OPRF from Input-Homomorphic Publicly Evaluable PRF

Yu Chen [*] Yamin Liu [†]

Abstract

Oblivious pseudorandom function (OPRF), a counterpart of PRF in distributed setting, has become a ubiquitous primitive used in secure multi-party computation. In this work, we provide a generic construction of OPRF from publicly evaluable PRF with input-homomorphic property. By plugging PEPRF from efficient group action (EGA) to the generic construction, we obtain a OPRF with plausible post-quantum security.

Keywords: OPRF, homomorphic, PEPRF

[*]Shandong University. Email: yuchen.prc@gmail.com
[†]Shield Lab, Huawei. Email: liuyamin3@huawei.com

Contents

1 Introduction

Pseudorandom functions (PRFs) is a core cryptographic primitive with numerous applications. Two decades later, Freedman et al. [FIPR05] introduce its counterpart in the distributed setting, namely oblivious pseudorandom functions (OPRF). In a nutshell, OPRF is an interactive 2-party protocol between a client with input $\{x_i\}_{i\in[n]}$, and a server with secret key k. It allows the client learn $\{F_k(x_i)\}_{i\in[n]}$ without anything else, and the server learns nothing. Formally, from the perspective of 2-party secure computation, an OPRF is a protocol securely fulfills the functionality $\mathcal{F}_{\text{oprf}} = (\perp, \{F_k(x_i)\}_{i\in[n]})$. OPRFs have shown to be a central primitive for building oblivious keyword search [FIPR05], private set intersection [KKRT16, HL10, CM20], secure data de-duplication [CCGS19], password-authenticated key exchange (PAKE) [JKK14, JKX18], private information retrieval [FIPR05], secure pattern matching [FHV13], cloud key management [JKR19].

So far, except the OPRFs based on symmetric primitives, all known efficient OPRF constructions rely on number-theoretical assumptions, which are not post-quantum secure. Recently, Boneh et al. [BKW20] propose an isogency-based OPRF. Albrecht et al. [ADDS21] present a lattice-based OPRF with *verifiable* property. These two constructions are post-quantum secure, but both them only serves as nice feasibility results, rather than practical results with good performance.

1.1 Motivation

Motivated by the state of affairs, we are intrigued to know:

Is there a generic construction of OPRF which admits for efficient instantiation?

1.2 Our Contribution

In this work, we make positive answers to the aforementioned questions. We present a generic construction of OPRF from PEPRF with input-homomorphic property, then give an efficient post-quantum secure instantiation from EGA.

2 Preliminaries

2.1 Notation

We use κ and λ to denote the computational and statistical parameter respectively. Let $\mathbb{Z}_n$ be the set $\{0, 1, \ldots, n-1\}$, $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \gcd(x, n) = 1\}$. We use $[n]$ to denote the set $\{1, \ldots, n\}$, and use $\mathsf{Perm}[n]$ to denote all the permutation over the set $\{1, \ldots, n\}$. We assume that every set X has a default order (e.g. lexicographical order), and write it as $X = \{x_1, \ldots, x_n\}$. For a set X, we use $|X|$ to denote its size and use $x \xleftarrow{\text{R}} X$ to denote sampling x uniformly at random from X. We use $(x_1, \ldots, x_n)$ to denote a vector, whose ith element is x_i. A function is negligible in κ, written $\mathsf{negl}(\kappa)$, if it vanishes faster than the inverse of any polynomial in κ. A probabilistic polynomial time (PPT) algorithm is a randomized algorithm that runs in polynomial time.

2.2 Weak Pseudorandom EGA

We begin by recalling the definition of a group action.

Definition 2.1 (Group Actions)**.** A group $\mathbb{G}$ is said to *act on* a set X if there is a map $\star : \mathbb{G} \times X \to X$ that satisfies the following two properties:

1. Identity: if e is the identity element of $\mathbb{G}$, then for any $x \in X$, we have $e \star x = x$.
2. Compatibility: for any $g, h \in \mathbb{G}$ and any $x \in X$, we have $(gh) \star x = g \star (h \star x)$.

From now on, we use the abbreviated notation $(\mathbb{G}, X, \star)$ to denote a group action. If $(\mathbb{G}, X, \star)$ is a group action, for any $g \in \mathbb{G}$ the map $\phi_g : x \mapsto g \star x$ defines a permutation of X.

We then define an effective group action (EGA) [AFMP20] as follows.

Definition 2.2 (Effective Group Actions)**.** A group action $(\mathbb{G}, X, \star)$ is *effective* if the following properties are satisfied:

1. The group $\mathbb{G}$ is finite and there exist PPT algorithms for:
   (a) Membership testing, i.e., to decide if a given bit string represents a valid group element in $\mathbb{G}$.
   (b) Equality testing, i.e., to decide if two bit strings represent the same group element in $\mathbb{G}$.
   (c) Sampling, i.e., to sample an element g from a uniform (or statistically close to) distribution on $\mathbb{G}$.
   (d) Operation, i.e., to compute gh for any $g, h \in \mathbb{G}$.
   (e) Inversion, i.e., to compute g^{-1} for any $g \in \mathbb{G}$.
2. The set X is finite and there exist PPT algorithms for:
   (a) Membership testing, i.e., to decide if a bit string represents a valid set element.
   (b) Unique representation, i.e., given any set element $x \in X$, compute a string $\hat{x}$ that canonically represents x.
3. There exists a distinguished element $x_0 \in X$, called the origin, such that its bit-string representation is known.
4. There exists an efficient algorithm that given (some bit-string representations of) any $g \in \mathbb{G}$ and any $x \in X$, outputs $g \star x$.

Definition 2.3 (Weak Pseudorandom EGA)**.** A group action $(G, X, \star)$ is weakly pseudorandom if the family of efficiently commutable permutation $\{\phi_g : X \to X\}_{g \in G}$ is weakly pseudorandom, i.e., there is no PPT adversary that can distinguish tuples of the form $(x_i, g \star x_i)$ from (x_i, u_i) where $g \xleftarrow{R} \mathbb{G}$ and each $x_i, u_i \xleftarrow{R} X$.

2.3 MPC in the Semi-honest Model

We use the standard notion of security in the presence of semi-honest adversaries. Let Π be a two-party protocol for computing the function $f(x_1, x_2)$, where party P_i has input x_i. We define security in the following way. For each party P, let $\text{View}_P(x_1, x_2)$ denote the view of party P during an honest execution of Π on inputs x_1 and x_2. The view consists of P's input, random tape, and all messages exchanged as part of the Π protocol.

Definition 2.4. 2-party protocol Π securely realizes f in the presence of semi-honest adversaries if there exists a simulator Sim such that for all inputs x_1, x_2 and all $i \in \{1, 2\}$:

$$\mathsf{Sim}(i, x_i, f(x_1, x_2)) \approx_c \text{View}_{P_i}(x_1, x_2)$$

Roughly speaking, a protocol is secure if the party with x_i learns no more information other than $f(x_1, x_2)$ and x_i.

3 Review of Pseudorandom Function

In this section, we first recall the notion of standard pseudorandom functions (PRFs) [GGM86], then recap its counterpart in the public-key setting, namely publicly evaluable pseudorandom functions [CZ14].

3.1 Standard PRF

Definition 3.1 (PRF)**.** A family of PRFs consists of three polynomial-time algorithms as follows:

- $\mathsf{Setup}(1^\kappa)$: on input a security parameter κ, outputs public parameter pp specifying (F, K, X, Y), where $F : K \times X \to Y$ is a family of keyed functions, K is the key space, X is domain, and Y is range.

- $\mathsf{KeyGen}(pp)$: on input pp, outputs a secret key $k \xleftarrow{R} K$.
- $\mathsf{Eval}(k, x)$: on input $k \in K$ and $x \in X$, outputs $y \leftarrow F(k, x)$. For notation convenience, we will write $F(k, x)$ as $F_k(x)$ interchangeably.

Correctness: For any $\kappa \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\kappa)$ and any $k \leftarrow \mathsf{KeyGen}(pp)$, we have $F_{sk}(x) = \mathsf{Eval}(k, x)$.

Pseudorandomness. Let $\mathcal{A}$ be an adversary against PRFs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\kappa) = \Pr\left[\beta' = \beta : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\kappa); \\ k \leftarrow \mathsf{KeyGen}(pp); \\ \beta \leftarrow \{0,1\}; \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ror}}(\beta,\cdot)}(\kappa); \end{array}\right] - \frac{1}{2},$$

where $\mathcal{O}_{\mathsf{ror}}(\beta, \cdot)$ denotes the real-or-random oracle controlled by β, i.e., $\mathcal{O}_{\mathsf{ror}}(0, x) = F_k(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$ (here H is chosen uniformly at random from all the functions from D to R[1]). $\mathcal{A}$ can adaptively access the oracle $\mathcal{O}_{\mathsf{ror}}(\beta, \cdot)$ polynomial many times. We say that F is pseudorandom if for any PPT adversary $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ. We refer to such security as full PRF security.

Sometimes the full PRF security is not needed and it is sufficient if the function cannot be distinguished from a uniform random one when challenged on random inputs. The formalization of such relaxed requirement is *weak pseudorandomness*, which is defined the same way as pseudorandomness except that the inputs of oracle $\mathcal{O}_{\mathsf{ror}}(\beta, \cdot)$ are uniformly chosen from X by the challenger instead of adversarially chosen by $\mathcal{A}$. PRFs that satisfy weak pseudorandomness are referred to as *weak PRFs*.

Remark 3.1. In a standard PRF, it is always harmless to explicitly introduce a public key, which includes the information related to secret key that can be made public. For example, in the Naor-Reingold PRF [NR04] based on the DDH assumption: $\mathsf{F}_{\vec{a}}(x) = (g^{a_0})^{\prod_{x_i=1} a_i}$, where $\vec{a} = (a_0, a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ is the secret key, g^{a_i} for $1 \le i \le n$ can be safely published as the public key. If no information can be made public, one can simply set $pk = \perp$. Looking ahead, such treatment allows us get a unified syntax of PRF in both minicrypt and cryptmania.

3.2 Publicly Evaluable PRF

Definition 3.2 (Publicly Evaluable PRFs)**.** A family of PEPRFs consists of four polynomial-time algorithms as follows:

- $\mathsf{Setup}(1^\kappa)$: on input a security parameter κ, output public parameter pp specifying (F, PK, SK, X, Y, L, W), where $F : SK \times X \to Y \cup \perp$ is family of keyed functions, SK is the secret key space, PK is the corresponding public key space, X is domain, Y is range, $L \subseteq X$ is a NP language defined by some hard relation R_L, and W is the space of associated witnesses. Here, we assume that R_L is efficiently sampleable, that is, there exists a PPT algorithm $\mathsf{SampRel}$ which on input random coins r, output a random tuple $(x, w) \in \mathsf{R}_L$.
- $\mathsf{KeyGen}(pp)$: on input pp, output a secret key sk and an associated public key pk.[2]
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \in Y \cup \perp$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$, output $y \in Y$.

Correctness: For any $\lambda \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$, we have:

$$\begin{aligned} \forall x \in X : &\quad F_{sk}(x) = \mathsf{PrivEval}(sk, x) \\ \forall x \in L \text{ with a witness } w : &\quad F_{sk}(x) = \mathsf{PubEval}(pk, x, w) \end{aligned}$$

[1] To efficiently simulate access to a uniformly random function H from D to R, one may think of a process in which the adversary's queries to $\mathcal{O}_{\mathsf{ror}}(1, \cdot)$ are "lazily" answered with independently and randomly chosen elements in R, while keeping track of the answers so that queries made repeatedly are answered consistently.

[2] In a standard PRF, it is also harmless to explicitly introduce a public key, which includes the information related to secret key that can be made public. For example, in the Naor-Reingold PRF [NR04] based on the DDH assumption: $\mathsf{F}_{\vec{a}}(x) = (g^{a_0})^{\prod_{x_i=1} a_i}$, where $\vec{a} = (a_0, a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ is the secret key, g^{a_i} for $1 \le i \le n$ can be safely published as the public key. If no information can be made public, one can always assume $pk = \{\perp\}$.

Input homomorphic: Suppose $L \subseteq X$ and Y are finite abelian groups. F is input homomorphic if for any $x_1, x_2 \in L$ we have $F_{sk}(x_1 + x_2) = F_{sk}(x_1) + F_{sk}(x_2)$.

Weak pseudorandomness: Let $\mathcal{A}$ be an adversary against PEPRFs and define its advantage as:

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \mathsf{KeyGen}(pp); \\ \{r_i^* \xleftarrow{\mathrm{R}} R, (x_i^*, w_i^*) \leftarrow \mathsf{SampRel}(r_i^*)\}_{i=1}^{p(\lambda)}; \\ \beta \leftarrow \{0,1\}; \\ \beta' \leftarrow \mathcal{A}(pp, pk, \{x_i^*, \mathcal{O}_{\mathsf{ror}}(\beta, x_i^*)\}_{i=1}^{p(\lambda)}); \end{array}\right] - \frac{1}{2},$$

where $p(\kappa)$ is any polynomial, $\mathcal{O}_{\mathsf{ror}}(\beta, \cdot)$ is the real-or-random oracle. We say that F is weak pseudorandom if for any PPT adversary $\mathcal{A}$ its advantage function $\mathsf{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ.

Looking ahead, to simplify the security proof we will use a one-time flavor definition that $\mathcal{A}$ only queries $\mathcal{O}_{\mathsf{ror}}$ once when arguing the security of PEPRF constructions. A standard hybrid argument shows that such seemingly restricted definition is actually equivalent to the standard multi-query definition.

Remark 3.2. Different from standard PRFs, PEPRFs require the existence of a language $L \subseteq X$ as well as a public evaluation algorithm. Due to this strengthening on functionality, we cannot hope to achieve standard pseudorandomness, and hence weak pseudorandomness is the best we can achieve.

4 OPRF from Input-Homomorphic PEPRF

4.1 Definition of OPRF

In Figure 1, we adapt the definion of multi-point OPRF [CM20] under unified PRF syntax.

Parameters: A PRF $F : SK \times X \to Y$, and two parties: server P_1 and client P_2

Functionality:

1. Generate a key pair (pk, sk) for PRF. Give (pk, sk) to P_1 and give pk to P_2.
2. Wait for input $(x_1, \ldots, x_n)$ from P_2. Give $(F_{sk}(x_1), \ldots, F_{sk}(x_n))$ to P_2

Figure 1: Ideal functionality $\mathcal{F}_{\mathsf{mqRPMT}}$ for multi-point OPRF

4.2 Construction from Input-Homomorphic PEPRF

In Figure 2, we show how to build OPRF $F : SK \times \{0,1\}^\ell \to Y$ from PEPRF $G : SK \times X \to Y$ and cryptographic hash function $\mathsf{H} : \{0,1\}^\ell \to L$.

Remark 4.1. Looking ahead, $\mathsf{H} : \{0,1\}^\ell \to L$ will be modeled as a random oracle. This implicitly requires L is dense.

Correctness. The correctness of the above OPRF construction follows from that of input-homomorphic PEPRF.

Theorem 4.1. *The OPRF protocol described in Figure 2 is secure in the semi-honest model assuming* H *is a random oracle and* F *is a family of PEPRFs.*

Proof. In the above construction $F_{sk}(x) : SK \times \{0,1\}^\ell \to Y$ is defined as $G_{sk}(\mathsf{H}(x))$. By assuming H is a random oracle, we can reduce the pseudorandomness of F to the weak pseudorandomness of G. In other words, H amplifies weak pseudorandomness to standard pseudorandomness by leveraging *programmability*.

We then prove the above construction is indeed a secure OPRF protocol. We exhibit simulators Sim_{P_1} and Sim_{P_2} for simulating corrupt P_1 and P_2 respectively, and argue the indistinguishability of the simulated transcript from the real execution.

Parameters:

- Common input: PEPRF $G : SK \times X \to Y$, hash function $\mathsf{H} : \{0,1\}^\ell \to L$.
- Input of server P_1: security parameter κ.
- Input of client P_2: $X = \{x_1, \ldots, x_n\} \subseteq \{0,1\}^\ell$

Protocol:

1. P_1 runs $pp \leftarrow \text{PEPRF.}\mathsf{Setup}(1^\kappa)$ and $(pk, sk) \leftarrow \text{PEPRF.}\mathsf{KeyGen}(pp)$, then sends pp and pk to P_2.
2. P_2 runs $\mathsf{SampRel}(pp)$ independently n times and obtains n tuples (z_i, w_i), then sends $\{\mathsf{H}(x_i) + z_i\}_{i\in[n]}$ to P_1.
3. P_1 computes $v_i \leftarrow \{F_{sk}(\mathsf{H}(x_i) + z_i)\}_{i\in[n]}$ and sends them to P_2. P_2 computes $y_i \leftarrow v_i - \text{PEPRF.}\mathsf{PubEval}(pk, z_i, w_i)$.

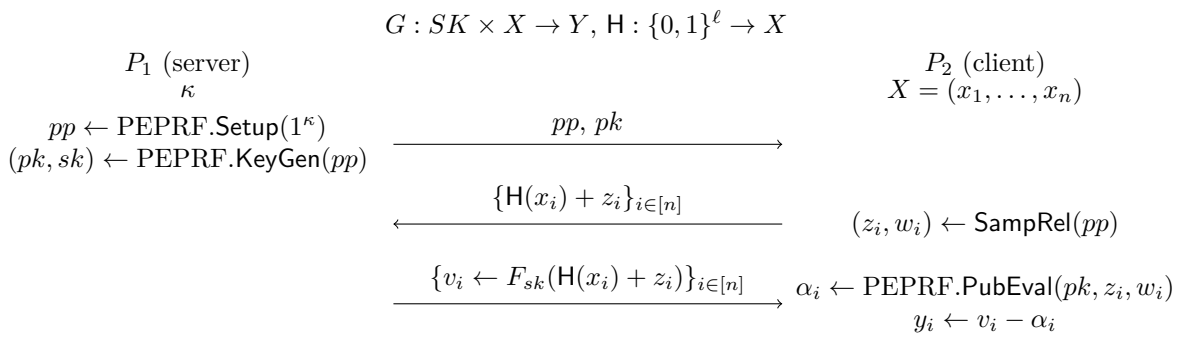

Figure 2: OPRF from input-homomorphic PEPRF

Security against corrupt server. Sim_{P_1} simulates the view of corrupt server P_1, which consists of P_1's randomness, input, output and received messages. We formally show Sim_{P_1}'s simulation is indistinguishable from the real execution via a sequence of hybrid transcripts.

Hybrid_0: P_1's view in the real protocol.

Hybrid_1: Given P_1's input X, Sim_{P_1} chooses the randomness for P_1 (i.e., picks $r_i \xleftarrow{\text{R}} R$ to sample $(z_i, w_i) \in \mathsf{R}_L$), and simulates with the knowledge of X. Sim_{P_1} outputs $(\mathsf{H}(x_i) + z_i, \ldots, \mathsf{H}(x_n) + z_n)$. Clearly, Sim_{P_1}'s simulated view in Hybrid_1 is identical to P_1's real view.

Hybrid_2: Sim_{P_1} simulates without the knowledge of X. Sim_{P_1} picks $r_i \xleftarrow{\text{R}} R$ to sample $(z_i, w_i) \in \mathsf{R}_L$, then outputs $(z_1, \ldots, z_n)$.

Clearly, the simulated view in Hybrid_1 and Hybrid_2 are identical due to z_i distributes uniformly at random over L.

Security against corrupt client. Sim_{P_2} simulates the view of corrupt client P_2, which consists of P_2's randomness, input, output and received messages. We formally show Sim_{P_2}'s simulation is indistinguishable from the real execution via a sequence of hybrid transcripts.

Hybrid_0: P_2's view in the real protocol.

Hybrid_1: Given P_2's input x_i, randomness r_i, and output pk and y_i, Sim_{P_2} samples $(z_i, w_i) \in \mathsf{R}_L$, then simulates the received messages $v_i := y_i + \text{PEPRF}.\mathsf{PubEval}(pk, z_i, w_i)$.

Clearly, the simulated view in Hybrid_1 and Hybrid_2 are identical due to the publicly evaluable property of the underlying PEPRF.

This proves the theorem. □

5 Instantiations of Input-Homomorphic PEPRF

It remains to give efficient instantiations of input-homomorphic PEPRF.

5.1 Input-Homomorphic PEPRF from the DDH Assumption

- $\mathsf{Setup}(1^\kappa)$: on input a security parameter κ, runs $(\mathbb{G}, g, p) \leftarrow \mathsf{GroupGen}(1^\kappa)$, output public parameter pp specifying (F, PK, SK, X, Y, L, W), where $PK = \mathbb{G}$, $SK = \mathbb{Z}_p$, $X = L = Y = \mathbb{G}$, $W = \mathbb{Z}_p$, $F : SK \times X \to Y$ is defined as $F_{sk}(x) := x^{sk}$, $(x, w) \in \mathsf{R}_L$ iff $g^w = x$.
- $\mathsf{KeyGen}(pp)$: on input pp, picks secret key $sk \xleftarrow{\text{R}} \mathbb{Z}_p$ and computes public key $pk \leftarrow g^{sk}$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \leftarrow x^{sk}$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ together with a witness $w \in W$, output $y \leftarrow pk^w$.

Input homomorphic. $\forall x_1, x_2 \in X$, we have $(x_1 \cdot x_2)^{sk} = x_1^{sk} \cdot x_2^{sk}$.

Theorem 5.1. *The above construction is weakly pseudorandom if the DDH assumption holds.*

Proof. The weak pseudorandomness immediately follows from the DDH assumption. We omit the security proof here due to its straightforwardness. □

The above construction and security proof directly carry to the weak-pusudorandom EGA setting.

5.2 Input-Homomorphic PEPRF from the LWE Assumption

LWE is parameterized by positive integers n and q, and an error distribution χ over $\mathbb{Z}$. For concreteness, n and q can be thought of as roughly the same as in SIS, and χ is usually taken to be a discrete Gaussian of width αq for some $\alpha < 1$, which is often called the relative "error rate".

Definition 5.1 (LWE distribution)**.** For a vector $s \in \mathbb{Z}_q^n$ called the secret, the LWE distribution $A_{s,\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $a \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \xleftarrow{\text{R}} \chi$, and outputting $(a, b = \langle s, a \rangle + e \bmod q)$.

Definition 5.2 (Decision-LWE$_{n,q,\chi,m}$)**.** Given m independent samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where every sample is distributed according to either: (1) $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q$ (fixed for all samples), or (2) the uniform distribution, distinguish which is the case (with non-negligible advantage).

We give a LWE-based PEPRF as below.

- $\mathsf{Setup}(1^\kappa)$: on input a security parameter κ, first chooses a uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, output public parameter pp specifying (F, PK, SK, X, Y, L, W), where $PK = \mathbb{Z}_q^m$, $SK = \mathbb{Z}_q^n$, $X = L = \mathbb{Z}_q^{n+1}$, $W = \{0,1\}^{m+1}$. Here the collection of $\mathcal{NP}$ languages indexed by $pk = \mathbf{u} \in \mathbb{Z}_q^m$ is defined as:

$$L_{\mathbf{u}} = \{\mathbf{x} = (\mathbf{c}, v) \in \mathbb{Z}_q^{n+1} | \exists \mathbf{w} = (\mathbf{a} \in \{0,1\}^m, b \in \{0,1\}) \text{ s.t. } \mathbf{c} = \mathbf{A}\cdot\mathbf{a}, v = \mathbf{u}^t\cdot\mathbf{a} + b\cdot\lfloor q/2\rceil\}$$

  $F : SK \times X \to Y$ is defined as $F_{\mathbf{s}}(\mathbf{x} = (\mathbf{c}, v)) := \mathsf{Decode}(v - \mathbf{s}^t\cdot\mathbf{c})$, where $\mathsf{Decode}(z)$ outputs "0" if z is closer to 0 or "1" if z is closer to $\lfloor q/2\rceil$ modulo q.

- $\mathsf{KeyGen}(pp)$: on input pp, first picks a secret key $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, then picks $\mathbf{e} \xleftarrow{\text{R}} \chi^m$ and computes public key $\mathbf{u} \leftarrow \mathbf{A}^t\cdot\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$.

- $\mathsf{PrivEval}(\mathbf{s}, \mathbf{x})$: on input $\mathbf{s}$ and $\mathbf{x} = (\mathbf{c}, v)$, output $y \leftarrow \mathsf{Decode}(v - \mathbf{s}^t\cdot\mathbf{c}) \in \{0,1\}$.

- $\mathsf{PubEval}(\mathbf{u}, \mathbf{x}, \mathbf{w})$: on input public key $\mathbf{u}$ and $\mathbf{x} = (\mathbf{u}, v) \in L$ together with a witness $\mathbf{w} = (\mathbf{a}, b) \in W$, output $y \leftarrow b$.

Correctness. Note that $\mathbf{s}^t\cdot\mathbf{c} - \mathbf{u}^t\cdot\mathbf{a} = \mathbf{s}^t\cdot\mathbf{A}\cdot\mathbf{a} - (\mathbf{A}^t\cdot\mathbf{s} + \mathbf{e})^t\cdot\mathbf{a} = \mathbf{e}^t\cdot\mathbf{a} \in \mathbb{Z}_q$. The correctness holds as long as $(\mathbf{e}^t\cdot\mathbf{a})/q$ is negligible in κ.

Input homomorphic. For any $\mathbf{x}_1 = (\mathbf{A}\cdot\mathbf{a}_1, \mathbf{u}^t\cdot\mathbf{a}_1 + b_1\cdot\lfloor q/2\rceil)$ and $\mathbf{x}_2 = (\mathbf{A}\cdot\mathbf{a}_2, \mathbf{u}^t\cdot\mathbf{a}_2 + b_2\cdot\lfloor q/2\rceil)$, we have:

$$\begin{aligned} F_{\mathbf{s}}(\mathbf{x}_1 + \mathbf{x}_2) &= \mathsf{Decode}(v_1 - \mathbf{s}^t\cdot\mathbf{c}_1 + v_2 - \mathbf{s}^t\cdot\mathbf{c}_2) \\ &= \mathsf{Decode}(\mathbf{e}^t\cdot\mathbf{a}_1 + b_1\lfloor q/2\rceil + \mathbf{e}^t\cdot\mathbf{a}_2 + b_2\lfloor q/2\rceil) \\ &\approx \mathsf{Decode}((b_1 + b_2)\lfloor q/2\rceil) \\ &= b_1 + b_2 \approx F_{\mathbf{s}}(\mathbf{x}_1) + F_{\mathbf{s}}(\mathbf{x}_2) \end{aligned}$$

Security. The weak pseudorandomness follows from the following hybrid arguments.

- Game$_0$: the original security game. $\mathcal{A}$ gets to see $(pk = \mathbf{u}, \mathbf{x} \xleftarrow{\text{R}} L_{\mathbf{u}}, k^*)$, where $k^* = b$ if $\beta = 0$ and k^* is a random bit if $\beta = 1$.

- Game$_1$: same as Game$_0$ except that $\mathcal{CH}$ changes $\mathbf{u}$ from the LWE distribution to a uniform distribution over $\mathbb{Z}_q^m$. By the LWE assumption, we have Game$_0 \approx_c$ Game$_1$.

- Game$_2$: same as Game$_1$ except that $\mathcal{CH}$ changes $\mathbf{x}$ to a uniform distribution over $\mathbb{Z}_q^{n+1}$. By the leftover hash lemma, we have Game$_1 \approx_s$ Game$_2$. Clearly, (even unbounded) $\mathcal{A}$'s advantage in Game$_2$ is zero.

5.3 Input-Homomorphic PEPRF from the Ring-LWR Assumption

Define ring $R_q = \mathbb{Z}_q[X]/(X^n+1), R_p = \mathbb{Z}_p[X]/(X^n+1)$, where $q > p \geq 2$ are integers, $n = 2^k$ for integer $k > 0$, and $X^n + 1$ is the 2^{k+1}-th cyclotomic polynomial.

Define the modular function $[\cdot]_p : \mathbb{Z} \to \mathbb{Z}_p$ as $[x]_p = x \mod p$, and define the rounding function $\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor x\rceil_p = \lfloor \frac{p}{q}\cdot x\rceil \mod p$. The definition can be extended naturally in a coefficient-wise manner on R_q and R_p.

- $\mathsf{Setup}(1^\kappa)$: define $pp = (F, PK, SK, X, Y, L, W)$, where $PK = R_p$, $SK = R_q$, $X = Y = L = R_p$, $W = R_q$.
  $F : SK \times X \to Y$ is defined as $F_{sk}(x) = \lfloor x\cdot sk\rceil_p$.
  For public parameter $a \in R_q$, $(x, w) \in R_L$ iff $x = \lfloor aw\rceil_p$.

- $\mathsf{KeyGen}(pp)$: on input pp, select the secret key $sk \leftarrow \chi(R_q)$, where χ is a sparse distribution over R_q, and set the public key as $pk \leftarrow \lfloor a \cdot sk \rceil_p$.
- $\mathsf{PrivEval}(sk, x)$: on input sk and $x \in X$, output $y \leftarrow \lfloor x \cdot sk \rceil_p$.
- $\mathsf{PubEval}(pk, x, w)$: on input pk and $x \in L$ with the witness $w \in W$, output $y \leftarrow \lfloor pk \cdot w \rceil_p$.

Properties of the above ring-LWR based IHPEPRF construction:

Correctness. For any $\kappa \in \mathbb{N}$, any $pp \leftarrow \mathsf{Setup}(1^\kappa)$ and any $(pk, sk) \leftarrow \mathsf{KeyGen}(pp)$:

- $\forall x \in X$: $F_{sk}(x) = \lfloor x \cdot sk \rceil_p = \mathsf{PrivEval}(sk, x)$.
- $\forall x \in L$ with witness w:

$$\mathsf{PubEval}(pk, x, w) = \lfloor pk \cdot w \rceil_p = \lfloor \lfloor \frac{p}{q} \cdot a \cdot sk \rceil \rceil_p \cdot w \rceil_p,$$

$$F_{sk}(x) = \lfloor x \cdot sk \rceil_p = \lfloor \lfloor \frac{p}{q} \cdot a \cdot w \rceil \rceil_p \cdot sk \rceil_p$$

Input homomorphic. A bit tricky while dealing with the rounding function: it is not linear.

$$F_{sk}(x_1 + x_2) = [\lfloor \frac{p}{q}(x_1 + x_2) \cdot sk \rceil]_p,$$

$$F_{sk}(x_1) + F_{sk}(x_2) = [\lfloor \frac{p}{q} \cdot x_1 \cdot sk \rceil]_p + [\lfloor \frac{p}{q} \cdot x_2 \cdot sk \rceil]_p.$$

If $\lfloor x_1 \rceil = \lfloor x_2 \rceil$, then $F_{sk}(x_1 + x_2) = F_{sk}(x_1) + F_{sk}(x_2)$.

Security. Guaranteed by the ring LWR assumption: $(a, \lfloor a \cdot s \rceil_p) \approx (a, u)$.

The above construction is floppy and could be problematic.

6 Concrete Post-quantum Secure OPRF

By plugging PEPRF from weak pseudorandom EGA in Section 5.1 to our generic construction, we obtain a OPRF with plausible post-quantum security.

We encounter a technical hurdle when plugging the instantiation in Section 5.2 to our generic construction. This is because the weak pseudorandomness of F is tied to special distribution over $L_{\mathbf{u}}$, not a uniform distribution. The consequence is that we can not assume there exists a random oracle from $\{0,1\}^*$ to $L_{\mathbf{u}}$ anymore. One attempt is to remove v from $\mathbf{x}$ and only keep $\mathbf{c}$. However, this somehow requires a fuzzy and homomorphic randomness extractor, which again seems quite tough.

References

[ADDS21] Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In *Public-Key Cryptography - PKC 2021*, volume 12711 of *Lecture Notes in Computer Science*, pages 261–289. Springer, 2021.

[AFMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *Advances in Cryptology - ASIACRYPT 2020*, volume 12492 of *LNCS*, pages 411–439. Springer, 2020.

[BKW20] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020*, volume 12492 of *Lecture Notes in Computer Science*, pages 520–550. Springer, 2020.

[CCGS19] Jan Camenisch, Angelo De Caro, Esha Ghosh, and Alessandro Sorniotti. Oblivious PRF on committed vector inputs and application to deduplication of encrypted data. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019*, volume 11598 of *Lecture Notes in Computer Science*, pages 337–356. Springer, 2019.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 34–63. Springer, 2020.

[CZ14] Yu Chen and Zongyang Zhang. Publicly evaluable pseudorandom functions and their applications. In *9th International Conference on Security and Cryptography for Networks, SCN 2014*, pages 115–134, 2014.

[FHV13] Sebastian Faust, Carmit Hazay, and Daniele Venturi. Outsourced pattern matching. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013*, volume 7966 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2013.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324. Springer, 2005.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[HL10] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. *J. Cryptol.*, 23(3):422–456, 2010.

[JKK14] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *Advances in Cryptology - ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2014.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*, pages 379–393. ACM, 2019.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In *Advances in Cryptology - EUROCRYPT 2018*, volume 10822 of *Lecture Notes in Computer Science*, pages 456–486. Springer, 2018.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *CCS 2016*, pages 818–829. ACM, 2016.

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

A Missing Definitions