \mathbf{r}, \mathbf{k_j} \rangle \rceil_{q_1},}$ otherwise, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_{gl}} \rangle - f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} - \sum_{P_j \in \mathcal{P}_\mathcal{C}, j \neq \mathsf{gl}} \langle \mathbf{r}, \mathbf{k_j} \rangle \rceil_{q_1}.}$ In the above expression, $\mathbf{k_j}$ is the key share of $P_j \in \mathcal{P}_\mathcal{C}$ corresponding to t-sized group $\{P_i\} \bigcup \mathcal{P}_\mathcal{C}$.	$\mathcal{C}$, on receiving a challenge input $\mathbf{x}^\star$ from $\mathcal{A}$, responds with $\boxed{f_{\mathbf{k}}(\mathbf{x}^\star).}$

$\mathsf{Hybrid}_\mathbf{1}$

Query Phase	Challenge Phase
$\mathcal{A}$ sends a query input $\mathbf{x}$ to $\mathcal{C}$. $\mathcal{C}$ responds with $f_{\mathbf{k}}(\mathbf{x})$ and $\{f_{\mathbf{k_i}}(\mathbf{x})\}_{P_i \in \mathcal{P} \setminus \mathcal{P}_\mathcal{C}}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, $\boxed{f_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_i} \rangle \rceil_{q_1}.}$	$\mathcal{C}$, on receiving a challenge input $\mathbf{x}^\star$ from $\mathcal{A}$, responds with a random $\boxed{y^\star \xleftarrow{R} \mathbb{Z}_p.}$

$\mathsf{Hybrid}_\mathbf{3}$

Query Phase	Challenge Phase
On receiving query input $\mathbf{x}$ from $\mathcal{A}$, $\mathcal{C}$ responds with PRF evaluation $f_{\mathbf{k}}(\mathbf{x})$ and simulated partial evaluations for the honest parties $\{f^{\mathbf{sim}}_{\mathbf{k_i}}(\mathbf{x})\}_{P_i \in \mathcal{P} \setminus \mathcal{P}_\mathcal{C}}$, where $\mathbf{r} = \mathcal{H}(\mathbf{x})$, $f_{\mathbf{k}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k} \rangle \rceil_p$, if $i == \mathsf{gl}$, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_i}}(\mathbf{x}) = \lfloor f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} + \sum_{P_j \in \mathcal{P}_\mathcal{C}} \langle \mathbf{r}, \mathbf{k_j} \rangle \rceil_{q_1},}$ otherwise, $\boxed{f^{\mathsf{sim}}_{\mathbf{k_i}}(\mathbf{x}) = \lfloor \langle \mathbf{r}, \mathbf{k_{gl}} \rangle - f_{\mathbf{k}}(\mathbf{x}) \cdot \frac{q}{p} - \sum_{P_j \in \mathcal{P}_\mathcal{C}, j \neq \mathsf{gl}} \langle \mathbf{r}, \mathbf{k_j} \rangle \rceil_{q_1}.}$ In the above expression, $\mathbf{k_j}$ is the key share of $P_j \in \mathcal{P}_\mathcal{C}$ corresponding to t-sized group $\{P_i\} \bigcup \mathcal{P}_\mathcal{C}$.	$\mathcal{C}$, on receiving a challenge input $\mathbf{x}^\star$ from $\mathcal{A}$, responds with a random $\boxed{y^\star \xleftarrow{R} \mathbb{Z}_p.}$

$\mathsf{Hybrid}_\mathbf{2}$

distinguish output of the PRF from the output of a truly random function on a challenge input, which is essentially different from the query inputs.

Recall that, after $\mathsf{PQDPRF}_{\mathsf{t},\mathsf{T}}.\mathsf{Setup}$, each $P_i \in \mathcal{P}$ gets to store $\binom{T-1}{t-1}$ number of secret shares, each corresponding to one of the t-sized subsets, that P_i may belong to. In each of the hybrids, if $P_i \in \mathcal{P} \setminus \mathcal{P}_{\mathcal{C}}$ is a honest party, we denote by $\mathbf{k_i}$ its key share corresponding to the t-sized group $\{P_i\} \bigcup \mathcal{P}_{\mathcal{C}}$, and by gl, the $\mathsf{group\_leader}$ of $\{P_i\} \bigcup \mathcal{P}_{\mathcal{C}}$.

Now we define four hybrids consisting of game between the PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as described in the tabular forms for the ease of exposition.

Indistinguishibility between the hybrids.
The indistinguishibility of Hybrid_3 from Hybrid_0 for (t, T)-distributed PRF can be proved analogously as done in Section 3.2.2 for (T, T)-distributed PRF. Please see the detailed hybrids on the next page.

3.4 Choice of Parameters

The security of our proposed DPRF directly relies on the security of underlying PRF (Theorem 3), which in turn relies on the hardness of decision-LWR problem (Theorem 1). Hence, we need to find a suitable set of parameters n, p, q for which LWR problem is hard. While an approach for determining LWR parameters (n, p, q) from LWE parameters (n, q, α) (where α is the rate of Gaussian LWE noise) could be to follow the theoretical analysis of LWE-to-LWR reduction maintaining polynomial "$\frac{q}{p}$" ratio with a priori bounded number of samples (Theorem 2), we find the line of works in [DKSRV18, CKLS18] more suitable to choose LWR parameters for practical instantiations. While analyzing the concrete hardness of LWR problem, they convert the LWR samples to LWE samples by multiplying the sample with $\frac{q}{p}$ and then analyze the cost of known attacks to solve LWE. The works [DKSRV18, ADPS16, CKLS18, BCD$^+$16] based on LWE or LWR assumption considers only primal and dual attacks as number of samples(m) in their case is at most $2n$. However, there are several known attacks against LWE [APS15] and none of the attacks on LWR exploits the theoretical gap between the hardness of LWE and LWR problem to perform better than an attack on LWE. So, we focus on finding parameters (n, q), such that for uniform noise in modulo $\frac{q}{q_1}$, corresponding LWE problem is hard provided a certain number of samples, which is equivalent to the allowed number of queries in the query phase of security game of DPRF. Note that *lattice estimator*[1] [APS15] evaluates the hardness of LWE problem for a given set of parameters based on its resistance against all practical attack methods. We use *lattice estimator* to find a suitable LWE parameter choice $n = 1024, q = 2^{64}$ with secret distribution being uniform over $\mathbb{Z}_q^n$, and LWE noise distribution being uniform over $\mathbb{Z}_{\frac{q}{q_1}}$, such that all the known attack methods have run time more than 2^{128}, which indicates that these parameter choices provide at least 128 bits of security. We accordingly choose our LWR parameters $n = 1024, q = 2^{64}, q_1 = 2^{42}$.

Parameter	**Value**
Modulus of input space ($\mathcal{X}$) of PQDPRF (q)	2^{64}
Modulus of partial evaluation space (q_1)	2^{42}
Modulus of output space ($\mathcal{Y}$) of PQDPRF (p)	2^{10}
Dimension of key in PQDPRF (n)	1024

Table 1: Parameters used in PQDPRF implementation

[1] `https://github.com/malb/lattice-estimator`

Now, given q and q_1, what remains is to choose a suitable value of modulus p ($p < q_1 < q$) such that $\frac{p}{q_1}$ is sufficiently small to ensure the correctness of our proposed DPRF. We observe that choosing $p = 2^{10}$, (while $q_1 = 2^{42}$) makes the value of $\frac{p}{q_1}$ sufficiently small. Note that the ratio $\frac{q}{p} = 2^{54}$ is still polynomial in security parameter $\lambda = 128$.

Finally, we provide our concrete choice of parameters for the proposed DPRF in Table 1. We continue using this set of parameters while using it in $\mathsf{PQ-DiSE}$.

3.5 Proposed PQDPRF vs. the Lattice-based DPRF in [LST21]

A robust construction of lattice-based distributed PRF in adaptive corruption settings with theoretically efficient parameters in the standard model was proposed in [LST21], which builds upon LWE-to-LWR reductions preserving polynomial large modulus-to-noise ratios [AKPW13, BGM+16]. On the contrary, our construction is in the random oracle model and is targeted for semi-honest settings against static corruption. Hence, a direct experimental comparison is not feasible. However, the fact that our construction is in the random oracle model makes it more efficient, and hence, more suitable for real-world applications.

We compare the overheads of a single DPRF evaluation in [LST21] vs a single DPRF evaluation in our case for the same LWR parameters (n, q, p). In [LST21] the PRF evaluation assumes a L-bit input and the evaluation (see Eq 2 of Section 3.2) requires (i) L matrix multiplications with each matrix in $\mathbb{Z}_q^{m\times m}$, which needs a $O(\log L)$-depth circuit with $\omega(m^2)$ field operations per matrix multiplication (leading to a total cost of $L \cdot \omega(m^2)$ field operations), (ii) A matrix multiplication between two matrices of dimension $n \times m$ and $m \times m$ respectively (leading to a cost of $\omega(mn)$ field operations), (iii) A matrix-vector multiplication where the matrix has dimension $n \times m$ and the vector has dimension n (leading to a cost of $O(mn)$ field operations). So, the overall cost of single PRF evaluation in [LST21] is $L \cdot \omega(m^2)$. On the other hand a single PRF evaluation in our case (see Section 3.1) requires multiplying two vectors of $\mathbb{Z}_q^n$ which only costs $O(n)$ field operations. In the case of the DPRFs obtained by distributing the evaluation of the above PRFs, the above cost analysis still applies for a single partial evaluation done by each of the parties.

We now present a back-of-the-envelope calculation to compare these overheads for typical parameters used in practical applications (e.g., $n = 1024$, $q = 2^{64}$ for LWR hardness, an input length of $L = n \log q$ and dimension $m = 2n \log q$). In this case, the number of field operations required for a single PRF evaluation (equivalently, a single DPRF partial evaluation) in [LST21] is at least $10^{12}\times$ larger than that for our construction. This clearly establishes that our construction is practically more efficient.

We defer the performance comparison of the proposed DPRF with other (more practically efficient) existing DPRFs (namely the AES-based DPRF and the quantum-broken DDH-based DPRF) till Section 5, where the experimental results are provided in the context of an application (DiSE).

4 Application

The LWR-based distributed PRF, that we propose and discuss in detail in the previous section, can be plugged into various real-world applications of distributed PRF. In this work, we particularly focus on the DiSE (Distributed Symmetric Encryption) protocol, originally proposed

in [AMMR18] and view it as an application of DPRF. We validate our proposed LWR-based DPRF by using it in DiSE to make it quantum-safe and call it $\mathsf{PQ-DiSE}$. For the sake of exposition, we dedicate one subsection below to recall the original DiSE protocol of [AMMR18] and then discuss the proposed $\mathsf{PQ-DiSE}$ in the following subsection.

4.1 An Overview of the DiSE Protocol

Like any other encryption scheme, the distributed symmetric-key encryption (DiSE) scheme also consists of three PPT algorithms: (i) Setup, (ii) Encrypt and (iii) Decrypt, but with a difference that, both Encrypt and Decrypt are distributed, i.e., encryption and decryption are performed by, instead of a single server, a number of servers in distributed manner. In a (t, T)-DiSE, any $t(< T)$ parties among the T parties are contacted with a request of encryption or decryption and each of them contributes some partially computed values, which are then combined in order to get the end result of encryption or decryption.

Definition 2 (Distributed Symmetric-key Encryption (DiSE))**.** *Let $\mathcal{P} = \{P_i\}_{i\in[T]}$ be the set of parties/servers to perform DPRF evaluation.* DiSE *protocol internally uses the following cryptographic primitives as its building blocks:*
(i) A DPRF $\mathsf{DP} = (\mathsf{DP.Setup}, \mathsf{DP.PartialEval}, \mathsf{DP.FinalEval})$*,*
(ii) A PRG (pseudo random generator) of polynomial stretch,
(iii) A commitment scheme $\mathsf{C} = (\mathsf{C.Setup}, \mathsf{C.Com})$*.*

DiSE *consists of the following three protocols built over these primitives,*

$\underline{\mathsf{DiSE.Setup}(1^\lambda, t, T)}$*:* $\mathsf{DP.Setup}(1^\lambda, t, T)$ *is executed to provide evaluation key shares ek_i to P_i $\forall i \in [T]$. Also* $\mathsf{C.Setup}(1^\lambda)$ *outputs public parameters* pp_{com}*.*

$\underline{\mathsf{DiSE.DistEncrypt}(m, S, \{ek_i\}_{P_i\in S})}$*: An entity* E *requiring encryption of plaintext m follows the method below.*

- E *contacts a set $S \subset \mathcal{P}$ of servers, such that $|S| = t$ and provides them with $\alpha = \mathsf{C.Com}(m, pp_{\mathsf{com}}; \rho)$, where ρ is randomness used in commitment.*
- *Now $z_i = \mathsf{DP.PartialEval}(\alpha, P_i, S)$ is generated parallelly by each $P_i \in S$ with its evaluation key share ek_i and sent back to* E*.*
- E *now computes $w = \mathsf{DP.FinalEval}(S, \{z_i\}_{P_i\in S})$ and then $e = \mathsf{PRG}(w) \bigoplus (m||\rho)$. Finally $c = (\alpha, e)$ is the ciphertext of m. Here, w can be viewed as the message-specific encryption key.*

$\underline{\mathsf{DiSE.DistDecrypt}(c, S, \{ek_i\}_{P_i\in S})}$*: Distributed decryption of a ciphertext c is performed by an entity* D *as follows.*

- D *parses c into (α, e) and contacts a set $S \subset \mathcal{P}$ of t servers and provides them with α.*
- *Each $P_i \in S$ computes $z_i = \mathsf{DP.PartialEval}(\alpha, P_i, S)$ with its evaluation key share and sends it to* D*.*
- D *combines the z_i's to retrieve $w = \mathsf{DP.FinalEval}(S, \{z_i\}_{P_i\in S})$. Next $e \bigoplus \mathsf{PRG}(w)$ gives back $m||\rho$. It then checks if α is indeed a commitment to m with randomness ρ. If it is, then m is returned as the result of distributed decryption.*

Relation Between DiSE and the Underlying DPRF DP. We recall a theorem from [AMMR18] in order to better understand how the security of DiSE depends upon the underlying distributed PRF (assume that other two primitives PRG and C are already secure).

Theorem 4. *DiSE is secure if the underlying DPRF DP is secure.*

Informally, the security of a DPRF DP implies that its output retains pseudorandomness even when evaluated in a distributed manner (See Section 2.3 for formal security notion). The interesting part of the theorem is that the security of the underlying DPRF DP directly implies the security of the distributed encryption scheme.

Instantiations of DP. DiSE [AMMR18] use the following two instantiations of DP for semi-honest settings:

- **DDH-based DPRF**: Proposed in [NPR99], DDH-based DPRF is secure due to the hardness of the classical DDH problem. It uses Shamir's secret sharing scheme for sharing the PRF evaluation key among the servers. However, it is vulnerable to quantum attack.
- **AES-based DPRF**: A general construction of DPRF from any existing PRF was proposed in [NPR99]. DiSE uses AES-based DPRF accordingly and proves it to be secure. It uses replicated secret sharing to share the evaluation key among the T servers. Although AES(256)-based DPRF is widely believed to be quantum-resilient, it is also not built upon any quantum-safe assumption.

The paper [AMMR18] compares performances of both these instantiations and concludes that DiSE performs well with AES-based DPRF for lesser values of T. Note that none of the underlying DPRF is inherently quantum-safe.

4.2 Our Improved PQ − DiSE Protocol

As security of DiSE directly depends upon the security of the underlying DP (Theorem 4), we obtain post-quantum secure version of DiSE (i.e., PQ − DiSE) by instantiating the underlying DP with our proposed post-quantum secure PQDPRF. Our implementation of PQ − DiSE is publicly available here[1].

Technical challenges of PQ − DiSE implementation. DPRF implementation in original DiSE generates 128-bit DPRF output from 128-bit input, whereas our proposed DPRF generates $\log p = 10$ bit output from $n \cdot \log q = 1024 \times 64$ bit input. We face two-fold challenge here: (i) converting 128-bit input to 1024×64-bit input in order to apply PQDPRF, and (ii) generating a total of 128 pseudorandom bits in the output. The first challenge is overcome by applying hash function on the input concatenated with a counter value repeatedly until the length of these concatenated hash outputs equals 1024×64 bits. The next challenge is handled by running 13 instances of PQDPRF together in order to obtain $(13 \times 10) = 130$ bits and extract 128 bits as the message-specific encryption key to be used later. We use Blake2[2] to instantiate the hash function, modeled as a random oracle.

An analysis on the key and key shares. Table 2 provides a comparative analysis on the size of secret key and key shares with respect to the three DPRF instantiations. Even with a larger key-size requirement, our proposed LWR-based DPRF outperforms the other two due to its highly parallelizable nature, as evident from the results in the next section.

[1] https://github.com/SayaniSinha97/PQDiSE-from-PQDPRF

[2] https://www.blake2.net/

DPRFs	Size of secret key (as well as each key share)	Total number of unique key shares	Number of key shares that each party stores	Secret sharing method
AES-based	128-bits	$\binom{T}{t-1}$	$\binom{T-1}{t-1}$	Replicated secret sharing
DDH-based	256-bits	T	1	Shamir's secret sharing
LWR-based (proposed)	**1024×64-bits**	$\boldsymbol{\binom{T}{t} \cdot t}$	$\boldsymbol{\binom{T-1}{t-1}}$	**Benaloh-Leichter LISSS**

Table 2: Comparison of key sizes for DPRFs

5 Experimental Result

We now provide a detailed performance analysis of our proposed PQDPRF in PQ – DiSE based on various metrics with respect to DDH-based DPRF and AES-based DPRF, used in DiSE, all in semi-honest adversarial settings. All experiments have been executed on a high-end server with an Intel(R) Xeon(R) Gold 6226 CPU (2.70GHz clock frequency), 96 cores, 256GB RAM. All graphs have their y-axis in *logarithmic scale*. During performance evaluation, we disable the use of AES-NI instructions by AES-based DPRF to ensure fair comparison among software implementations of the three DPRFs. We optimize our LWR-based DPRF implementation that involves arithmetic in $\mathbb{Z}_q$ using NTL[1].

Partial evaluation time vs. (t, T) values [Figure 1]. In any (t, T)-distributed PRF scheme, the partial evaluation of the DPRF on a given input is computed parallelly by all t collaborating parties with their respective secret share. Here, we analyze the maximum partial evaluation time required by any of the t participating parties for all three DPRFs under consideration.

- **AES-based DPRF**: A linear increase in partial evaluation time in a logarithmic y-axis actually reflects an exponential increase in time. In AES-based DPRF, computation of partial evaluations by t parties involves all $\binom{T}{t-1}$ key shares; however, the load of computation is not evenly distributed among all t parties. In particular, the maximum computation time increases linearly with $\binom{T-1}{t-1}$.
- **DDH-based and LWR-based DPRF**: Partial evaluation time remains almost constant with increasing T for both these DPRFs. Because, in both cases, given an input, each of the t parties parallelly performs a similar computation with its own secret share. Hence, each of the t parties requires a similar time in computing partial evaluation. Thus, the maximum time taken by any party to complete the partial evaluation phase does not depend upon the value of t or T. The graph line of LWR-based DPRF lies slightly below the line of DDH-based DPRF due to the fact that the modular dot product of two vectors in $\mathbb{Z}_q^n$ takes less time than modular exponentiation.

Final evaluation time vs. (t, T) values [Figure 2]. This graph compares the three DPRFs in terms of final evaluation time required by them with varying T.

- The final evaluation time of combining t partial evaluations increases linearly with increasing

[1] `https://libntl.org/`

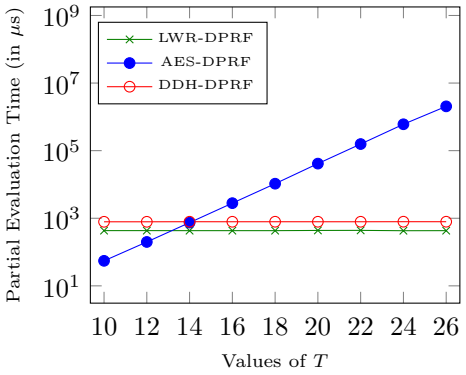

Figure 1: Partial evaluation time of $(\frac{T}{2}, T)$-DPRFs

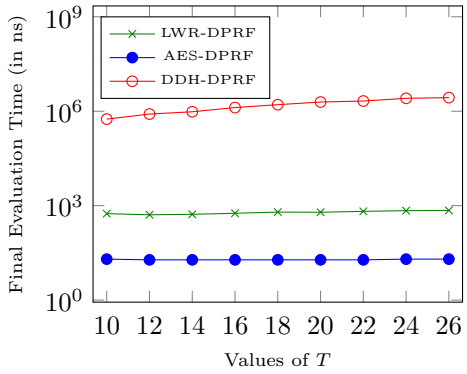

Figure 2: Final evaluation time of $(\frac{T}{2}, T)$-DPRFs

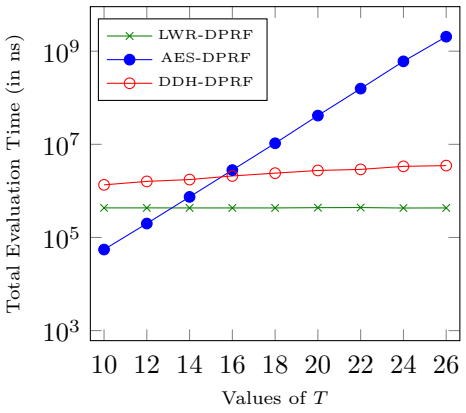

Figure 3: Total evaluation time of $(\frac{T}{2}, T)$-DPRFs

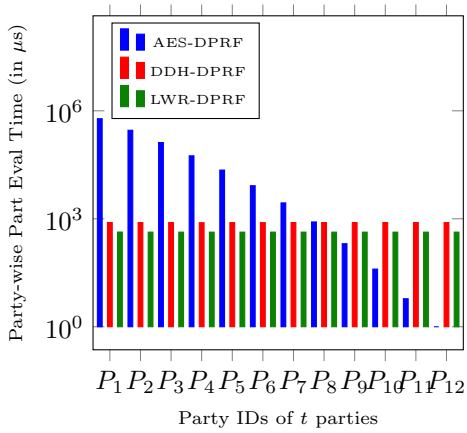

Figure 4: Individual part-eval time for $(12, 24)$-DPRF

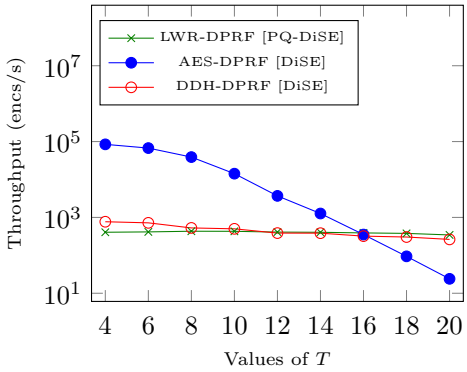

Figure 5: Throughput in PQ-DiSE and DiSE

value of t for all the three DPRFs due to the fact that, in final evaluation phase, LWR-based DPRF involves modular vector addition of t partial evaluations, whereas AES-based DPRF involves XORing of t partial evaluations. XORing, being a lighter operation than vector addition, places AES-based DPRF at a lower position in y-axis than LWR-based DPRF. DDH-based DPRF involves exponentiation and then multiplication of t partial evaluations, leading to its higher value along y-axis. Note that although the final evaluation time of the

proposed LWR-based DPRF is more than that of AES-based DPRF, we argue the efficiency of LWR-based DPRF considering the total (partial + final) evaluation time of both the DPRFs, as we discuss next.

Total evaluation time vs. (t, T) values [Figure 3]. This graph compares the three DPRFs in terms of total (partial + final) evaluation time required by them with varying T. Keeping Figure 1 and Figure 2 in mind, the plots here are quite self-explanatory and clearly depict the efficiency of the proposed LWR-based DPRF for larger values of T.

Partial evaluation time vs. party-id [Figure 4]. We plot in this graph the partial evaluation time taken by each of the t collaborating parties in a t-sized subset for a specific pair of values, $(t, T) = (12, 24)$.

- **AES-based DPRF**: As mentioned earlier, all the t parties here do not have the same amount of computation load during partial evaluation phase. Without loss of generality, if we denote the collaborating parties with $\{P_1, \ldots, P_t\}$, P_1 requires computation using $\binom{T-1}{t-1}$ key shares, P_2 requires $\binom{T-2}{t-2}$ key shares and so on. Finally P_t requires $\binom{T-t}{0} = 1$ key share in its partial evaluation, thus involving all $\binom{T}{t-1}$ key shares. Thus, each participating party has a different computation cost, as depicted in the graph.

- **DDH-based and LWR-based DPRF**: In both cases, each participating party needs only one key share for partial evaluation computation and involves the same modular dot product operation between vectors (LWR-based DPRF) or exponentiation operation (DDH-based DPRF) irrespective of its party-id. This feature can be useful while enabling parallel computation by all participating parties.

Throughput vs. (t, T) values [Figure 5]. We plot the throughput (number of encryptions per second) of DiSE using DDH-based and AES-based DPRF and PQ − DiSE using LWR-based DPRF.

- DiSE: When instantiated with AES-based DPRF, its throughput decreases with increasing value of t, T, but remains stable with increasing value of t, T if DDH-based DPRF is used.

- PQ − DiSE: Its throughput is stable for all values of t, T. It performs slightly better than DiSE using DDH-based DPRF and significantly better than DiSE using AES-based DPRF for larger values of t, T.

  Explanation: Our PQ − DiSE outperforms DiSE in terms of throughput owing to the fact that LWR-based DPRF outperforms AES-based and DDH-based DPRF in terms of evaluation cost as discussed in analysis of Figure 1.

Note: Although we provide the analysis with respect to $(\frac{T}{2}, T)$-distributed PRF, the graph patterns of Figure 1, 2, 4, 5 retain for any $1 < t \leq T$. However we prefer $(\frac{T}{2}, T)$-distributed PRF for the sake of analysis, as the value of $\binom{T}{t}$ is the largest for $t = \frac{T}{2}$.

Concluding Remark. AES-128 provides 128-bit classical security and 64-bit quantum security (against Grover's algorithm [Gro96]), which is also the security level for the AES-based DPRF implemented in DiSE. One could upgrade to AES-256 to provide stronger quantum security, but this would only degrade the performance of the AES-based DPRF further. The DDH-based DPRF provides 128 bits of classical security, and is quantum-broken. In contrast, our proposed DPRF uses an LWR parameter set that provides the quantum-equivalent of 128-bit classical security (as per the latest lattice estimator) but still outperforms AES-based DPRF for higher values of T and DDH-based DPRF slightly for all values of T.

6 Conclusion and Future Work

We proposed a (T, T)-distributed quantum-safe PRF based on Learning with Rounding (LWR) problem and its generalized (t, T)-distributed version in this work. We proved its correctness, consistency as well as security. We also showed how to use our proposed DPRF to obtain an efficient quantum-safe version of DiSE [AMMR18], namely PQ − DiSE. We outline some future research directions below.

- **Scalability with an even larger number of parties.** Our (t, T)-DPRF requires each party to store $\binom{T-1}{t-1}$ number of key shares after threshold secret sharing, thus suffering from high space complexity. Future works may consider modifying the linear integer secret sharing protocol in order to reduce space complexity and make the DPRF scalable for an even larger number of parties.

- **Adaptive security.** We assumed that the corrupted set of parties is statically fixed before the game begins between the challenger and the adversary. We leave it as an open problem to allow our DPRF to handle the scenario, where parties are corrupted dynamically during the game.

- **Security in Quantum Random Oracle Model.** We leave it as an open problem to prove the security of the proposed LWR-based distributed PRF in the quantum random oracle model, where the quantum adversary has quantum access to the random oracle and, thus is able to query the random oracle with a state in superposition.

References

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key {Exchange—A} new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, 2016.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 57–74, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[AMMR18] Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. Dise: distributed symmetric-key encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1993–2010, 2018.

[AMP19] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Symmetric primitives with structured secrets. In *Annual International Cryptology Conference*, pages 650–679. Springer, 2019.

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

[ASA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from lwe to lwr. *Cryptology ePrint Archive*, 2016.

[BCD+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1006–1018, 2016.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, 2011.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *Theory of Cryptography Conference*, pages 209–224. Springer, 2016.

[BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Annual Cryptology Conference*, pages 410–428. Springer, 2013.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *Annual Cryptology Conference*, pages 353–370. Springer, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012.

[BUK19] Xenia Bogomolec, John Gregory Underhill, and Stiepan Aurélien Kovac. Towards post-quantum secure symmetric cryptography: A mathematical perspective. *Cryptology ePrint Archive*, 2019.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions. In *Theory of Cryptography Conference*, pages 1–30. Springer, 2015.

[CGMS21] Mihai Christodorescu, Sivanarayana Gaddam, Pratyay Mukherjee, and Rohit Sinha. Amortized threshold symmetric-key encryption. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2758–2779, 2021.

[CKLS18] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! a practical post-quantum public-key encryption from lwe and lwr. In *International Conference on Security and Cryptography for Networks*, pages 160–177. Springer, 2018.

[DKSRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In *Progress in Cryptology–AFRICACRYPT 2018: 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7–9, 2018, Proceedings 10*, pages 282–305. Springer, 2018.

[DT06] Ivan Damgård and Rune Thorbek. Linear integer secret sharing and distributed exponentiation. In *International Workshop on Public Key Cryptography*, pages 75–90. Springer, 2006.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of cryptology*, 9(4):199–216, 1996.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *FOCS*, volume 30, pages 248–253, 1989.

[LST21] Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed prfs from lwe. *Journal of Cryptology*, 34(3):1–49, 2021.

[Mon18] Hart Montgomery. A nonstandard variant of learning with rounding with polynomial modulus and unbounded samples. In *International Conference on Post-Quantum Cryptography*, pages 312–330. Springer, 2018.

[MS95] Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions, with applications to clipper-like key escrow systems. In *Annual International Cryptology Conference*, pages 185–196. Springer, 1995.

[Nie02] Jesper Buus Nielsen. A threshold pseudorandom function construction and its applications. In *Annual International Cryptology Conference*, pages 401–416. Springer, 2002.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 327–346. Springer, 1999.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

[NS78] Roger M Needham and Michael D Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.