Multi-Point HashDH OPRF using Multiplicative Blinding with Application to Private Set Intersection

Minglang Dong**

Abstract

The privacy set intersection (PSI) protocol with the oblivious pseudorandom function (OPRF) as the core component is a crucial member of PSI family, and the most efficient PSI protocol at present also belongs to this category. Based on DDH assumption, Hash Diffie-Hellman (HashDH) PSI is one of the most classical PSI protocols. Benefiting by its low communication overhead, it still has tremendous research value today. The OPRF subprotocol beneath the classical DH-PSI protocol falls into our abstract blind-query-de-blinding OPRF paradigm, while employing the exponential blinding (Exp-HashDH) method. An alternative method called multiplication blinding (Mult-HashDH) offers the improvement which the exponential blinding can't give in performance. This technique substitutes multiple variable-base exponentiations with fixed-base exponentiations, and by taking full advantage of this outstanding feature and pre-computation, the computational efficiency of the client can be at least doubled. However, neither Mult-HashDH OPRF nor Mult-HashDH PSI can provide a rigorous security proof under the semi-honest model, which makes the security of this scheme reel from a crisis of confidence. In this paper, the security proof of a modified Mult-HashDH OPRF protocol is formally given under the semi-honest model, which not only ensures the security of the scheme but also have no influence on damaging the efficiency of the protocol. Then along comes a secure HashDH PSI protocol constructed based on it. The experimental comparison shows that our protocol achieves 2.65—13.20× speedup in running time.

Keywords: OPRF, HashDH OPRF, PSI, DH-PSI, Multiplicative Blinding

1 Introduction

As one of the most fundamental and intriguing primitive employed in cryptographic protocol, Oblivious Pseudorandom Function (OPRF) is a secure two-party protocol with a formal definition computing a deterministic functionality $f(k, x) = (\Lambda, F_k(x))$, where F_k is a Pseudorandom Function (PRF).

This default definition can be extended into multiple queries variation, and what we focus on in this paper is non-adaptive seting, which means client must determine all points $x_1, x_2, \cdots, x_n$ for multiple queries as inputs before protocol execution. We call OPRF meeting this seting as multi-point OPRF.

A notable paradigm but easily come to mind is Blind-Query-Unblind paradigm, where the first-round messages are generated by client blinding its inputs and sending to server, then server performs as an oracle in PRF definition, i.e. computes F_k on every blinded inputs and returns to client, ultimately client unblinds received pseudorandom function values to recover F_k on real inputs. We call the OPRF protocol that falls into the Blind-Query-Unblind paradigm as Blind-Query-Unblin OPRF.

Based on the Decisional Diffie-Hellman (DDH) Assumption, HashDH OPRF is the most canonical and significant member in this paradigm, which can be classified into two categories, namely HashDH OPRF using Exponential Blinding(Exp) and HashDH OPRF using Multiplicative Blinding(Mult), occupying the leading positions in both theory and practice.

The most customary blinding way in HashDH OPRF is exponential blinding(Exp-HashDH OPRF), which was put forward together with OPRF. Another alternative blinding technique, multiplicative blinding, is a slight variation of Blind-Query-Unblind OPRF(Mult-HashDH OPRF), providing an excellent performance improvement of client attributing to substituting variable-base exponentiations with fixed-base exponentiations. For example, in HashDH-PSI protocol on elliptic curve groups, it can replace the client's variable-point multiplication with the fixed-point multiplication and at least doubles the computation efficiency of the client through precomputation.

*School of Cyber Science and Technology, Shandong University. Email: minglangdong@icloud.com

Although the multiplicative blinding possessing such an thrilling performance, it also has an potential vulnerability, the lack of provable security in standard OPRF definition under rigid security paradigm for MPC, which leads to its long being ignored and neglected. The essential reason why this happens is that the standard OPRF defines the key k as the input of server, so as to achieve stronger privacy protection of the key k, while in the preceding Mult-HashDH OPRF, the server's sending g^k is equivalent to the active disclosure of part of the key information. In consideration of distinguisher is dictated to obtain two parties's inputs including server's key k of pseudorandom function, it can trivially distinguish g^k and a uniformly random message, so this message cannot be simulated by the simulator of the corrupted client.

One exisiting solution [JKX21] is to regard g^k as the public key of PRF and append it to two parties's inputs separately, which means to amend the OPRF protocol into public-key setting where two parties jointly compute a functionality $f((sk, pk), (x, pk)) = (\Lambda, F_{sk}(x))$. The biggest problem with this approach is deviating the standard OPRF definition, resulting in most application scenarios, e.g. Private Set Intersection (PSI), it is not applicable. The most significant contribution of our paper is to propose a new Mult-HashDH OPRF protocol, which can be formally proves in standard OPRF definition, so that can be applied in most applications scenarios such as PSI. On the basis of the original protocol, we first revises g as a uniformly sampled element by client in the protocol rather than the generator of the group $\mathbb{G}$, and then employs a relaxed version for server's security while sufficient for most application scenarios to give the security proof of Mult-HashDH OPRF in the MPC security definition under the semi-honest model and the standard OPRF definition, which not only ensures the security of the scheme but also does not affect the efficiency of the protocol.

There is a plain observation that multi-point rOPRF naturally implies PSI protocol, which means Mult-HashDH OPRF can be automatically applied to classical HashDH-PSI. Therefore, another contributions of our work is that we construct a Mult-HashDH PSI, which is not only as secure as classical HashDH-PSI and reserves the advantage of low communication, but also possesses a higher computational efficiency than it. We implemented the protocol in c++ and compared it with the classical DH-PSI protocol using Bloom filter optimization techniques. The experiment result indicates that our protocol achieves 2.65−13.20× speedup in running time.

Although the classical DH-PSI is much slower than current state of the art PSI (because the number of power operations increases linearly with the number of elements), it is still of great value to study and improve it. Mike Rosulek et al. [RT21] proposed the following reasons: First, communication cost is more important than computation cost in some cases. For example, Ion et al. [IKN+19] reported their actual deployment of PSI-like functionality in Google. They chose to deploy Diffie-Hellman PSI, because *"Somewhat surprisingly, for the offline 'batch computing' scenarios we consider, communication costs are far more important than computation. This is especially the case for a secure protocol involving multiple businesses, where servers cannot be co-located (Wide area network solutions). Networks are inherently shared, and it is much less expensive to add CPUs to a shared network than to expand network capacity."* Second, DH-PSI is the cheapest protocol in the case of small set inputs. For example, DH-PSI is the most efficient choice in balanced scenarios where both sets consist of several hundred elements or in extremely unbalanced scenarios where the input of one of the largest sets is several thousand elements.

2 Preliminaries

2.1 Pseudorandom Functions

Pseudorandom functions (PRFs) is a family of functions which is indistinguishable from truly random functions. Formally, it's defined as follows.

Definition 2.1 (PRFs)**.** Let $F : \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^t$ be an efficient, keyed function, where the first argument is called the key and denoted k, and the second argument is called the input to the function. Consider the set $Func_n$ of all functions mapping n-bit strings to n-bit strings. The adversary D is given oracle access to some function F. Namely, D has access to outputs of the function F on polynomially many inputs $x_1, x_2, \cdots$ of its adaptive choices, which we denote as D^F . F is a pseudorandom function if for all probabilistic polynomial-time distinguishers D, there is a negligible function $negl$ such that:

$$\left| Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq negl(n),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in Func_n$ and the randomness of D.

2.2 Oblivious Pseudorandom Function

Freedman et al. [FIPR05] first formalized the conception of oblivious pseudorandom function (OPRF) with two versions: strong (standard) OPRF, as opposed to weak (also called relaxed) OPRF, which does not prevent the client from learning partial information about the key. The strongest definition of OPRF, which is also the standard definition of OPRF, is a secure two-party protocol with a formal definition computing a deterministic functionality $f(k, x) = (\Lambda, F_k(x))$, where F_k is some pseudorandom function family.

Definition 2.2 (Strong OPRF)**.** A two-party protocol π is said to be a strong OPRF if there exists some PRF family F_k, such that π privately realizes the following functionality.

- Inputs: Client holds an evaluation point x; Server holds a key k.
- Outputs: Client outputs $F_k(x)$; Server outputs nothing.

The strong OPRF provides such a strong security level that some natural and efficient OPRF protocols do not satisfy this strong definition, yet are sufficient for the most applications. So Freedman et al. proposed a alternative OPRF definition providing a relaxed server's security at the same time as they put forward the formalization of strong OPRF.

Definition 2.3 (Relaxed OPRF)**.** A two-party protocol π is said to be a (non-adaptive, 1-time) relaxed OPRF if there exists some PRF family F_k, such that the following hold.

(Correctness and client's privacy) These properties remain the same as in Definition 2.2, i.e., using the functionality $f(k, x) = (\Lambda, F_k(x))$.

(Server's privacy) Consider an augmented protocol π' in which the input of S consists of m evaluation points $y_1, \cdots, y_m$ (instead of a key k) and the input of C is an evaluation point x (as in π). Protocol π' proceeds as follows: (1) S picks a key k at random; (2) S, C invoke π on inputs (k, x); (3) S outputs $(F_k(y_1), \cdots, F_k(y_m))$ and C outputs its output in π. We require that the augmented protocol π' provides server security with respect to the following randomized functionality f':

- Inputs: Client holds an evaluation point x; Server holds an arbitrary set of evaluation points $(y_1, y_2, \cdots, y_m)$.
- Outputs: Client outputs $F_k(x)$ and server outputs $(F_k(y_1), \cdots, F_k(y_m))$, where the key k is uniformly chosen by the functionality.

Specifically, for any (efficient, malicious) client C' attacking π', there is a simulator C'' playing the client's role in the ideal implementation of f', such that on all inputs $(y_1, y_2, \cdots, y_m), x)$, the view of C' concatenated with the output of S in π' is computationally indistinguishable from the output of C'' concatenated with that of S in the ideal implementation of f'.

2.3 Security in MPC for Semi-honest parties

In Secure Multi-Party Computation (MPC), proving security of a protocol always be achieved via the use of simulation. Generally, we aim to construct an ideal world including a simulator who konws nothing about the honest party's input while can successfully simulate the corrupted party's view concatenated with two parties' outputs, so that for any probabilistic polynomial-time distinguisher, the ideal world is indistinguishable from the real world where the protocol is executed.

Formally, if $f(x, y) = (f_1(x, y), f_2(x, y))$ is a probabilistic polynomial-time functionality and π is a two-party protocol for computing f, where x, y are two parties's inputs and $f_1(x, y), f_2(x, y)$ are are two parties's outputs respectively. We use $\mathbf{view}_i^\pi$ to denote the corrupted party i's view to be compared, which contains its input, internal random tape and all messages it received, while use $\mathbf{output}^\pi = (\mathbf{output}_1^\pi, \mathbf{output}_2^\pi)$ to denote the joint outputs in the real protocol's execution.

Definition 2.4. Let $f = (f_1, f_2)$ be a functionality. We say that π securely computes f in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms S_1 and S_2 such that

$$\{(S_1(1^n, x, f_1(x, y)), f(x, y)\}_{x,y,n} \stackrel{c}{\equiv} \{(view_1^\pi(x, y, n), output^\pi(x, y, n))\}_{x,y,n},$$
$$\{(S_2(1^n, y, f_2(x, y)), f(x, y)\}_{x,y,n} \stackrel{c}{\equiv} \{(view_2^\pi(x, y, n), output^\pi(x, y, n))\}_{x,y,n},$$

where $x, y \in \{0, 1\}^*$ such that $|x| = |y|$, and $n \in \mathbb{N}$.

Especially, if f is a deterministic polynomial-time functionality, the aforementioned definition can be compacted into a simpler formulation which contains two requirements: (a) **correctness**, meaning that the output of the parties is correct, and (b) **privacy**, meaning that the view of each party can be (separately) simulated.

Definition 2.5. (Correctness) There exists a negligible function ϵ such that for every $x, y \in \{0, 1\}^*$ and every n,

$$Pr[output^\pi(x, y, n) \neq f(x, y)] \leq \epsilon(n)$$

(Privacy) There exist probabilistic-polynomial time S_1 and S_2 such that

$$\{(S_1(1^n, x, f_1(x, y))\}_{x,y,n} \stackrel{c}{\equiv} \{(view_1^\pi(x, y, n)\}_{x,y,n},$$
$$\{(S_2(1^n, y, f_2(x, y))\}_{x,y,n} \stackrel{c}{\equiv} \{(view_2^\pi(x, y, n)\}_{x,y,n},$$

2.4 Private Set Intersection

Private set intersection (PSI) enables two parties, each holding a private set of elements, to compute the intersection of the two sets while revealing nothing more than the intersection itself.

The guarantees of PSI are captured in the ideal functionality $\mathcal{F}_{\mathsf{PSI}}$ defined in Figure 1.

Parameters: Sender S, receiver R, set sizes m, n.

Functionality:

- Obtain inputs $X = \{x_1, \cdots, x_m\} \subseteq \{0, 1\}^*$ from sender S.
- Obtain inputs $Y = \{y_1, \cdots, y_n\} \subseteq \{0, 1\}^*$ from receiver R.
- Give output $X \cap Y$ to receiver R.

Figure 1: Ideal functionality $\mathcal{F}_{\mathsf{PSI}}$ for PSI

2.5 Pseudorandom Functions from DDH Assumption

[NPR99] presented a construction of standard pseudorandom functions based on the DDH Assumption known as HashDH as below.

- $\mathsf{Setup}(1^\kappa)$: runs $\mathsf{GroupGen}(1^\kappa) \to (\mathbb{G}, g, p)$, picks a cryptographic hash function H from domain D to $\mathbb{G}$, outputs $pp = (\mathbb{G}, g, p, \mathsf{H})$. pp defines a family of functions from $\mathbb{Z}_p \times D$ to $\mathbb{G}$, which takes $k \in \mathbb{Z}_p$ and $x \in D$ as input and outputs $F_k(\mathsf{H}(x)) = \mathsf{H}(x)^k$.
- $\mathsf{KeyGen}(pp)$: outputs $k \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$.
- $\mathsf{Eval}(k, x)$: on input $k \in \mathbb{Z}_p$ and $x \in D$, outputs $\mathsf{H}(x)^k$.

Theorem 2.1. *$F_k(\mathsf{H}(x))$ is a family of PRF assuming H is a random oracle and the DDH assumption holds w.r.t. $\mathsf{GroupGen}(1^\kappa) \to (\mathbb{G}, g, p)$.*

It's not so indiscoverable that removing the cryptographic hash function H modeled as random oracle, the aforementioned standard PRFs construction is sharply degraded to a construction of weak PRFs as follows.

- $\mathsf{Setup}(1^\kappa)$: runs $\mathsf{GroupGen}(1^\kappa) \to (\mathbb{G}, g, p)$, outputs $pp = (\mathbb{G}, g, p)$. pp defines a family of functions from $\mathbb{Z}_p \times \mathbb{G}$ to $\mathbb{G}$, a.k.a. on input $k \in \mathbb{Z}_p$ and $x \in \mathbb{G}$ outputs $F_k(x) = x^k$.
- $\mathsf{KeyGen}(pp)$: outputs $k \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.
- $\mathsf{Eval}(k, x)$: on input $k \in \mathbb{Z}_p$ and $x \in D$, outputs $y \leftarrow x^k$.

Theorem 2.2. *$F_k(x)$ is a family of weak pseudorandom functions assuming the hardness the DDH assumption holds w.r.t.* $\mathsf{GroupGen}(1^\kappa) \to (\mathbb{G}, g, p)$.

3 Multi-Point OPRF from Blind-Query-Unblind Paradigm

Definition 3.1. A two-party protocol π is said to be amenable to the Blind-Query-Unblind Paradigm in Figure 2 with respect to some functionality F if the following properties hold.

(Correctness) π computes OPRF functionality $f(k, (x_1, \cdots, x_n)) = (\Lambda, (F(k, x_1), \cdots, F(k, x_n))$ correctly. This becomes immediate when the Blind-Unblind transformation possesses function F's input homomorphism as $Unblind(F(k, Blind(x, r)), r) = F(x)$.

In this paper, what we focus on is the case when F is some weak PRF family. Notice F can't be a standard PRF because standard pseudorandomness denies input homomorphism. Specifically speaking, if F_k is some PRF family and satisfies $Unblind(F_k(Blind(x, r)), r) = F_k(x)$, then exists adversary $\mathcal{A}$ querying oracle with inputs $\{x_1, x_2 = Blind(x_1, r)\}$ with some random r and receiving answers as $\{y_1, y_2\}$. If $Unblind(y_2, r) = y_1$, then $\mathcal{A}$ outputs '1', otherwise $\mathcal{A}$ outputs '0'. Clearly, A breaks the definition of PRF with advantage $1/2 - \mathsf{negl}(n)$. In contrast, the definition of weak PRF claims the inputs of oracle are uniformly chosen from D by the challenger instead of adversarially chosen by $\mathcal{A}$, so $\mathcal{A}$ has no chance to query some related points which admits weak PRF possessing some potential algebraic structure.

(Server's privacy) The simulator $\mathsf{Sim}_\mathcal{R}$ simulates the view of corrupt client which consists of client's randomness, input, output and received messages, then for any inputs k and $\{x_1, x_2, \cdots, x_n\}$ the simulated view is indistinguishable from the real execution in the protocol π.

Hybrid_0: C's view in the real protocol.

Hybrid_1: Given C's input $\{x_1, x_2, \cdots, x_n\}$ and output $\{F(k, x_1), F(k, x_2), \cdots, F(k, x_n)\}$, $\mathsf{Sim}_\mathcal{R}$ chooses the randomness $r_1, r_2, \cdots, r_n$ for C, and simulates the received messages with $\{Blind(F(k, x_1), r_1), Blind(F(k, x_2), r_2), \cdots, Blind(F(k, x_n)\}$.

Clearly, $\mathsf{Sim}_\mathcal{R}$'s simulated view in Hybrid_1 is identical to C's real view on the premise correctness holds as $F(x_i) = Unblind(F(k, Blind(x_i, r_i)), r_i)$ deriving $Blind(F(x_i), r_i) = Blind(Unblind(F(k, Blind(x_i, r_i)), r_i), r_i) = F(k, Blind(x_i, r_i)) = F(k, a_i)$.

(Client's privacy) The simulator $\mathsf{Sim}_\mathcal{S}$ simulates the view of corrupt server which consists of server randomness, input, output and received messages, then for any inputs k and $\{x_1, x_2, \cdots, x_n\}$ the simulated view is indistinguishable from the real execution in the protocol π.

Hybrid_0: S's view in the real protocol.

Hybrid_1: Given S's input k and no output, $\mathsf{Sim}_\mathcal{S}$ simulates the received messages with uniformly sampled elements $u_1, \cdots, u_n$.

When proving π is secure against corrupted server S, the simulator $\mathsf{Sim}_\mathcal{S}$ has to simulate the first and the only received messages $\{a_1, a_2, \cdots, a_n\}$ with no information but the given input k, which has nothing to do with $\{a_1, a_2, \cdots, a_n\}$ at all, so the only strategy $\mathsf{Sim}_\mathcal{S}$ can take is to output uniformly sampled elements, since the only konwledge k is of no use to itself. Therefore, the client's privacy is preserved if and only if the messagees $\{a_1, a_2, \cdots, a_n\}$ and uniformly sampled elements $u_1, \cdots, u_n$ are at least computationally indistinguishable for any inputs $\{x_1, x_2, \cdots, x_n\}$, i.e. $\{a_1, a_2, \cdots, a_n\}$ is at least pseudorandom. What needs illustration is that this isn't equivalent to *Blind* is a pseudorandom function,

since client may preprocess it's inputs before blinding. If we composite the preprocessing and blinding as a whole function, denoted as $Blind^*$ sufficient for $Blind^*(x_i, r_i) = a_i$, then we can say $Blind^*$ is at least a pseudorandom function on the premise that $Blind^*$'s outputs remain the first messages in protocol and π preserves the client's privacy, and vice versa.

In summary, if a protocol π in Figure 2 with respect to some functionality F satisfys F's input homomorphism and blinding method's pseudorandom, then it is a protocol in Blind-Query-Unblind Paradigm securely computing $f(k, (x_1, \cdots, x_n)) = (\Lambda, (F(k, x_1), \cdots, F(k, x_n))$. Hereinafter we will pay our specific attention on constructing a multi-point OPRF protocol from the Blind-Query-Unblind Paradigm, where client first exploit a cryptographic hash function H modeled as a random oracle to map the inputs to some domain as the aforementioned input preprocessing and $F = F_k$ is some weak PRF family in Figure 3.

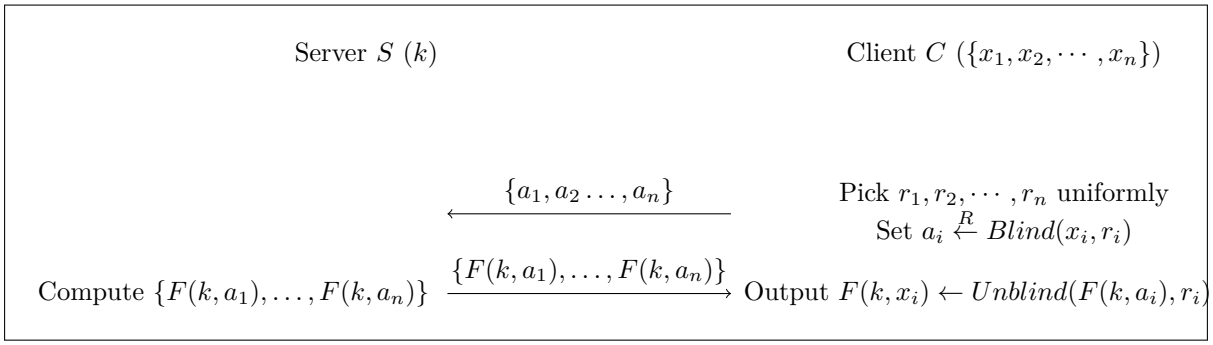

Figure 2: Blind-Query-Unblind Paradigm

Theorem 3.1. *If a two-party protocol π is a Blind-Query-Unblind Paradigm in Figure 3 with respect to some weak PRF family F_k and H modeled as a random oracle, then π must conforms to the definition of multi-point OPRF, a.k.a. π computes multi-point OPRF functionality $f(k, (x_1, \cdots, x_n)) = (\Lambda, (F_k^*(x_1), \cdots, F_k^*(x_n))$, where $F_k^* = F_k \circ \mathsf{H}$.*

Noticeably, $F_k^* = F_k(H(x))$ is standard PRF constructed by mapping the input to $\mathbb{G}$ via a cryptographic hash function H first, then apply F_k in a cascade way, yielding a composite function $F_k \circ \mathsf{H}$. By leveraging the programmability of H, we can easily reduce the pseudorandomness of the composite function F_k^* to the weak pseudorandomness of F_k. In other words, random oracle amplifies weak pseudorandomness to standard pseudorandomness. Specifically, $\mathcal{B}$ be an adversary against the weak pseudorandomness of F_k. $\mathcal{B}$ interacts with an adversary $\mathcal{A}$ in the pseudorandomness experiment, with the aim to determine if $\mathcal{B}$'s' given oracle is real or random oracle, simulates the random oracle H and $\mathcal{A}$'s real-or-random oracle as below:

Setup: $\mathcal{B}$ initializes two empty table $\mathsf{T_1}, \mathsf{T_2}$ and obtains polynomial many time input-ouput pair from its own oracle $\mathcal{O}_\mathcal{B}$ denoted as $\{(y_1, z_1), (y_2, z_2), \cdots\}$ stored in $\mathsf{T_2}$

Random oracle query: For random oracle query $\langle x_i \rangle$ from $\mathcal{A}$, $\mathcal{B}$ programs $\mathsf{H}(x_i) := y_i$, and then stores the key-value pair (x_i, y_i) in table $\mathsf{T_1}$.

Real-or-random query: Once receiving a query $\langle x_i \rangle$ to oracle $\mathcal{O}_\mathcal{A}$ from $\mathcal{A}$, $\mathcal{B}$ looks up $\mathsf{T_1}$ and if x_i is in $\mathsf{T_1}$, then looks up $\mathsf{T_2}$ to find the output z_i of its corresponding value y_i, sends $\langle z_i \rangle$ as the answer of $\langle x_i \rangle$ to $\mathcal{A}$

Guess: $\mathcal{A}$ makes a guess $\beta' \in \{0, 1\}$ for β, where '0' indicates real mode and '1' indicates random mode. $\mathcal{B}$ forwards β' to its own challenger.

Clearly, if $\mathcal{A}$ simulates the real oracle for all $i \in [n]$, then $\mathcal{B}$ simulates the real oracle. If $\mathcal{A}$ simulates the random oracle, then $\mathcal{B}$ simulates the random oracle. Thereby, $\mathcal{B}$ breaks the weak pseduorandomness of $F_k(\cdot)$ with the same advantage as $\mathcal{A}$ breaks the pseduorandomness of $F_k^*(\cdot)$.

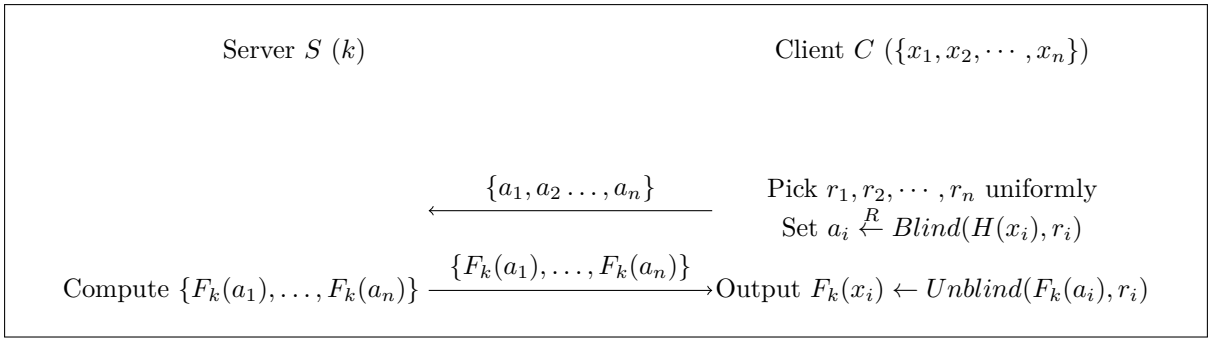

Figure 3: multi-point OPRF from Blind-Query-Unblind Paradigm

4 Multi-Point OPRF from two Blindings on DDH Assumption

4.1 Hashed DH multi-point OPRF using Exponential Blinding

In several of applications, the underlying blinding function in Blind-Query-Unblind Paradigm is instantiated with the Hashed Diffie-Hellman (HashDH) construction with $r_1 = r_2 = \cdots = r_n = r$, consistent with the underlying PRFs:

$$F_r(x) = H(x)^r$$

where hash functions H is defined as $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ for a multiplicative group $\mathbb{G}$ of prime order q, and the PRF key r is a random element in $\mathbb{Z}_q$. The protocol for the oblivious computation of HashDH PRFs employing this so-called exponential blinding method, also known as Exp-HashDH, is shown in Figure 4. Client C sends to server S its input x blinded as $a_i = (H(x_i))^r$, for the same randomness $r \xleftarrow{R} \mathbb{Z}_q$, and then unblinds the server's response $F_k(a_i) = {a_i}^k$ as $F_k(x_i) = F_k(a_i)^{1/r} = ({a_i}^k)^{1/r} = (((H(x_i))^r)^k)^{1/r} = (H(x_i))^k$ and outputs $F_k(x_i)$. It is easy to see that the client's inputs through blinding is computationally indistinguishable from random elements in $\mathbb{G}$ for server according to the definition of PRF F_k, namely the client's privacy in protocol is preserved.

Theorem 4.1. *If the DDH Assumption holds and* H *is a random oracle, then the protocol in Figure 4 securely realizes multi-point OPRF functionality for the PRF family* $F_k^*(x) = F_k(\mathsf{H}(x)) = \mathsf{H}(x^k)$*, where* F_k *is a weak PRF.*

It is evident that the protocol in Figure 4 follows Definition 3.1 since $Unblind(F_k(Blind(\mathsf{H}(x_i), r_i)), r_i)$ $= ((\mathsf{H}(x_i)^r)^k)^{-r} = \mathsf{H}(x_i)^k = F_k(\mathsf{H}(x))$ and $a_i = H(x_i)^r$ is a PRF. Combined with $F_k(x) = x^k$ is a weak PRF and H is a random oracle, the protocol conforms to Theorem 3.1, hence is qualify for multi-point OPRF. The details are listed as following.

4.2 Hashed DH multi-point OPRF using Multiplicative Blinding

An alternative multiplicative blinding technique, denoted Mult-HashDH, is shown in Figure 5. The protocol is an equivalent of Chaum's technique for blinding RSA signatures: Given generator g of group $\mathbb{G}$, the client blinds its input as $a_i = H(x_i) \cdot g^{r_i}$ with different randomness r_i,then unblinds the server's response $F_k(a_i) = {a_i}^k$ as $F_k(x_i) = F_k(a_i) \cdot (g^k)^{r_i} = {a_i}^k \cdot (g^k)^{r_i} = (H(x_i) \cdot g_i^r)^k \cdot g^{r_i k} = (H(x_i))^k$ using the public key g^k corresponding to the PRF key k, which is received from server. Since r_i is random elements in $\mathbb{Z}_q$, g^{r_i} is random elements in $\mathbb{G}$, leading multiplicative blinding to be a One-Time Pad in nature. Therefore, it is easy to see that this blinding hides the client's inputs with perfect security.

Comparing the computational cost of the two blinding techniques, we discover n variable-base exponentiations are required in both cases for the server. However, for the client, Exp-HashDH requires $2n$ variable-base exponentiations for blinding and unblinding, while Mult-HashDH involves n fixed-base exponentiations for blinding and n variable-base exponentiations for unblinding. In applications the client could store the public key g^k, which allows the latter exponentiations to employ fixed-base optimization, reducing the client's total computation to $2n$ fixed base exponentiations. Given that exponentiation

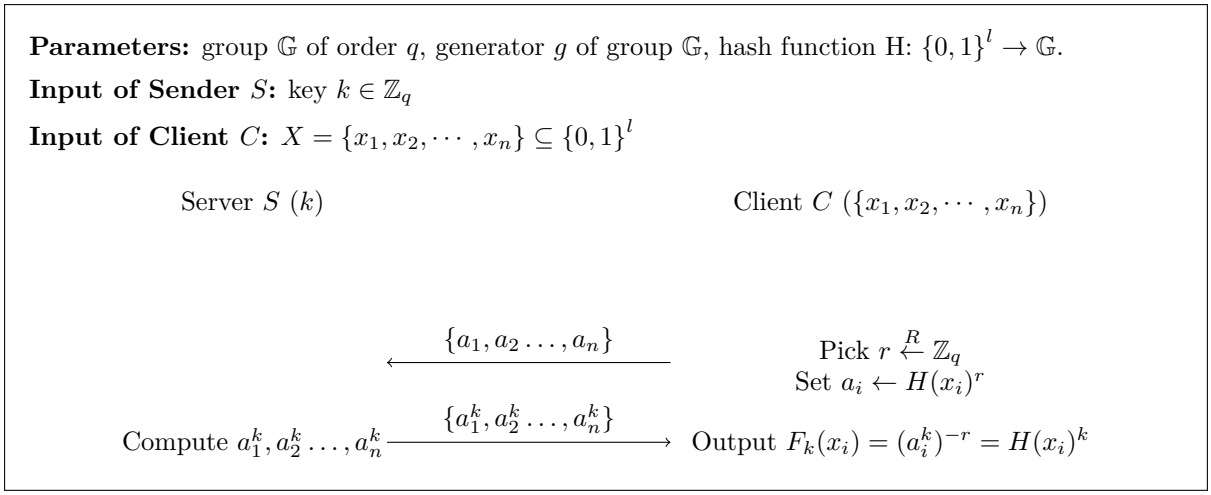

Figure 4: Hashed DH multi-point OPRF using Exponential Blinding

with a fixed base gives an efficiency about 6-7 times faster than that with a variable base, Mult-HashDH becomes at least 1.7 faster than Exp-HashDH and 6x faster if g^k is stored at the client and treated as a fixed base. [JKX21] The concrete contrast of theoretical computational costs between Exp-HashDH OPRF/PSI and Mult-HashDH OPRF/PSI protocols as Table 1.

Although there is good news for performance in multiplicative blinding, regarding security, things are not as straightforward. It is almost impossible to prove that Mult-HashDH multi-point OPRF realizes security in the definition of (strong) OPRF functionality, in consideration of the protocol slightly deviating from multi-point OPRF from Blind-Query-Unblind Paradigm in 3 as the client reserving the public key g^k from server, which actually involves some information about the PRF key k as the server's input, and yields that the simulator for corrupted client $\mathsf{Sim}_{\mathcal{R}}$ can't simulate the message g^k with no konwledge about the PRF key k, which is nevertheless available to the distinguisher as the indices of the joint distribution. We break the ice in two steps: First, we replace the generator g with a uniformly generated element h on group $\mathbb{G}$ by client C and send to server along with blinding messages $a_1, a_2, \cdots, a_n$, and server still behave as a oracle returning weak PRF value of inputs as before, including h^k. We can thereby view the protocol in Figure 6 as appending a dummy input $x_{n+1} = \Lambda$, and assuming x_{n+1} is transformed to $a_{n+1} = h$ after blinding. Since h is uniformly sampled from $\mathbb{G}$, so this minor amendment doesn't undermine the (pseudo)randomness of blinding method and maintains the pseudorandomness of the output of weak PRF $F_k(x) = x^k$ as well. Ultimately, the protocol in Figure 6 can perfectlt fit in our universal paradigm in Figure 3.1, which simplifies our proof substantially; However, there still exists a problem that in simulating the view of corrupt client the simulator $\mathsf{Sim}_{\mathcal{R}}$ can't make up message h^k by obtaining its corresponding output on account of its corresponding dummy input. The last resort we use is the pseudorandomness of h^k which built on the distinguisher's ignorance of PRF key k nevertheless. So we introduce the concept of relaxed OPRF, loosing up the security of server's privacy by employing the augmented points as the server's inputs instead of k, while placing k on the server's random tape, which is invisible for both $\mathsf{Sim}_{\mathcal{R}}$ and distinguisher. In other words, relaxed OPRF doesn't require so rigid security for the server's PRF key k and gives opportunity to send messages covering information of k to client. Therefore, it can only be certifiable that Mult-HashDH multi-point OPRF meets security in the definition of relaxed OPRF functionality, which is already adequate for most applications as well.

Theorem 4.2. *If the DDH Assumption holds and* H *is a random oracle, then the protocol in Figure 6 securely realizes multi-point relaxed OPRF functionality for the PRF family* $F_k^*(x) = F_k(\mathsf{H}(x)) = \mathsf{H}(x^k)$*, where* F_k *is a weak PRF.*

(Correctness) $Unblind(F_k(Blind(\mathsf{H}(x_i), r_i)), r_i) = ((\mathsf{H}(x_i) \cdot h^{r_i})^k) \cdot (h^k)^{-r_i} = \mathsf{H}(x_i)^k = F_k(\mathsf{H}(x))$.

(Server's privacy) In accordance with the definition of server's privacy in Definition 2.3, the augmented protocol of Hashed DH multi-point rOPRF using Multiplicative Blinding in Figure 6 is exhibited in the Figure 7. Then we will construct a simulator $\mathsf{Sim}_{\mathcal{R}}$ simulating the view of corrupt client which consists of

client's randomness, input, output and received messages, so that for any inputs k and $\{x_1, x_2, \cdots, x_n\}$ the simulated view is indistinguishable from the real execution in the protocol π via a sequence of hybrid transcripts.

Hybrid_0: C's view in the real protocol.

Hybrid_1: Given C's input $\{x_1, x_2, \cdots, x_n\}$ and output $\{F_k^*(x_1), F_k^*(x_2), \cdots, F_k^*(x_n)\}$, $\mathsf{Sim}_{\mathcal{R}}$ chooses the randomness $r_1, r_2, \cdots, r_n \xleftarrow{R} \mathbb{Z}_q$ and $h \xleftarrow{R} \mathbb{G}$ for C, and simulates the received messages with $\{F_k^*(x_1) \cdot h^{kr_1}, F_k^*(x_2) \cdot h^{kr_2}, \cdots, F_k^*(x_n) \cdot h^{kr_n}, h^k\}$ and $\{H(y_1)^k, H(y_2)^k \ldots, H(y_m)^k\}$ with the konwledge of Y and k.

Clearly, $\mathsf{Sim}_{\mathcal{R}}$'s simulated view in Hybrid_1 is identical to C's real view because $F_k^*(x_i) \cdot h^{kr_i} = a_i^k \cdot (h^k)^{-r_i} \cdot h^{kr_i} = a_i^k$.

Hybrid_2: Given C's input $\{x_1, x_2, \cdots, x_n\}$ and output $\{F_k^*(x_1), F_k^*(x_2), \cdots, F_k^*(x_n)\}$, $\mathsf{Sim}_{\mathcal{R}}$ chooses the randomness $r_1, r_2, \cdots, r_n \xleftarrow{R} \mathbb{Z}_q$ and $h \xleftarrow{R} \mathbb{G}$ for C, and simulates the received messages with $\{F_k^*(x_1) \cdot u^{r_1}, F_k^*(x_2) \cdot u^{r_2}, \cdots, F_k^*(x_n) \cdot u^{r_n}, u\}$ and $\{u_1, u_2 \ldots, u_m\}$ without the konwledge of Y and k, where $u_1, u_2, \cdots, u_m, u \xleftarrow{R} \mathbb{G}$.

We argue that the simulated view in Hybrid_2 and Hybrid_3 are computationally indistinguishable by reducing it to the weak pseudorandomness of the function $F_k(x) = x^k$. Specifically, suppose the distinguisher $\mathcal{D}$ can successfully distinguishes Hybrid_2 and Hybrid_3, then we can demonstrate there exists an simulator $\mathcal{S}$ which can easily determine if the given oracle is real or random oracle computing F_k i.e. break the weak pseudorandomness of F_k, and it simulates the random oracle H and generates $\mathcal{D}$'s challenge as below:

Setup: Given polynomial many time uniformly random points and relevant outputs denoted as $\{(s_0, t_0), (s_1, t_1), (s_2, t_2) \cdots (s_m, t_m)\}$ from real-or-random oracle, $\mathcal{S}$ shares the common konwledge with the distinguisher $\mathcal{D}$, including two parties' inputs $\{x_1, x_2, \cdots, x_n\}$, $\{y_1, y_2, \cdots, y_m\}$ as the indices of the joint distribution, and client's output $\{F_k^*(x_1), F_k^*(x_2), \cdots, F_k^*(x_n)\}$.

Simulation: Given C's input $\{x_1, x_2, \cdots, x_n\}$ and output $\{F_k^*(x_1), F_k^*(x_2), \cdots, F_k^*(x_n)\}$, $\mathcal{S}$ chooses the randomness $r_1, r_2, \cdots, r_n \xleftarrow{R} \mathbb{Z}_q$ and $h = s_0$ for C, and simulates the received messages with $\{F_k^*(x_1) \cdot t_0^{r_1}, F_k^*(x_2) \cdot t_0^{r_2}, \cdots, F_k^*(x_n) \cdot t_0^{r_n}, t_0\}$ and $\{t_1, t_2 \ldots, t_m\}$.

Random oracle query: For random oracle query $y_i \in Y$ from $\mathcal{D}$, $\mathcal{S}$ programs $\mathsf{H}(y_i) := s_i$.

If $t_i = F_k(s_i) = s_i^k$ for $i = 0, 1, \cdots, m+1$, then $\mathcal{S}$'s simulation is identical to Hybrid_1. If t_i are random values, then $\mathcal{S}$'s simulation is identical to Hybrid_2. Therefore, $\mathcal{D}$ distinguishes Hybrid_1 and Hybrid_2 with the same advantage as $\mathcal{S}$ breaks the weak pseduorandomness of F_k. In view of F_k is a weak PRF, Hybrid_1 and Hybrid_2 are computationally indistinguishable, then so as Hybrid_0 and Hybrid_2, which proves the server's privacy.

(Client's privacy) The simulator $\mathsf{Sim}_{\mathcal{S}}$ simulates the view of corrupt server which consists of server randomness, input, output and received messages, then for any inputs k and $\{x_1, x_2, \cdots, x_n\}$ the simulated view is indistinguishable from the real execution in the protocol π.

Hybrid_0: S's view in the real protocol.

Hybrid_1: Given S's input k and no output, $\mathsf{Sim}_{\mathcal{S}}$ simulates the received messages with uniformly sampled elements $u_1, \cdots, u_n$.

In consideration of $r_1, r_2, \cdots, r_n \xleftarrow{R} \mathbb{Z}_q$, $h^{r_1}, h^{r_2}, \cdots, h^{r_n}, h \xleftarrow{R} \mathbb{G}$ play the role of One-Time Pad and above two hybrids are perfect indistinguishable.

Party	Basic Operation	Exp-HashDH OPRF	Mult-HashDH OPRF	Exp-HashDH PSI	Multi-HashDH PSI
Server	variable-base exponentiation	n	n	$2 \times n$	$2 \times n$
Client	variable-base exponentiation	$2 \times n$	–	$2 \times n$	–
Client	fixed-base exponentiation	–	$2 \times n$	–	$2 \times n$

Table 1: Theoretical communication costs between Exp-HashDH OPRF/PSI and Mult-HashDH OPRF/PSI protocols

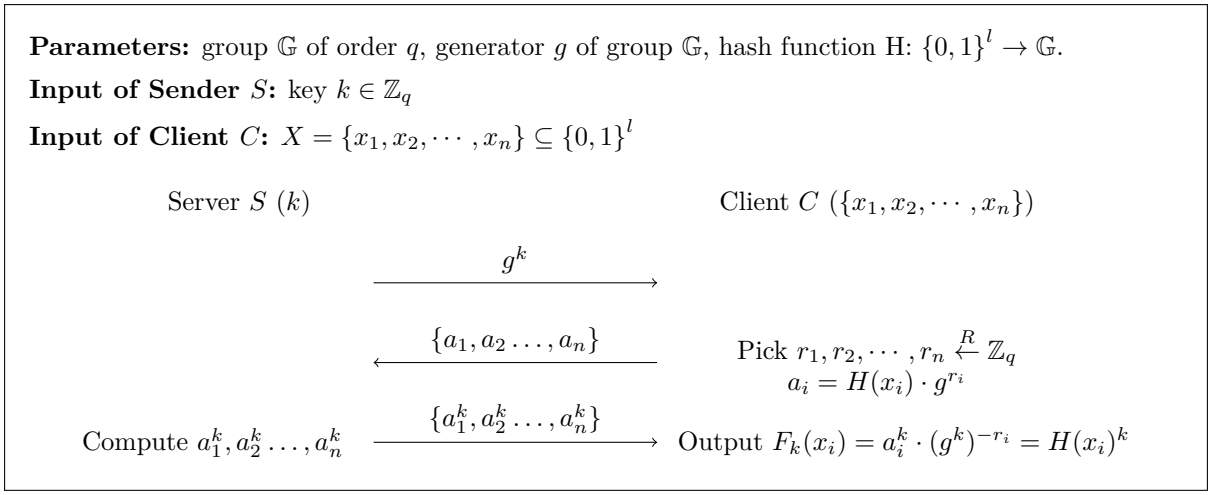

Figure 5: Preceding Hashed DH multi-point OPRF using Multiplicative Blinding

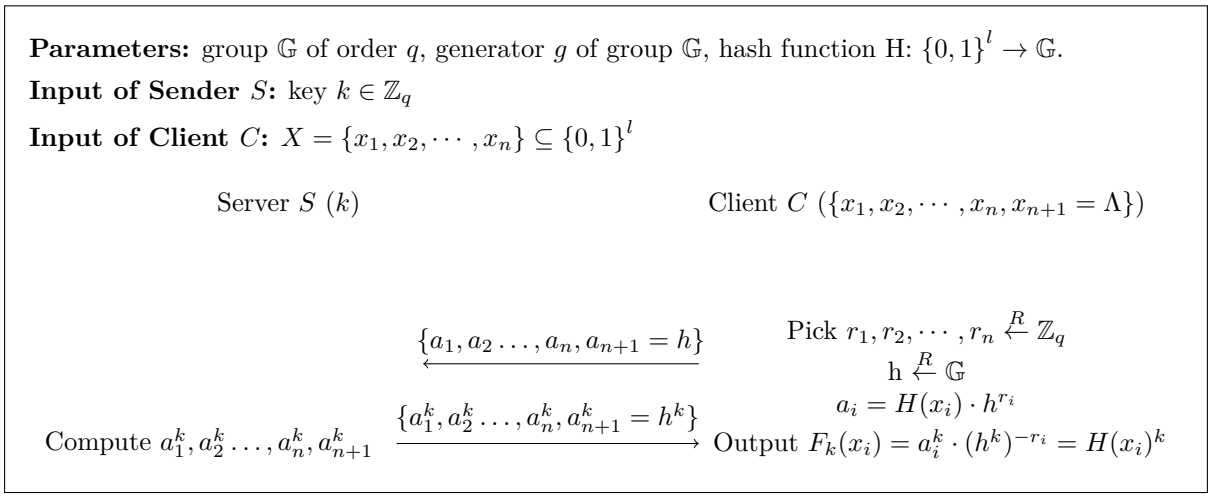

Figure 6: Our improved Hashed DH multi-point rOPRF using Multiplicative Blinding

5 PSI from Multi-Point Relaxed OPRF

A multi-point (non-adaptive, n-time) relaxed OPRF can naturally generate a PSI protocol: Consider a PSI protocol π_{psi} in which the inputs of S consists of m elements $y_1, \cdots, y_m$ and the inputs of C consists of n elements $x_1, \cdots, x_n$. Protocol π_{psi} proceeds as follows: (1) S picks a key k at random; (2) S, C invoke a multi-point (non-adaptive, n-time) relaxed OPRF protocol π on inputs $(k, (x_1, \cdots, x_n))$ with corresponding augmented protocol $\pi^{'}$ computing functionality $g((y_1, \cdots, y_m), (x_1, \cdots, x_n)) = ((F_k(y_1), \cdots, F_k(y_m), (F_k(x_1), \cdots, F_k($ (3) After the execution of multi-point relaxed OPRF, S can calcute the pseudorandom function values on its inputs, i.e. $(F_k(y_1), ..., F_k(y_m))$, while C obtains the PRF values $(F_k(x_1), ..., F_k(x_n))$ on its inputs, sharing the same values when elements are in the intersection. (4) S sends all its PRF values to C, C compares received values with its own values. If there exists equivalent element, then it is in the intersection.

Theorem 5.1. *If protocol π is a multi-point (non-adaptive, n-time) relaxed OPRF with the augmented protocol $\pi^{'}$ w.r.t. functionality $g((y_1, \cdots, y_m), (x_1, \cdots, x_n)) = ((F_k(y_1), \cdots, F_k(y_m), (F_k(x_1), \cdots, F_k(x_n))$, then the above-mentioned protocol securely realizes $\mathcal{F}_{\mathsf{PSI}}$ in Figure 8.*

(Correctness) The above protocol is correct except the case $F_k(x_i) \in B = \{F_k(y_1), F_k(y_2) \ldots, F_k(y_m)\}$ while $x_i \notin Y$, whose probability of occurrence is $mn \cdot 2^{-l}$.

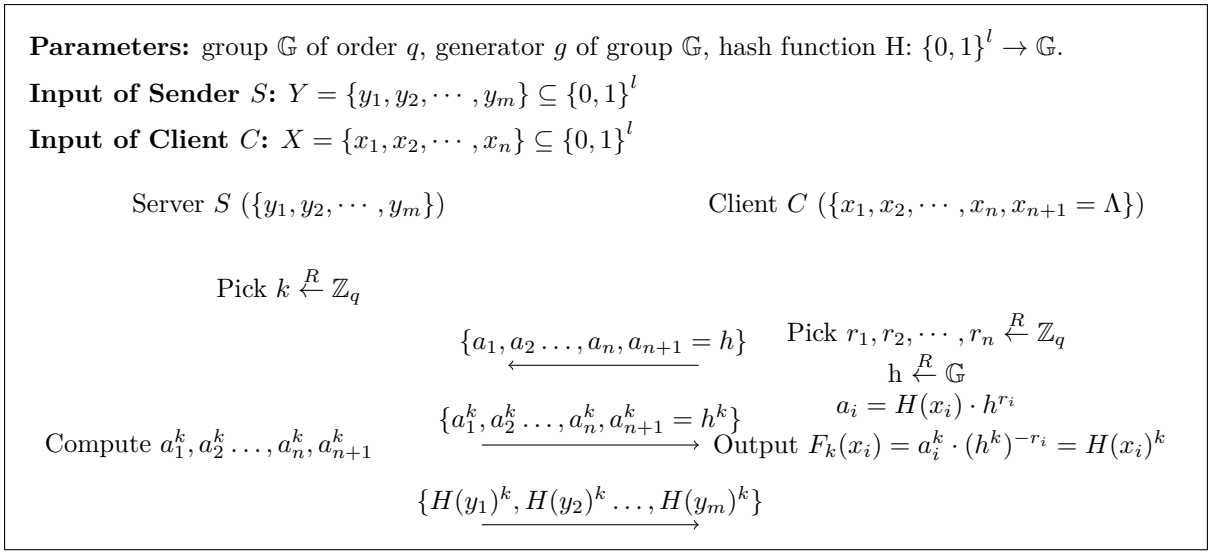

Figure 7: the augmented protocol of Hashed DH multi-point rOPRF using Multiplicative Blinding

(Server's privacy) On the grounds of the definition of rOPRF in Definition 2.3, the server's privacy of the augmented protocol of multi-point rOPRF immediately implies the server's privacy of the composite protocol PSI.

(Client's privacy) Without any additional incoming message of server S, the client's privacy of multi-point rOPRF immediately implies the client's privacy of the composite protocol PSI.

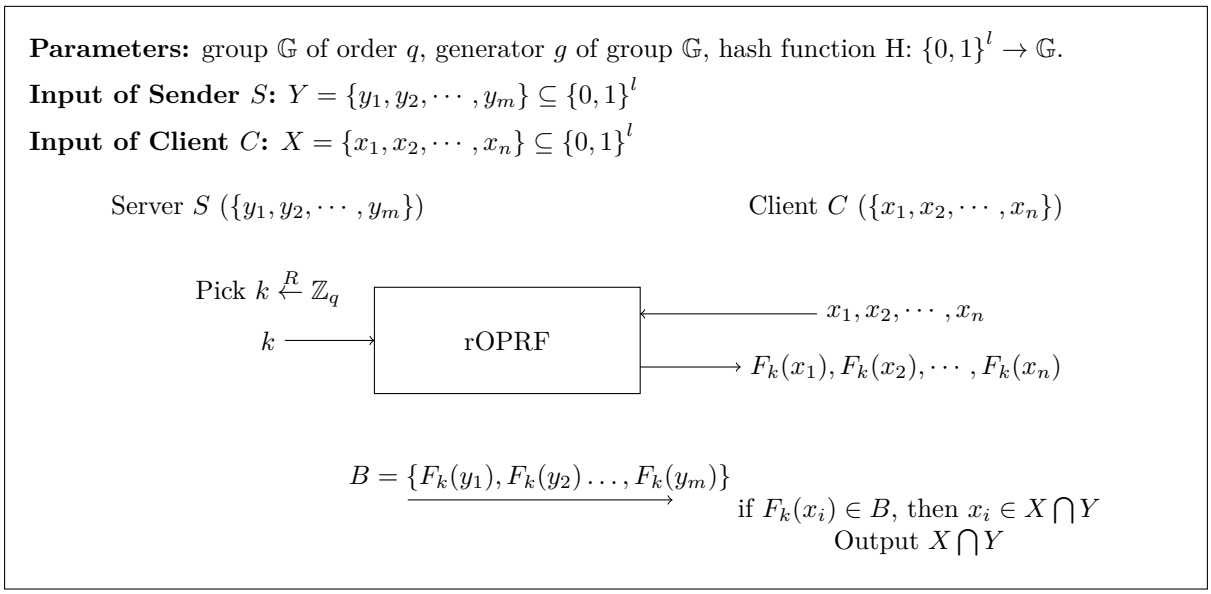

Figure 8: PSI from multi-point rOPRF

Optimization: The naive protocol above sending all PRF values to the client and leaving it to figure out which element is in the intersection is not an efficient approach for client, which requires about $O(n^2)$ compare operations asymptotically. A frequent optimization is to ask the server to hash all its PRF values in a hash table ahead of time, and then send the hash table to the client. This technique increases the server's computational cost for hashing its n elements to hash table, but reduces the client's computation

complexity to $O(n)$ asymptotically. Another optimization for the sake of reducing the communication cost is to leverage some data structures for member tests, such as Bloom Filter, Cuckoo Filter, by making the server insert its PRF values into these data structures and send the filter out instead of sending them straightforward.

6 Performance

We describe details of our implementation and report the performance of the following set operations: (1) **psi**: intersection of the sets; (2) **card**: cardinality of the intersection; (3) **card-sum**: sum of the associated values for every item in the intersection with cardinality; (4) **psu**: union of the sets; (5) **private-ID**: a universal identifier for every item in the union. We compare our work with the current fastest known protocol implementation for each functionality.

6.1 Implementation Details

Our protocols are written in C++ with detailed documentations, which can be found at https://github.com/yuchen1024/Kunlun/mpc. In consistency with our paper, we implement our protocols in a modular fashion. We first implement the core mqRPMT protocol, then build various PSO protocols upon it. Our implementation only relies on the OpenSSL library [**?**], and it can smoothly run on both Linux and MacOS. In contrast, most existing PSO programs rely on multiple libraries and require sophisticated parameters tuning, while sometimes the optimized parameters setting are not explicitly given. Thereby, even running these programs successfully on the same environment would require tremendous efforts.

PSI	Running time (s)						Comm. (MB)		
	LAN			WAN			total		
	2^{12}	2^{16}	2^{20}	2^{12}	2^{16}	2^{20}	2^{12}	2^{16}	2^{20}
Exp-HashDH PSI	5.07	80.89	1296.67	5.27	81.79	1355.42	0.30	4.74	76.60
Mult-HashDH PSI	0.58	6.43	98.259				0.27	4.35	69.60

Table 2: Practical communication cost and running time of two DH PSI protocols, calculated using computational security $= 128$ and statistical security $= 40$. Each item has 128-bit length.

References

[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.

[IKN+19] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. *IACR Cryptol. ePrint Arch.*, 2019:723, 2019.

[JKX21] Stanisław Jarecki, Hugo Krawczyk, and Jiayu Xu. On the (in)security of the diffie-hellman oblivious prf with multiplicative blinding. In Juan A. Garay, editor, *Public-Key Cryptography – PKC 2021*, pages 380–409, Cham, 2021. Springer International Publishing.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999.

[RT21] Mike Rosulek and Ni Trieu. Compact and malicious private set intersection for small sets. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1166–1181, New York, NY, USA, 2021. Association for Computing Machinery.