Private Set Operations from Multi-Query Reverse Private Membership Test

Abstract

Private set operations allow two parties perform secure computation on two private sets, such as intersection or union related functions. In this paper, we identify a framework for performing private set operations. At the technical core of our framework is multi-query reverse private membership test (mqRPMT), which is a natural extension of RPMT recently proposed by Kolesnikov et al. [KRTW19]. In mqRPMT, a client with elements vector $(x_1, \ldots, x_n)$ interacts with a server holding a set Y. As a result, the server only learns a bit vector $(e_1, \ldots, e_n)$ indicating whether $x_i \in Y$ but without knowing the value of x_i, while the client learns nothing. We first show how to build mqRPMT from a new cryptographic primitive called commutative weak pseudorandom function (cwPRF), which in turn can be instantiated from the decisional Diffie-Hellman like assumptions in the random oracle model. We then show a relaxed version of mqRPMT can be build from a category of mqPMT (which in turn can be based on homomorphic encryption), making the first step towards establishing the relation between the two primitives.

We demonstrate the practicality of our framework with an implementation. By plugging our wcPRF-based mqRPMT to the general framework, we obtain competitive PSU protocol. For input sets of size 2^{20}, the resulting PSU protocol requires roughly 100 MB bandwidth, and 200 seconds using a single thread. To the best of our knowledge, it requires the least communication among all the known PSU protocols. By plugging our FHE-based mqRPMT* to the general framework, we obtain a PSU* suitable for unbalanced setting, whose communication complexity is linear in the size of the smaller set, and logarithmic in the larger set.

Keywords: PSO, PSU, commutative weak PRF, multi-query RPMT

Contents

1 Introduction

Private set operation enables two parties, a sender (P_1) and a receiver (P_2), with respective input sets X and Y to compute the intersection $X \cap Y$ or union $X \cup Y$ without revealing anything else. When measuring the efficiency of a PSO protocol, there are two major metrics. The first is *computation cost*, which is the amount of computing time necessary to run the protocol. Optimizing the computation cost is especially important in practice because of limited computational resources. The second is *communication cost*, which is the total amount of communication in the protocol. Minimizing the communication cost is also crucial in practice due to limited network bandwidth. A hybrid metric is the total running time of the protocol, which includes both the computation time and the time to transmit and receive the messages. Recent works [PRTY19, IKN+20] suggest an alternative efficiency metric – the *monetary cost* to run the protocol on a cloud computing service. This new metric takes both computation cost and communication cost into consideration.

We briefly review PSO protocols in the semi-honest model as below. (add balanced and unbalanced here)

Private set intersection. PSI has found many applications including privacy-preserving sharing, private contact discovery, DNA testing and pattern matching. In the past two decades PSI has been extensively studied and has become truly practical with extremely fast implementation. We refer to [PSZ18] for a good survey of different PSI paradigms. State-of-the-art two party PSI protocols [KKRT16, PRTY19, CM20, GPR+21, RS21] mainly rely on symmetric-key operations, except a little public-key operations in base OT used in the OT extension protocol.

Private computing on the intersection. Many real-world applications requires only partial/aggregate information about the intersection to be revealed, such as PSI-card for intersection cardinality [HFH99] and PSI-sum for intersection sum [IKN+20, MPR+20].

Private set union. Like well-researched PSI, PSU also has numerous applications in practice, such as cyber risk assessment and management via joint IP blacklists and joint vulnerability data. There are two categories of existing PSU protocols. The first is mainly based on public-key techniques [KS05, Fri07, HN10, DC17]. The second is mainly based on symmetric-key techniques [KRTW19, GMR+21]. In contrast to the affairs of PSI, the efficiency of the state-of-the-art PSU is less satisfactory. None of existing protocols is optimal in the sense that both communication and computation complexity is linear in the size of sets.

PSI, PSI-card/PSI-sum and PSU are closely related functionalities. It is somewhat surprising that the state-of-the-art protocols for these functionalities have significantly different efficiency. In balanced setting, there is no PSU protocol with linear complexity in the literature. The simplest PSI-card is concretely about 20× slower and requires over 30× more communication than PSI. PSU is concretely about 20× slower and requires over 30× more communication than PSI. In unbalanced setting, there is no PSU protocol with sublinear complexity in the larger set in the literature. Why is this the case? We observe that the core primitive that underlies almost all PSI protocols is a protocol called multi-query private membership test (PMT), which has very efficient realizations. However, it seems that mqPMT does not readily implies PSI-card/PSI-sum or PSU. The reason is that mqPMT reveals information about intersection, which should be kept privately in PSI-sum/PSI-card and PSU.

1.1 Motivation

The above discussion indicates that the most efficient approach for PSI cannot be used in PSI-card/PSI-sum and PSU. Therefore, different approaches are employed in different private set operations, creating much more engineering effort. We are motivated to seek for the core protocol that enables all private set operations, with the hope to deploy PSO in a unified framework. Moreover, given the huge efficiency difference between PSI and other closely related protocols, we are motivated to give efficient construction of the core protocol to close the gap. In summary, we are motivated to answer the following questions:

Is there a core protocol that enables a unified framework for all private set operations? If so, can we give efficient constructions?

1.2 Our Contribution

In this work, we make positive progress on the aforementioned questions. We summarize our contribution as below.

A framework of PSO. We identify that multi-query reverse private membership test (mqRPMT) is a "Swiss army knife" for private set operations. More precisely, mqRPMT itself implies PSI-card; by coupling with OTe, mqRPMT implies PSI and PSU; by further coupling with secret sharing or additively homomorphic encryption, mqRPMT implies PSI-sum. Therefore, we can build a PSO framework central around mqRPMT, which can perform operations in a unified and flexible manner.

Efficient construction of mqRPMT. We propose a generic construction of mqRPMT with linear complexity from a new cryptographic primitive called commutative weak PRF, which in turn can be instantiated from the DDH like assumptions in the random oracle model. The construction is simple and enjoys linear communication and computation complexity. Note that the asymptotic complexity of our PSO framework is dominated by the underlying mqRMPT. Therefore, all protocols derived from our framework achieve linear complexity. Particularly, to the best of our knowledge, it is the first time to have PSU with linear complexity.

Relaxed mqRPMT. We proprose a relaxed version of mqRPMT (denoted mqRPMT* hereafter). Compared to the standard mqRPMT, mqRPMT* allows the sender learn the size of intersection. We show that mqRMPT* can be build from a special category of mqPMT in a black-box manner via the "permutate-then-test" recipe. This makes the initial step towards exploring the connection between mqRPMT and mqPMT. By instantiating the conversion with fully homomorphic encryption (FHE), we obtain an effcient mqRPMT* in unbalanced setting, which immediately gives rise to a PSU* protocol in unbalanced setting.

Evaluations. We implement our framework. The experimental results demonstrate that our PSU protocol is superior to all the known PSU protocols in terms of communication cost.

1.3 Technical Overview

PSO from mqRPMT. As discussed above, mqPMT that underlies leading PSI protocols is not applicable for computing PSU as well as other closely related function such as PSI-card and PSI-sum. Consquently, there is no unified framework for PSO. We examine the reverse direction, i.e., whether the core protocol underlying PSU can be used for computing PSI. We identify that the technical core beneath all the existing PSU protocols is mqRPMT, which is a generalization of RPMT proposed in [KRTW19]. Roughly speaking, mqRPMT is a two party protocol between a client with set X and a server S with set X. After execution of the protocol, the server learns an indication bit vector $(e_1, \ldots, e_n)$ such that $e_i = 1$ if and only if $x_i \in Y$ but without knowning x_i, while the client learns nothing. Superficially, mqRPMT is similar to mqPMT, except that the server but not the client learns the test results instead. This tiny difference turns out to be significant. To see this, note that in mqRPMT the information of intersection (except its cardinality) is hidden from both sides, while in mqPMT the intersection is finally known by the client. In light of this difference, mqRPMT is particular suitable for functionalities that have to keep intersection private. A PSU protocol is immediate by having the sender (play the role of client) and the receiver (play the role of server) invoke a mqRPMT protocol on the first place, then carrying out n one-out-two OT with e_i and $(\perp, y_i)$ respectively. PSI and other protocols such as PSI-card and PSI-sum can be constructed similarly by coupling with additively homomorphic encryption or secret sharing.

mqRPMT from cwPRFs. The seminal PSI protocol [Mea86] (related ideas were appeared in [Sha80, HFH99]) is based on the commutative properties of the DH function. After roughly four decades, the DH-based PSI protocol is still the most easily understood and communication efficient one among numerous PSI protocols. It is somewhat surprisingly that no counterpart is known in the PSU setting yet. An intriguing question is: Can DH strike back?

In this work, we give an affirmative answer. We propose a new cryptographic primitive called commutative weak PRF. Let $F : K \times D \to R$ be a family of weak PRFs, where $R \subseteq D$. We say F is commutative if for any $k_1, k_2 \in K$ and any $x \in D$, it holds that $F_{k_1}(F_{k_2}(x)) = F_{k_2}(F_{k_1}(x))$. In other words, the two composite functions $F_{k_1} \circ F_{k_2}$ and $F_{k_2} \circ F_{k_1}$ are essentially the same function, say, $\hat{F}$.

We then show how to build mqRPMT from cwPRFs. Let server and client generate cwPRF key k_1 and k_2 respectively, and both of them map their items to elements in the domain D of F via a

common cryptographic hash function H, which will be modeled as a random oracle. We begin with the construction of the basic single-query RPMT. We first observe that cwPRF immediately gives rise to a private equality test (PEQT) protocol. Suppose server with y and client with x, they perform PEQT via the following steps: (1) server computes and sends $F_{k_1}(\mathsf{H}(y))$ to client; (2) client computes and sends $F_{k_2}(\mathsf{H}(x))$ and $F_{k_2}(F_{k_1}(\mathsf{H}(y)))$ to server; (3) P_1 then learns the test result by comparing $F_{k_1}(F_{k_2}(\mathsf{H}(x))) =? F_{k_2}(F_{k_1}(\mathsf{H}(y)))$. The commutative property of F guarantee the correctness. The weak pseudorandomness of F guarantee that P_2 learns nothing and P_1 learns nothing beyond the test result. At a high level, $F_{k_2}(F_{k_1}(\mathsf{H}(\cdot))) = F_{k_1}(F_{k_2}(\mathsf{H}(\cdot))) = \hat{F}(\mathsf{H}(\cdot))$ serves as pseudorandom encoding in the joint view, while $F_{k_1}(\mathsf{H}(\cdot))$ and $F_{k_2}(\mathsf{H}(\cdot))$ serve as partial pseudorandom encoding in the views of server and client respectively.

But, direct extension of the above PEQT protocol by sending back $F_{k_2}(F_{k_1}(\mathsf{H}(y_i)))$ for each $y_i \in Y$ in the same order with server's first move message $F_{k_1}(\mathsf{H}(y_i))$ does not lead to a RPMT protocol. The reason is that $\{\hat{F}(\mathsf{H}(y_i))\}$ serves as an order preserving pseudorandom encoding of set Y. As a consequence, the server will learn the exact value of x if $x \in Y$. In order to perform the membership test in an oblivious manner, the idea is to make the pseudorandom encoding of Y independent of the order known by the server. The most straightforward approach is to permutate $\{\hat{F}(\mathsf{H}(y_i))\}$. In this way, we build a single query RMPT protocol from cwPRFs, and the resulting protocol can be easily batched to handle multiple queries by reusing the pseudorandom encoding of X. A simple calculation shows that the computation cost is $3n$ times evaluation of F and n times look up, and the communication cost is $3n$ elements in the range of F. The mqRPMT protocol is optimal in the sense that both computation and communication complexity is linear to the set size. A more efficient method is to insert $\{\hat{F}(\mathsf{H}(y_i))\}$ into an order hiding data structure such as the Bloom filter, instead of permuting them.

mqRPMT* from mqPMT. We first abstract a category of PMT protocol called amortized Sigma-PMT with stateless test. Such-PMT naturally give rises to mqPMT, which underlies many PSI protocols with linear complexity. Following the permute-then-recover approach, we can tweak such mqPMT to mqRPMT* with same asymptotic complexity.

Applications of multi-query RPMT. With multi-query RPMT in hand, we can build a general PSO framework. Multi-query RPMT itself immediately give rise to private set intersection/union cardinality. Coupling with oblivious transfer, we can obtain PSI, private set intersection sum or PSU, depending the messages on OT sender's side.

1.4 Related Works

2 Preliminaries

2.1 MPC in the Semi-honest Model

We use the standard notion of security in the presence of semi-honest adversaries. Let Π be a 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 2.1. 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.

2.2 Private Set Operation

PSO is a special case of secure two-party computation.

Parameters: size of sets n.

Functionality: On input $X = \{x_1, \ldots, x_n\} \subseteq \{0,1\}^\ell$ (and possibly $V = \{v_1, \ldots, v_n\}$) from the sender P_1 and $Y = \{y_1, \ldots, y_n\} \subseteq \{0,1\}^\ell$ from the receiver P_2:

- **intersection:** give $X \cap Y$ to the receiver P_2.
- **union:** give $X \cup Y$ to the receiver P_2.
- **union*:** give $|X \cap Y|$ to the sender P_1 and $X \cup Y$ to the receiver P_2.
- **intersection cardinality:** give $|X \cap Y|$ to the receiver P_2.
- **intersection sum with cardinality:** give $|X \cap Y|$ and $S = \sum_{i:x_i \in Y} v_i$ to the sender.

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

3 Protocol Building Blocks

3.1 Oblivious Transfer

Oblivious Transfer (OT) [Rab] is a central cryptographic primitive in the area of secure computation. 1-out-of-2 OT allows a sender with two input strings (m_0, m_1) and a receiver with an input choice bit $b \in \{0,1\}$. As a result of the OT protocol, the receiver learns m_b and neither party learns any additional information. Though expensive public-key operations is unavoidable for a single OT, a powerful technique called OT extension [IKNP03, KK13, ALSZ15] allows one to perform n OTs by only performing $O(\kappa)$ public-key operations (where κ is the computational security parameter) and $O(n)$ fast symmetric-key operations. In Figure 2 we formally define the ideal functionality for OT that provides n parallel instances of OT.

Parameters: number of OT instances n; string length ℓ.

Functionality: On input $\{(m_{i,0}, m_{i,1})\}_{i \in n}$ from the sender P_1 where each $m_{i,b} \in \{0,1\}^\ell$, and input $\vec{b} \in \{0,1\}^n$ from the receiver P_2:

- Give output $(m_{1,b_1}, \ldots, m_{n,b_n})$ to the receiver.

Figure 2: Ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for OT

3.2 Multi-Query RPMT

RPMT [KRTW19] refers to a protocol where the client with input x interacts with a server holding a set Y. As a result, the server learns (only) the bit indicating whether $x \in Y$, while the client learns nothing about the set Y. The default notion of RPMT allows the client to query for a single element. While this procedure can be repeated several times, one may seek more efficient solutions allowing the client to make n distinct queries at a reduced cost. This generalized notion of n-time RPMT is straightforward to define. Hereafter, we refer to n-time RPMT as multi-query RMPT. In Figure 3 we formally define the ideal functionality for mqRPMT. We also define a relaxed version of mqRPMT called mqRPMT*, in which the client is given $|X \cap Y|$.

Parameters: number of RPMT queries n.

Functionality: On input set Y from the server P_1 and input set $X = (x_1, \ldots, x_n) \subseteq \{0,1\}^\ell$ from the client P_2:

- Give output a vector $\vec{e} = (e_1, \ldots, e_n) \in \{0,1\}^n$ to P_1, where $e_i = 1$ if $x_i \in Y$ and $e_i = 0$ otherwise. *Also give $|X \cap Y|$ to P_1.

Figure 3: Ideal functionality $\mathcal{F}_{\mathsf{mqRPMT}}$ for multi-query RPMT

4 Commutative Weak Pseudorandom Functions

4.1 Definition of Commutative Weak PRF

We first recall the notion of standard pseudorandom functions (PRFs) [GGM86].

Definition 4.1 (PRF)**.** A family of PRFs consists of three polynomial-time algorithms as follows:

- $\mathsf{Setup}(1^\lambda)$: on input a security parameter λ, outputs public parameters pp, which specifies a family of keyed functions $F : K \times D \to R$.
- $\mathsf{KeyGen}(pp)$: on input pp, outputs a secret key $k \xleftarrow{R} K$.
- $\mathsf{Eval}(k, x)$: on input $k \in K$ and $x \in D$, outputs $y \leftarrow F(k, x)$. For notation convenience, we will write $F(k, x)$ as $F_k(x)$ interchangeably.

The standard security requirement for PRFs is pseudorandomness.

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

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

where $\mathcal{O}_{\mathsf{ror}}(0, x) = F_k(x)$, $\mathcal{O}_{\mathsf{ror}}(1, x) = \mathsf{H}(x)$ (here H is chosen uniformly at random from all the functions from D to R[1]). Note that $\mathcal{A}$ can adaptively access the oracle $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ polynomial many times. We say that F is pseudorandom if for any PPT adversary its advantage function $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible in λ. 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 $\mathcal{O}_{\mathsf{ror}}(b, \cdot)$ are uniformly chosen from D by the challenger instead of adversarially chosen by $\mathcal{A}$. PRFs that satisfy weak pseudorandomness are referred to as *weak PRFs*.

Composable. For a family of keyed function F, F is 2-composable if $R \subseteq D$, namely, for any $k_1, k_2 \in K$, the function $F_{k_1}(F_{k_2}(\cdot))$ is well-defined. In this work, we are interested in a special case namely $R = D$.

Commutative. For a family of composable keyed function, we say it is commutative if:

$$\forall k_1, k_2 \in K, \forall x \in X : F_{k_1}(F_{k_2}(x)) = F_{k_2}(F_{k_1}(x))$$

It is easy to see that the standard pseudorandomness denies commutative property. Consider the following attack against the standard pseudorandomness of F_k as below: the adversary $\mathcal{A}$ picks $k' \xleftarrow{R} K$, $x \xleftarrow{R} D$, and then queries the real-or-random oracle at point $F_{k'}(x)$ and point x respectively, receiving

[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.

back responses y' and y. $\mathcal{A}$ then outputs '1' iff $F_{k'}(y) = y'$. Clearly, $\mathcal{A}$ breaks the pseudorandomness with advantage $1/2 - \mathsf{negl}(\lambda)$. Provided commutative property exists, the best security we can expect is weak pseudorandomness. Looking ahead, weak pseudorandomness and commutative property may co-exist based on some well-studied assumptions.

Definition 4.2 (Commutative Weak PRF)**.** Let F be a family of keyed functions $K \times D \to D$. F is call commutative weak PRF if it satisfies weak pseudorandomness and commutative property simultaneously.

Further generalization. Instead of sticking to one family of keyed functions, commutative property can be defined over two families of keyed functions. Let F be a family of weak PRF $K \times D \to D$, G be a family of weak PRF $S \times D \to D$. If the following equation holds,

$$\forall k \in K, s \in S, \forall x \in X : F_k(G_s(x)) = G_s(F_k(x))$$

we say (F, G) is a tuple of commutative weak PRF.

We note that our notion if commutative weak PRF is similar to but strictly weaker than a previous notion called commutative encryption [AES03]. The difference is that cwPRF neither require F_k be a permutation nor F_k^{-1} be efficiently computable.

4.2 Instantiations of Commutative Weak PRF

We present two instantiations of commutative weak PRF. The first is built from the well-studied DDH assumption. The second is built from weak pseudorandom effective group actions (EGA), which in turn can be realized from the CSI-DDH assumption [dKGV20].

Construction from DDH. We build a concrete commutative weak PRF from the DDH assumption.

- $\mathsf{Setup}(1^\lambda)$: runs $\mathsf{GroupGen}(1^\lambda) \to (\mathbb{G}, g, p)$, outputs $pp = (\mathbb{G}, g, p)$.
- $\mathsf{KeyGen}(pp)$: outputs $k \xleftarrow{\text{R}} \mathbb{Z}_p$. Each k defines a function from $\mathbb{G}$ to $\mathbb{G}$, which takes $x \in \mathbb{G}$ as input and outputs x^k.

It is straightforward to verify that F is commutative. The following lemma establishes its pseudorandomness based on the DDH assumption.

Lemma 4.1. *The above construction is weak pseudorandom assuming the hardness of the DDH problem.*

Proof. Towards a tight security reduction, we utilize the random self-reducibility of the DDH problem [NR95]. Let $\mathcal{B}$ be an algorithm against the DDH assumption. Given a DDH challenge instance (g, g^a, g^b, g^c) over cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p, $\mathcal{B}$ interacts with an adversary $\mathcal{A}$ in the weak pseudorandom experiment, with the aim to determine if $c = ab \bmod p$ or not.

Setup: $\mathcal{B}$ sends $pp = (\mathbb{G}, g, p)$ to $\mathcal{A}$. $\mathcal{B}$ implicitly set a as the key of PRF.

Real-or-random query: Upon receiving the i-th query to oracle $\mathcal{O}_{\mathsf{ror}}$, $\mathcal{B}$ picks $d_i, e_i \xleftarrow{\text{R}} \mathbb{Z}_p$, sets the i-th random input $x_i := (g^b)^{d_i} \cdot g^{e_i}$, computes $y_i = g^{cd_i + ae_i}$, then sends (x_i, y_i) to $\mathcal{A}$.

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

Clearly, if $c = ab \bmod q$ then $\mathcal{B}$ simulates the real mode perfectly, else $\mathcal{B}$ simulates the random mode perfectly. Thereby, $\mathcal{B}$ breaks the DDH assumption with the same advantage as $\mathcal{A}$ breaks the pseduorandomness of F. □

Construction from EGA. We build cwPRF from weak pseudorandom EGA [AFMP20] as below.

- $\mathsf{Setup}(1^\lambda)$: generate pp that describes an effective group action $(\mathbb{G}, X, \star)$.
- $\mathsf{KeyGen}(pp)$: picks a random group element g from $\mathbb{G}$ as key. Each g defines a function from X to X, which takes $x \in X$ as input and outputs $g \star x$.

It is straightforward to verify that when $\mathbb{G}$ is an abelien group, the above construction constitutes cwPRF.

5 Multi-query RPMT from Commutative Weak PRF

In Figure 4, we show how to build mqRPMT from cwPRF $F : K \times D \to D$ and cryptographic hash function $\mathsf{H} : \{0,1\}^\ell \to D$.

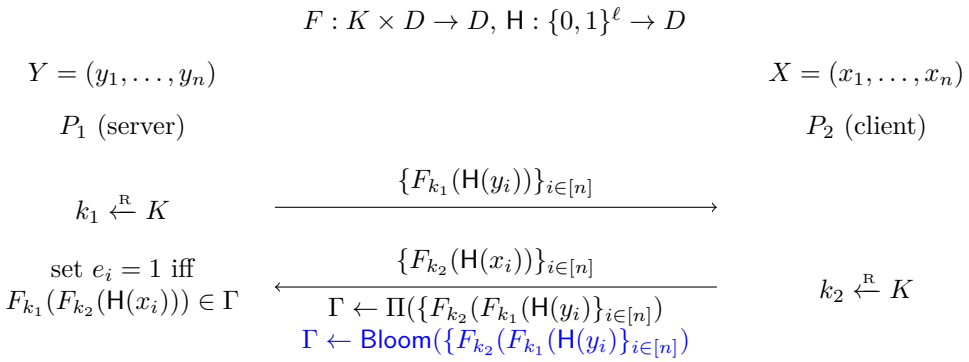
Parameters:

- Common input: $F : K \times D \to D$, hash function $\mathsf{H} : \{0,1\}^\ell \to D$.
- Input of server P_1: $Y = \{y_1, \ldots, y_n\} \subseteq \{0,1\}^\ell$.
- Input of client P_2: $X = \{x_1, \ldots, x_n\} \subseteq \{0,1\}^\ell$.

Protocol:

1. P_1 picks $k_1 \xleftarrow{R} K$, then sends $\{F_{k_1}(\mathsf{H}(y_1)), \ldots, F_{k_1}(\mathsf{H}(y_n))\}$ to P_2.
2. P_2 picks $k_2 \xleftarrow{R} K$, computes and sends $\{F_{k_2}(\mathsf{H}(x_1)), \ldots, F_{k_2}(\mathsf{H}(x_n))\}$ to P_1; then computes $\{F_{k_2}(F_{k_1}(\mathsf{H}(y_1)), \ldots, F_{k_2}(F_{k_1}(\mathsf{H}(y_n))\}$, sends its permutation or Bloom filter Γ to P_1;
3. P_1 computes $\{F_{k_1}(F_{k_2}(\mathsf{H}(x_1)), \ldots, F_{k_2}(F_{k_1}(\mathsf{H}(x_n))\}$, then sets $e_i = 1$ iff $F_{k_1}(F_{k_2}(\mathsf{H}(x_i))) \in \Gamma$.

Figure 4: Multi-query RPMT from commutative weak PRF



Correctness. The above protocol is correct except the case E that $F_{k_1}(F_{k_2}(\mathsf{H}(x))) = F_{k_1}(F_{k_2}(\mathsf{H}(y)))$ for some $x \neq y$ occurs. We further divide E to E_0 and E_1. E_0 denotes the case that $\mathsf{H}(x) = \mathsf{H}(y)$. E_1 denotes the case that $\mathsf{H}(x) \neq \mathsf{H}(y)$ but $F_{k_1}(F_{k_2}(\mathsf{H}(x))) = F_{k_1}(F_{k_2}(\mathsf{H}(y)))$, which can further be divided into sub-cases E_{10} — $F_{k_2}(\mathsf{H}(x)) = F_{k_2}(\mathsf{H}(y))$ and E_{11} — $F_{k_2}(\mathsf{H}(x)) \neq F_{k_2}(\mathsf{H}(y))$ but $F_{k_1}(F_{k_2}(\mathsf{H}(x))) = F_{k_1}(F_{k_2}(\mathsf{H}(y)))$. By the collision resistance of H, we have $\Pr[E_0] = 2^{-\sigma}$. By the weak pseudorandomness of F, we have $\Pr[E_{10}] = \Pr[E_{11}] = 2^{-\ell}$. Therefore, we have $\Pr[E] \leq \Pr[E_0] + \Pr[E_{10}] + \Pr[E_{11}] = 2^{-\sigma} + 2^{-\ell+1}$.

Theorem 5.1. *The above multi-query RPMT protocol is secure in the semi-honest model assuming* H *is a random oracle and* F *is a family of cwPRF.*

Proof. 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 produced transcript from the real execution. Let $|X \cap Y| = m$.

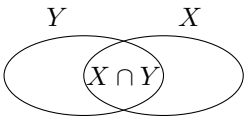
Corrupt client: Sim_{P_2} simulates the view of corrupt client P_2, which consists of P_2's randomness, input, output and received messages.

We argue the output of Sim_{P_2} is indistinguishable from the real execution. For this, we formally show the simulation by proceeding the sequence of hybrid transcripts, where T_0 is the real view of P_2, and T_1 is the output of Sim_{P_2}.

Hybrid$_0$: Sim_{P_2} chooses the randomness for P_1 (i.e., picks $k_1 \xleftarrow{R} K$), and simulates with the knowledge of Y.

- RO queries: for random oracle query $\langle z_i \rangle$, picks $\alpha_i \xleftarrow{R} D$ and set $\mathsf{H}(z_i) := \alpha_i$.
- Let $\beta_i = \mathsf{H}(y_i)$ for each $y_i \in Y$. Sim_{P_2} outputs $(F_{k_1}(\beta_1), \ldots, F_{k_1}(\beta_n))$.

Clearly, Sim_{P_2}'s simulation is identical to the real view of P_2.



$\mathsf{H}(z_i) := \alpha_i \xleftarrow{R} D$

Hybrid$_1$: Sim_{P_2} does not choose the randomness for P_1, and simulates without the knowledge of Y. It simulates the RO queries the same way as in Hybrid$_0$, and only changes the simulation of P_1's message.

- Sim_{P_2} outputs $(s_1, \ldots, s_n)$ where $s_i \xleftarrow{R} D$.

We argue that the view in Hybrid$_0$ and Hybrid$_1$ are computationally indistinguishable. More precisely, a PPT adversary $\mathcal{A}$ (with knowledge of X and Y) against cwPRF (with secret key k) is given n tuples (β_i, s_i) where $\beta_i \xleftarrow{R} D$, and is asked to distinguish if $s_i = F_k(\beta_i)$ or s_i are random values. $\mathcal{A}$ implicitly sets P_1's randomness $k_1 := k$.

- RO queries: for random oracle query $\langle z_i \rangle$ where $z_i \notin Y$, picks $\alpha_i \xleftarrow{R} D$ and set $\mathsf{H}(z_i) := \alpha_i$; for random oracle query $\langle z_i \rangle$ where $z_i \in Y$, sets $\mathsf{H}(z_i) := \beta_i$.
- $\mathcal{A}$ outputs $(s_1, \ldots, s_n)$.

If $s_i = F_k(\beta_i)$ for $i \in [n]$, then $\mathcal{A}$'s simulation is identical to Hybrid$_0$. If s_i are random values, then $\mathcal{A}$'s simulation is identical to Hybrid$_1$.

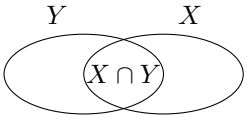
Corrupt server: Sim_{P_1} simulates the view of corrupt server P_1, which consists of P_1's randomness, input, output and received messages.

We argue the output of Sim_{P_1} is indistinguishable from the real execution. For this, we formally show the simulation by proceeding the sequence of hybrid transcripts, where T_0 is the real view of P_1, and T_1 is the output of Sim_{P_1}.

Hybrid$_0$: Sim_{P_1} chooses the randomness for P_2 (i.e., picks $k_2 \xleftarrow{R} K$), and simulates with the knowledge of X.

- RO queries: for random oracle query $\langle z_i \rangle$, picks $\alpha_i \xleftarrow{R} D$ and sets $\mathsf{H}(z_i) := \alpha_i$.
- Sim_{P_1} outputs $\{F_{k_2}(\mathsf{H}(x_i))\}_{x_i \in X}$ and $\Gamma \leftarrow \Pi(\{F_{k_2}(F_{k_1}(\mathsf{H}(y_i))\}_{y_i \in Y})$.

Clearly, Sim_{P_1}'s simulation is identical to the real view of P_1.



$\mathsf{H}(z_i) := \alpha_i \xleftarrow{R} D$

Hybrid$_1$: Sim_{P_1} does not choose randomness for P_1, and simulates without the knowledge of X, but only with the knowledge of $\vec{e} = (e_1, \ldots, e_n)$ where $e_i = 1$ iff $x_i \in Y$. Let the Hamming weight of $\vec{e}$ be m. It simulates the RO queries the same way as in Hybrid$_0$, and changes it simulation of P_2's message.

- Sim_{P_1} picks $v_i \xleftarrow{R} D$ for $i \in [n]$ (associated with $F_{k_2}(\mathsf{H}(x_i))$ where $x_i \in X$), outputs $\{v_i\}_{i \in [n]}$; picks $t_j \xleftarrow{R} D$ for $j \in [n-m]$ (associated with $F_{k_2}(\mathsf{H}(y_j))$ where $y_j \in Y - X \cap Y$), outputs the permutation of $(\{F_{k_1}(v_i)\}_{e_i=1}, \{F_{k_1}(t_j)\}_{j \in [n-m]})$.

We argue that the view in Hybrid_0 and Hybrid_1 are computationally indistinguishable. More precisely, a PPT adversary $\mathcal{A}$ (with knowledge of X and Y) against cwPRF are given $n+m$ tuples (α_t, s_t) where $\alpha_t \xleftarrow{R} D$, and is asked to determine if $s_t = F_k(\alpha_t)$ or random values. $\mathcal{A}$ implicitly sets P_2's randomness $k_2 := k$, picks $k_1 \xleftarrow{R} K$.

- RO queries: for $z_i \notin X \cup Y$, picks $\beta_i \xleftarrow{R} D$ and returns $\mathsf{H}(z_i) := \beta_i$; for $z_i \in X \cup Y$, returns $\mathsf{H}(z_i) := \alpha_i$.
- $\mathcal{A}$ picks out s_t such that $\mathsf{H}(z_t) := \alpha_t$ for $z_t = x_i \in X$ to form $\{v_i\}_{i\in[n]}$, picks out s_t such that $\mathsf{H}(z_t) := \alpha_t$ for $z_t = x_i \in X \backslash Y$ to form $\{w_j\}_{j\in[n-m]}$. Finally, $\mathcal{A}$ outputs $\{v_i\}_{i\in[n]}$ and the permutation or Bloom filter of $(\{F_{k_1}(v_i)\}_{x_i\in X\cap Y}, \{F_{k_1}(w_j)\}_{j\in[n-m]}$.

If s_i are function values, then the simulation is identical to Hybrid_0, else it is identical to Hybrid_1.

This proves the theorem. □

Remark 5.1. In the above construction of mqRPMT, P_1's message is of the form $F_{k_1}(y_i)$, part of P_2's message is of the form $F_{k_2}(x_i)$. The common theme is to apply H and F_k in a cascade way, yielding a composite function $F_k \circ \mathsf{H} : \{0,1\}^* \to D$. By leveraging the programmability of H, the composite function $F_k \circ \mathsf{H}$ is family of PRF in the random oracle model. In essence, random oracle amplifies weak pseudorandomness to standard pseudorandomness.

6 mqRPMT from mqPMT

6.1 Private Membership Test

Private membership test (PMT) protocol [PSZ14] is a two-party protocol in which the client with input x learns whether or not its item is in the input set Y of the server. PMT can be viewed as a special case of private keyword search protocol [FIPR05] by setting the payload as any indication string. We consider three-message PMT as below, which we refer to Sigma-PMT hereafter. Sigma-PMT proceeds via following pattern.

1. Server P_1 sends the first round message a to sender P_2, which is best interpreted as an encoding of Y.
2. Sender P_2 sends query q w.r.t. to his item x.
3. Server P_1 responds with t.

After receiving t, client P_2 can decide if $x \in Y$ by running $\mathsf{Test}(a, x, q, t)$. The basic notion of Sigma PMT allows the client (P_2) to test for a single item. While this procedure can be repeated several times, one may seed more efficient protocol allowing the client to test n items at a reduced cost. We consider such an extension which we refer to as amortized Sigma-PMT, which satisfies the following two properties:

- **Reusable:** the first round message is performed by the server (P_1) once and for all
- **Non-adaptive:** each test query q_i is independent of previous messages

Clearly, amortized Sigma-PMT admits parallelization, hence the round complexity is unchanged even when handling multiple items. Amortized Sigma-PMT may enjoy an additional property:

- **Stateless:** the test algorithm can work in a memoryless way, namely, for any xi and associated (q_i, t_i), we have: $\mathsf{Test}(a, x_i, q_i, t_i) = \mathsf{Test}(a, t_i)$.

We depict amortized Sigma-PMT that supports stateless testing in Figure 5.

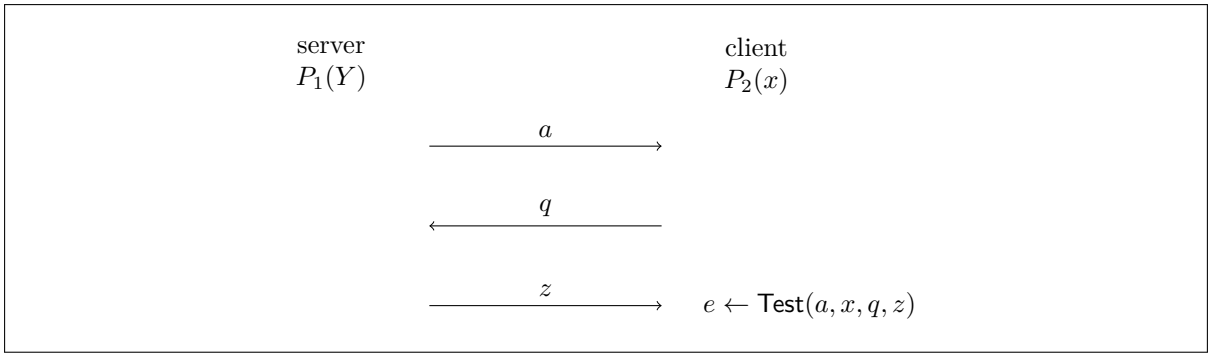


Figure 5: Sigma PMT

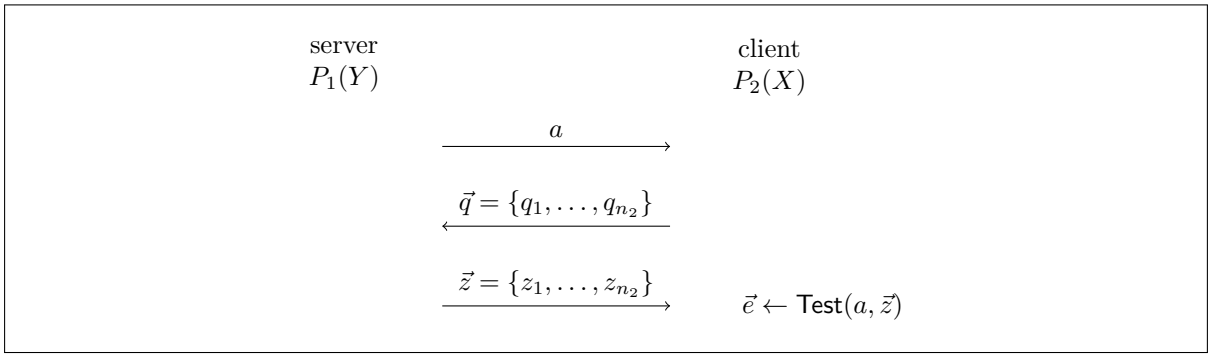


Figure 6: mqPMT from Amortized Sigma PMT Supporting Stateless Test

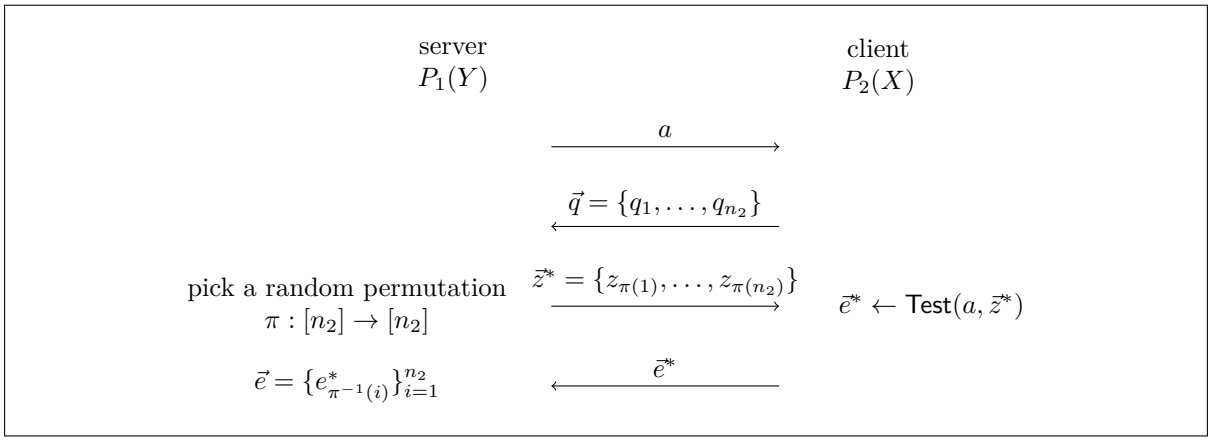


Figure 7: mqRPMT from mqPMT

6.2 Connection to mqRPMT

Theorem 6.1. *The above mqRPMT* protocol is secure in the semi-honest model assuming the semi-honest security of the starting mqPMT protocol.*

Proof. We exhibit simulators Sim_{P_1} and Sim_{P_2} for simulating corrupt server P_1 and corrupt client P_2 respectively. Let $|X \cap Y| = m$.

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 argue that the output of Sim_{P_2} is indistinguishable from the real execution. For this, we formally show the simulation by proceeding the sequence of hybrid transcripts, where T_0 is the real view of P_2, and T_1 is the output of Sim_{P_2}.

Hybrid_0: Sim_{P_2} chooses the randomness r for P_1, and simulates with the knowledge of Y. Clearly, Sim_{P_2}'s simulation is identical to the real view of P_2.

Hybrid_1: Sim_{P_2} does not choose the randomness for P_1, and simulates without the knowledge of Y. Instead, it invokes the mqPMT's simulator for P_2 on his private input X and output $\vec{e}^*$ to generate $(a, \vec{z}^*)$, outputs $(a, \vec{z}^*)$.

Hybrid_2: Sim_{P_2} first generate a random indication vector $\vec{t}^*$ with Hamming weight $m = |X \cap Y|$, then invokes the mqPMT's simulator for P_2 on his private input X and output $\vec{t}^*$ to generate $(a, \vec{z}^*)$, outputs $(a, \vec{z}^*)$.

Clearly, the view in Hybrid_0 and Hybrid_1 are computationally indistinguishable based on the semi-honest security of mqPMT on P_2's side. Besides, since both e^* and t^* follow the same distribution, thus the view in Hybrid_1 and Hybrid_2 are identical. In conclusion, the view in Hybrid_0 and Hybrid_2 are computationally indistinguishable.

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 argue that the output of Sim_{P_1} is indistinguishable from the real execution. For this, we formally show the simulation by proceeding the sequence of hybrid transcripts, where T_0 is the real view of P_1, and T_1 is the output of Sim_{P_1}.

Hybrid_0: Sim_{P_1} chooses the randomness r for P_2, and simulates with the knowledge of X. Clearly, Sim_{P_1}'s simulation is identical to the real view of P_1.

Hybrid_1: Sim_{P_1} does not choose the randomness for P_2, and simulates without the knowledge of X. Instead, it first invokes the mqPMT's simulator for P_2 on his input Y to generate $\vec{q}$, then picks a random permutation π, computes $\vec{e}^* = \pi(\vec{e})$, outputs $(\vec{q}, \pi(\vec{z}))$.

Clearly, the view in Hybrid_0 and Hybrid_1 are computationally indistinguishable based on the semi-honest security of mqPMT on P_1's side.

This proves the theorem. □

7 Applications of Multi-Query RPMT

We show how to build a PSO framework central around mqRPMT in Figure 8.

Theorem 7.1. *The resulting PSU protocol is semi-honest secure by assuming the semi-honest security of mqRPMT and OT.*

How to formally prove this theorem? Shall we interpret client's private input in mqRPMT as a set X or as a vector?

8 Implementation

Table 1: The computation and communication complexity of PSU

	Protocol	Setting (LAN)	Set size n		
			2^{12}	2^{16}	2^{20}
Time (s)	Ours	1 Mbps			
		10 Mbps			
		100 Mbps			
		68 Gbps	0.45	5.3	86.9
Comm. (MB)	Ours		0.47	7.4	117.6

Here

Parameters:

- Input of receiver P_1: $Y = \{y_1, \ldots, y_{n_1}\} \subseteq \{0,1\}^\ell$.
- Input of sender P_2: $X = \{x_1, \ldots, x_{n_2}\} \subseteq \{0,1\}^\ell$ and $V = \{v_1, \ldots, v_{n_2}\}$.

Protocol:

1. P_1 (playing the role of server) with Y and P_2 (playing the role of receiver) with X invoke $\mathcal{F}_{\mathsf{mqRPMT}}$. P_1 obtains an indication bit vector $\vec{e} = (e_1, \ldots, e_{n_2})$. P_2 obtains nothing.
   - **cardinality:** P_1 learns the cardinality by calculating the Hamming weight of $\vec{e}$.
2. P_1 and P_2 invoke n_2 instances of OT via $\mathcal{F}_{\mathsf{OT}}$. P_1 uses $\vec{e}$ as the choice bits.
   - **intersection:** P_2 uses $(\perp, x_i)$ as input to the ith OT. P_1 learns $\{x_i \mid e_i = 1\}_{i\in[n_2]} = X \cap Y$.
   - **union:** P_2 uses $(x_i, \perp)$ as input to the ith OT. P_1 learns $\{x_i \mid e_i = 0\}_{i\in[n_2]} = X/Y$, and outputs $\{X/Y\} \cup Y = X \cup Y$.
   - **intersection sum with cardinality:** P_2 randomly generate r_i subject to the constraint $\sum_{i=1}^{n_2} r_i = 0$, then uses $(r_i, r_i + v_i)$ as input to the ith OT. P_1 learns $\{\sum_{i=1}^{n_2} v_i \mid e_i = 1\}_{i\in[n_2]} = X \cap Y$.

Figure 8: PSO from mqRPMT

and collect the benchmarks on a MacBook Pro with an 2.6GHz Intel CPU and 16GB of RAM

References

[AES03] Rakesh Agrawal, Alexandre V. Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97. ACM, 2003.

[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.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 673–701. Springer, 2015.

[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.

[DC17] Alex Davidson and Carlos Cid. An efficient toolkit for computing private set operations. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017*, volume 10343 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2017.

[dKGV20] Bor de Kock, Kristian Gjøsteen, and Mattia Veroni. Practical isogeny-based key-exchange with optimal tightness. In *Selected Areas in Cryptography - SAC 2020*, volume 12804 of *LNCS*, pages 451–479. Springer, 2020.

[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.

[Fri07] Keith B. Frikken. Privacy-preserving set union. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007*, volume 4521 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2007.

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

[GMR+21] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In *Public-Key Cryptography - PKC 2021*, volume 12711 of *Lecture Notes in Computer Science*, pages 591–617. Springer, 2021.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *Advances in Cryptology - CRYPTO 2021*, volume 12826 of *Lecture Notes in Computer Science*, pages 395–425. Springer, 2021.

[HFH99] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)*, pages 78–86. ACM, 1999.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *Public Key Cryptography - PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 312–331. Springer, 2010.

[IKN+20] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020*, pages 370–389. IEEE, 2020.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. Improved OT extension for transferring short secrets. In *Advances in Cryptology - CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2013.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016*, pages 818–829. ACM, 2016.

[KRTW19] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. Scalable private set union from symmetric-key techniques. In *Advances in Cryptology - ASIACRYPT 2019*, volume 11922 of *Lecture Notes in Computer Science*, pages 636–666. Springer, 2019.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2005.

[Mea86] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy*, pages 134–137. IEEE Computer Society, 1986.

[MPR+20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2020.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of psuedo-random functions. In *36th Annual Symposium on Foundations of Computer Science, FOCS 1995*, pages 170–181. IEEE Computer Society, 1995.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse OT extension. In *Advances in Cryptology - CRYPTO 2019*, volume 11694 of *Lecture Notes in Computer Science*, pages 401–431. Springer, 2019.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *Proceedings of the 23rd USENIX Security Symposium, 2014*, pages 797–812. USENIX Association, 2014.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 21(2):7:1–7:35, 2018.

[Rab] Michael O. Rabin. How to exchange secrets with oblivious transfer. http://eprint.iacr.org/2005/187.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12697 of *Lecture Notes in Computer Science*, pages 901–930. Springer, 2021.

[Sha80] Adi Shamir. On the power of commutativity in cryptography. In *ICALP 1980*, volume 85 of *Lecture Notes in Computer Science*, pages 582–595. Springer, 1980.

A Missing Definitions

A.1 Weak Pseudorandom EGA

We begin by recalling the definition of a group action.

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

Definition A.2 (Effective Group Actions)**.** A group action $(\mathbb{G}, X, \star)$ is *effective* a set X 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 represents the same group element in $\mathbb{G}$.
   (c) Sampling, i.e., to sample an element g from a uniform (or statistically close to) distritution 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 arbitrary 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 A.3 (Weak Pseudorandom EGA)**.** There is no PPT adversary that can distinguish tuples of the form $(x_i, g \star x_i)$ from (x_i, u_i) where $g \leftarrow \mathbb{G}$ and each $x_i, u_i \leftarrow X$ are sampled uniformly at random.