Private Set Operations from Multi-query Reverse Private Membership Test

Yu Chen[1,2,3(✉)], Min Zhang[1,2,3], Cong Zhang[4], Minglang Dong[1,2,3], and Weiran Liu[5]

[1] School of Cyber Science and Technology, Shandong University, Qingdao 266237, China
yuchen@sdu.edu.cn, {zm_min,minglang_dong}@mail.sdu.edu.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao 266237, China
[4] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
zhangcong@mail.tsinghua.edu.cn
[5] Alibaba Group, Hangzhou, China
weiran.lwr@alibaba-inc.com

Abstract. Private set operations allow two parties to perform secure computation on their private sets, including intersection, union and functions of intersection/union. In this paper, we put forth a framework to perform private set operations. The technical core of our framework is the multi-query reverse private membership test (mqRPMT) protocol (Zhang et al., USENIX Security 2023). We present two constructions of mqRPMT from newly introduced cryptographic notions, one is based on commutative weak pseudorandom function (cwPRF), and the other is based on permuted oblivious pseudorandom function (pOPRF). Both cwPRF and pOPRF can be realized from the decisional Diffie-Hellman (DDH)-like assumptions in the random oracle model.

We demonstrate the practicality of our framework with implementations. By plugging our cwPRF-based mqRPMT into the framework, we obtain various PSO protocols that are superior or competitive to the state-of-the-art protocols. For intersection functionality, our protocol is faster than the most efficient one for small sets. For cardinality functionality, our protocol achieves a $2.4-10.5\times$ speedup and a $10.9-14.8\times$ reduction in communication cost. For cardinality-with-sum functionality, our protocol achieves a $28.5-76.3\times$ speedup and $7.4\times$ reduction in communication cost. For union functionality, our protocol is the first one that achieves strictly linear complexity, and requires the lowest concrete computation and communication costs in all settings, achieving a $2.7-17\times$ speedup and about $2\times$ reduction in communication cost. Furthermore, our improvement on PSU also translates to related functionality, yielding the most efficient private-ID protocol to date.

Keywords: PSO · PSU · multi-query RPMT · commutative weak PRF · permuted OPRF

Q. Tang and V. Teague (Eds.): PKC 2024, LNCS 14603, pp. 387–416, 2024.
https://doi.org/10.1007/978-3-031-57725-3_13

1 Introduction

Consider several parties, each with a private set of items, want to perform computation on their private sets without revealing any other information to each other. Private set operation (PSO) refers to such family of interactive cryptographic protocols that fulfill this task, which take private sets as inputs and compute the desired function, delivering the result to the participants. In this work, we focus on two-party PSO protocols with semi-honest security. In what follows, we briefly review related works in terms of typical functionalities.

Private Set Intersection (PSI). PSI allows two parties, the sender and the receiver, to compute the intersection of their private sets X and Y, such that the receiver only learns $X \cap Y$ and the sender learns nothing. PSI has found numerous applications including privacy-preserving location sharing [NTL+11], private contact discovery [DRRT18], DNA testing and pattern matching [TKC07]. Due to its importance and wide applications, in the past two decades PSI has been extensively studied in a long sequence of works and has become truly practical with extremely fast implementations. The most efficient PSI protocols [KKRT16, PRTY19, CM20, GPR+21, RS21] mainly rely on symmetric-key operations, except $O(\kappa)$ public-key operations (where κ is a computational security parameter) in base OT used in the OT extension protocol. We refer to [PSZ18] for a good survey of different PSI paradigms.

Private Computing on Set Intersection (PCSI). Certain real-world application scenarios only require partial/aggregated information about the intersection. In this setting fine-grained private computation on set intersection (PCSI) is needed, such as PSI-card for intersection cardinality [HFH99, AES03, CGT12], PSI-card-sum for intersection cardinality and sum [IKN+20, GMR+21]. For general-purpose PCSI (also known as circuit-PSI) [HEK12, PSTY19], the parties learn secret shares of elements in the set intersection, which can be further fed into generic 2PC to compute $g(X \cap Y)$ for arbitrary function g.

Private Set Union (PSU). PSU allows two parties, the sender and the receiver, to compute the union of their private sets X and Y, such that the receiver only learns $X \cup Y$ and the sender learns nothing. Like PSI, PSU also has many applications in practice, such as cyber risk assessment and management [LV04], IP blacklist and vulnerability data aggregation [HLS+16], private DB supporting full join [KRTW19] and private ID [GMR+21]. Existing PSU protocols can be broadly divided into two categories based on the underlying cryptographic techniques used. The first category mainly relies on public-key techniques [KS05, Fri07, HN10, DC17], while the second category mainly relies on symmetric-key techniques [KRTW19, GMR+21, JSZ+22]. We refer to [ZCL+23] for a comprehensive survey of existing PSU protocols.

Among PSO protocols, PSI has been extensively studied. Numerous PSI protocols achieve linear complexity, and the current state-of-the-art PSI [RR22] is almost as efficient as the naive insecure hash-based protocol. In contrast, the study of PCSI and PSU is less satisfactory. In the case of PCSI, while a few protocols [PSTY19, IKN+20] achieve linear complexity, their practical performance

is poor. As shown in [GMR+21], even in the simplest case of semi-honest PCSI - like PSI-card - is concretely about 20× slower and requires over 30× more communication than PSI. In the case of PSU, no protocol with linear complexity in either balanced or unbalanced setting is known for a long time being. It is until very recently, Zhang et al. [ZCL+23] make a breakthrough by proposing the first PSU with linear complexity. However, their work does not close this issue. Their concrete PSU protocols have a large constant term in computation complexity, incurring a significant efficiency gap compared with PSI: roughly 25× slower and requires at least 3× more communication than PSI.

It is somewhat surprising that different PSO protocols have significantly different efficiency. One may wonder: what is the reason for this discrepancy? Observe that PSI can be essentially viewed as multi-query private membership test (mqPMT), which has efficient realizations in both balanced and unbalanced settings. However, mqPMT generally does not imply PCSI or PSU. The reason is that mqPMT reveals information about the intersection, which should be hidden from the receiver in PCSI and PSU.

1.1 Motivation

Our motivation of this work is threefold. First, the above discussion indicates that the most efficient PSI protocols may not be easily adapted to PCSI and PSU protocols. Consequently, constructions of different PSO protocols differ vastly in the types of techniques they employ, requiring significant engineering effort and making it difficult to deploy PSO systematically. This calls for a modular approach that allows for an easier navigation in the huge design space. We are thus motivated to seek for a common principal that enables all private set operations through a unified framework. Second, given the large efficiency gap between PSI and other related protocols, we are also motivated to give efficient instantiations of the framework to narrow the gap. Last but not least, it is worth noting that the seminal PSI protocol, DH-PSI [Mea86] (related ideas were appeared in [Sha80,HFH99]), which was derived from the Diffie-Hellman key-exchange protocol, based on the decisional Diffie-Hellman (DDH) assumption, is still the most easily understood and implemented one among many PSI protocols for over four decades. Somewhat surprisingly, no PSU counterpart of DH-PSI has been discovered yet. It is curious to know whether the DDH assumption is also useful in the PSU setting. In sum, our work focus on the following questions:

Is there a central building block that enables a unified framework for all private set operations? If so, can we give efficient instantiations with optimal asymptotic complexity and good concrete efficiency? And finally, can the DDH assumption be used to construct efficient PSU protocols?

1.2 Our Contribution

We provide affirmative answers to the aforementioned questions. Our main results are summarized as below.

A Framework of PSO. We identify that multi-query reverse private membership test (mqRPMT) [ZCL+23] is actually a "Swiss Army Knife" for various private set operations. mqRPMT already implies PSI-card by itself; by coupling with OT, mqRPMT implies PSI and PSU; by additionally coupling with simple secret sharing, mqRPMT implies PSI-card-sum and PSI-card-secret-sharing, where the latter further admits general-purpose PCSI with cardinality. Therefore, mqRPMT enables a unified PSO framework, which can perform a variety of private set operations in a flexible manner.

Efficient Construction of mqRPMT. We present two generic constructions of mqRPMT. The first is based on a newly formalized cryptographic primitive called commutative weak PRF (cwPRF), while the second is based on a newly introduced secure protocol called permuted oblivious PRF (pOPRF). Both of them can be realized from DDH-like assumptions in the random oracle model, yielding incredibly simple mqRPMT constructions with linear communication and computation complexity. Note that the complexity of our PSO framework is dominated by the underlying mqRPMT. Therefore, all resulting PSO protocols inherit optimal linear complexity. Notably, the obtained PSU protocol is arguably the most simple and efficient one among existing PSU protocols.

Evaluations. We give efficient implementation of our generic framework from the cwPRF-based mqRPMT protocol. The experimental results demonstrate that all PSO protocols derived from our generic framework are superior or competitive to the state-of-the-art corresponding protocols.

1.3 Technical Overview

PSO from mqRPMT. As discussed above, mqPMT (equivalent to PSI) is generally not applicable for computing PCSI and PSU. We examine the reverse direction, i.e., whether the core protocol underlying PSU can be used for computing PSI and PCSI. We identify that the recently emerged mqRPMT protocol [ZCL+23], which is a generalization of RPMT formalized in [KRTW19], is actually a central protocol underlying all the existing PSU protocols. Roughly speaking, mqRPMT is a two-party protocol between a server holding a set Y and a client holding a vector $X = (x_1, \ldots, x_n)$. After the execution, 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 knowing x_i, while the client learns nothing. Superficially, mqRPMT is similar to mqPMT, except that it is the server but not the client learns the test results. This subtle difference turns out to be crucial. To see this, note that in mqRPMT the intersection information (i.e. x_i and e_i) is shared between two parties, while in mqPMT the intersection information is entirely known by the client. In light of this difference, mqRPMT is not only particularly suitable for functionalities that have to keep intersection private, but also retains the necessary information to compute the intersection.

More precisely, we can build a family of PSO protocols from mqRPMT in a modular fashion. PSI-card protocol is immediate since the cardinality of intersection is exactly the Hamming weight of indication vector. PSI (resp. PSU)

protocol can be created by having the receiver (playing the role of server) and the sender (playing the role of client) invoke a mqRPMT protocol in the first place, then carry out n one-sided OTs with $1-e_i$ (resp. e_i) and x_i. PSI-card-sum and PSI-card-secret-sharing protocols can be constructed by additionally coupling with OT and simple secret-sharing trick. We defer the construction details to Sect. 6.

Next, we give two generic constructions of mqRPMT. For clarity, we explicitly parameterize RPMT and PMT with two parameters n_1 and n_2, namely (n_1, n_2)-(R)PMT, where n_1 is the size of server's set Y, n_2 is the length of client's vector X, a.k.a. the number of membership test queries.

mqRPMT from cwPRF. Observe that private equality test (PEQT) protocol [PSZ14] not only can be viewed as an extreme case of mqPMT, but also can be viewed as an extreme case of mqRPMT. Under the parameterized notions, PEQT is essentially $(1, 1)$-PMT and $(1, 1)$-RPMT. We choose PEQT as the starting point of our first mqRPMT construction.

The basic idea of building $(1, 1)$-RPMT protocol amenable to extension is using *oblivious joint encoding*, by which an element can only be encoded to a codeword by two parties in a joint manner, while the process reveals nothing to the party without the element. To implement this idea, we formalize a new cryptographic primitive called commutative weak PRF (cwPRF). Let $F : K \times D \to R$ be a family of weak PRF, where $R \subseteq D$. F is commutative if $F_{k_1}(F_{k_2}(x)) = F_{k_2}(F_{k_1}(x))$ for any $k_1, k_2 \in K$ and any $x \in D$. 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, denoted by $\hat{F}$.

Now we are ready to describe the construction of $(1, 1)$-RPMT from cwPRF. The server P_1 holding y and the client P_2 holding x can conduct PEQT functionality via the following steps: (1) P_1 and P_2 generate cwPRF key k_1 and k_2 respectively, and map their items to domain D of F using a common cryptographic hash function H, which will be modeled as a random oracle; (2) P_1 computes and sends $F_{k_1}(\mathsf{H}(y))$ to P_2; (3) P_2 computes and sends $F_{k_2}(\mathsf{H}(x))$ and $F_{k_2}(F_{k_1}(\mathsf{H}(y)))$ to P_1; (4) 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 ensures the correctness. The weak pseudorandomness of F guarantees that P_2 learns nothing and P_1 learns nothing more than the test result. In the above construction, $F_{k_2}(F_{k_1}(\mathsf{H}(\cdot))) = F_{k_1}(F_{k_2}(\mathsf{H}(\cdot))) = \hat{F}_k(\mathsf{H}(\cdot))$ serves as a pseudorandom encoding function in the joint view, while $F_{k_1}(\mathsf{H}(\cdot))$ and $F_{k_2}(\mathsf{H}(\cdot))$ serve as a partial encoding function in the individual views of the server and client respectively.

We then extend the above $(1, 1)$-RPMT protocol to $(n_1, 1)$-RPMT. Note that naive repetition by sending back $F_{k_2}(F_{k_1}(\mathsf{H}(y_i)))$ for each $y_i \in Y$ in the same order of the server's first move message $F_{k_1}(\mathsf{H}(y_i))$ does not lead to a secure $(n_1, 1)$-RPMT. This is because $\{\hat{F}_k(\mathsf{H}(y_i))\}_{i\in[n_1]}$ constitutes an order-preserving pseudorandom encoding of $(y_1, \ldots, y_{n_1})$, and as a consequence, the server will learn the exact value of x if $x \in Y$. The idea to perform the membership test in an oblivious manner is making the pseudorandom encoding of $(y_1, \ldots, y_{n_1})$ independent of the order known by the server. A straightforward approach is to shuffle $\{\hat{F}_k(\mathsf{H}(y_i))\}$. This yields a $(n_1, 1)$-RPMT protocol from cwPRF, which

can be batched to a full-fledged (n_1, n_2)-RPMT protocol by reusing the encoding key k_2. A simple calculation shows that for a (n_1, n_2)-RPMT protocol, the computation cost is $(n_1 + n_2)$ mappings, $(2n_1 + n_2)$ evaluations of F, n_2 lookups and one shuffling, and the communication cost is $(2n_1 + n_2)$ elements in the range of F. The resulting mqRPMT protocol is optimal in the sense that both computation and communication complexity are linear to the set size. To further reduce the communication cost, we can insert $\{\hat{F}(\mathsf{H}(y_i))\}$ into an order-hiding data structure such as a Bloom filter [Blo70] instead of shuffling them.

In Sect. 4.2, we show that cwPRF can be realized from DDH-like assumptions. Combining this with the above results, DDH implies all PSO protocols. Remarkably, it strikes back with an incredibly simple PSU protocol, once again demonstrating that the DDH assumption is truly a golden mine in cryptography.

mqRPMT from permuted OPRF. We choose $(n, 1)$-RPMT as the starting point of our second mqRPMT construction. The idea is *oblivious permuted encoding*, in which only one party say P_2 is able to encode, and the other party say P_1 learns the codewords of its elements $(y_1, \ldots, y_n)$ in a permuted order, while both parties learn nothing more. A tempting approach to implement this idea is using multi-point OPRF that underlies many PSI protocols [PRTY19, CM20]. More precisely, having P_1 (acts as the OPRF's client) and P_2 (acts as the OPRF's server) engage in an OPRF protocol. Eventually, P_1 obtains PRF values of $(y_1, \ldots, y_n)$ as encodings, and P_2 obtains a PRF key k. However, OPRF does not readily enable oblivious permuted encoding, because the standard OPRF functionality always gives the PRF values with the same order of inputs. To remedy this issue, we introduce a new cryptographic protocol called permuted OPRF (pOPRF). pOPRF can be viewed as a generalization of OPRF. The difference is that the server additionally obtains a random permutation π over $[n]$ besides PRF key k, while the client obtains PRF values in a permuted order as per π. pOPRF immediately implies a $(n, 1)$-RPMT protocol: The server P_1 with $Y = (y_1, \ldots, y_n)$ (acts as the pOPRF's client) and the client P_2 with $X = \{x\}$ (acts as the pOPRF's server) first engage in a pOPRF protocol. As a result, P_1 obtains $\{F_k(y_{\pi(i)})\}_{i\in[n]}$, and P_2 obtains a PRF key k and a permutation π over $[n]$. P_2 then computes and sends $F_k(x)$ to the server as an RPMT query for x. Finally, P_1 learns if $x \in Y$ by testing whether $F_k(x) \in \{F_k(y_{\pi(i)})\}_{i\in[n]}$, but learns nothing more since its received PRF values are of permuted order. At a high level, $F_k(\cdot)$ serves as an encoding function in mqRPMT-client's view, while $F_k(\pi(\cdot))$ serves as a permuted pseudorandom encoding function in mqRPMT-server's view. Extending the above $(n, 1)$-RPMT to full-fledged (n_1, n_2)-RPMT is straightforward by having the client reuse k and send $\{F_k(x_i)\}_{i\in[n_2]}$ as RPMT queries.

Given the above, it remains to investigate how to build pOPRF. Recall that a common approach to build OPRF is "mask-then-unmask", we choose OPRF along this line as the starting point. The rough idea is exploiting the input homomorphism to mask inputs[1], then unmask the outputs. If the mask procedure is

[1] Standard pseudorandomness denies input homomorphism. Rigorously speaking, we utilize the homomorphism over intermediate input.

different per input, then different unmask procedure must be carried out accordingly. For this reason, OPRF protocols of this case cannot be easily adapted to pOPRF, since the receiver is unable to perform the unmask procedure over permuted masked outputs correctly, namely, to recover outputs in permuted order. The above analysis indicates us that if the masking procedure can be done via a universal manner, then the client might be able to unmask the permuted masked outputs correctly. Observe that the simplest way to perform unified masking is to apply a weak pseudorandom function F_s to the intermediate input $\mathsf{H}(x)$, where H is a cryptographic hash function that maps input x to the domain of F_s. To enable efficient unmasking, we further require that F_s is a permutation and commutative with respect to F_k. This yields a simple pOPRF construction from commutative weak pseudorandom permutation. More precisely, to build pOPRF, the server picks a random PRP key k for F, while the client with input $X = (x_1, \ldots, x_n)$ picks a random PRP key s for F. The client then sends $\{F_s(\mathsf{H}(x_i))\}_{i\in[n]}$ to the server. Upon receiving the masked intermediate inputs, the server applies F_k to them, then sends the results in permuted order, a.k.a. $\{F_k(F_s(\mathsf{H}(x_{\pi(i)})))\}_{i\in[n]}$. Finally, the client applies F_s^{-1} to the permuted masked outputs, and will obtain $\{F_k(\mathsf{H}(x_{\pi(i)}))\}_{i\in[n]}$ by the commutative property.

Note that many efficient OPRF constructions [PRTY19, CM20, RS21] seem not amenable to pOPRF due to the lack of nice algebra structures. This somehow explains the efficiency gap between the state-of-the-art PSI and PCSI/PSU.

1.4 Related Works

We review previous PSI-card, PSI-card-sum and PSU protocols that are relevant to our work. Ion et al. [IKN+20] showed how to transform single-point OPRF-based [PSZ14, KKRT16], garbled Bloom filter-based [DCW13, RR17], and DDH-based [HFH99] PSI protocols into ones for computing PSI-card-sum by leveraging additively homomorphic encryption (AHE). However, their conversions are inefficient due to the usage of AHE, and as noted by the authors, detailed conversions to each category of protocols differ significantly, especially in the way of making use of the underlying AHE. In contrast, we distill Sigma-mqPMT from a broad class of PSI protocols, then show how to tweak it to mqRPMT* in a generic and black-box manner, without relying on any additional cryptographic tools. Our abstraction of Sigma-mqPMT is more broadly applicable, and our conversion works at a higher level. Miao et al. [MPR+20] put forward shuffled distributed oblivious PRF as a central tool to build PSI-card-sum with malicious security. Compared to shuffled distributed OPRF, our notion of permuted OPRF is much simpler and should be best viewed as a useful extension of standard OPRF. The conceptual simplicity lends it to be easily built from commutative weak pseudorandom permutation and find more potential applications. For example, permuted OPRF immediately implies permuted matrix private equality test, which is a key tool in building FHE-based PSU [TCLZ23]. Davidson and Cid [DC17] proposed a framework for constructing PSI, PSU, and PSI-card. Their protocols have linear complexity, but both the computation and

communication complexity additionally rely on the statistical security parameter λ (a typical concrete choice is 40), resulting in low performance in practice. Kolesnikov et al. [KRTW19] proposed the notion of reverse private membership test (RPMT), then used it to build a PSU protocol whose performance is much better than [DC17]. Garimella et al. [GMR+21] proposed a framework for all private set operations from permuted characteristic, which could be viewed as a variant of RPMT. Nevertheless, the oblivious shuffle in permuted characteristic functionality is not necessary for PSO, but seems unavoidable due to the use of oblivious switching networks, which in turn incurs superlinear complexity to permuted characteristic protocol and all the enabling PSO protocols. Besides, we note that the PSI-card-sum functionality defined in [GMR+21] differs from the original functionality defined in [IKN+20]. The distinction is that in the original functionality of PSI-card-sum, both parties are given the cardinality of intersection, and the party initially holding values is also given the intersection sum, while in the functionality described in [GMR+21], the party without holding values is given the cardinality and sum of the intersection. To distinguish this subtle difference, we refer to the functionality presented in [GMR+21] as reverse PSI-card-sum.

Very recently, Zhang et al. [ZCL+23] extended the notion of RPMT [KRTW19] to multi-query RPMT (mqRPMT), and proposed a generic construction from oblivious key-value store (OKVS) [GPR+21], set-membership encryption and oblivious vector decryption-then-test protocol. By instantiating their generic construction from symmetric-key and public-key encryption respectively, they obtained two concrete mqRPMT protocols with linear complexity. However, their two mqRPMT protocols have a large multiplicative constant (the statistical security parameter) in computation complexity, and so is the resulting PSU protocol. Besides, as noted by the authors, their more efficient PKE-based mqRPMT protocol is leaky, failing to meet the standard security. Compared with their work, our generic construction of mqRPMT is much simpler, and our two concrete instantiations meet the standard definition yet achieve strict linear complexity.[2] Moreover, we identify mqRPMT as a central building block for a family of private set operations, while their focus is limited to PSU.

Other Related Works. PSO are primarily designed for the balanced scenario, where the sizes of two sets are approximately the same. Recently, a line of research has started considering the unbalanced scenario, where one set is much larger than the other. Hereafter, let the sizes of small and large sets be m and n, respectively. [CLR17, CHLR18, CMdG+21] showed how to leverage FHE to build PSI protocols suitable for unbalanced scenario with communication complexity $O(m \log n)$, which is linear to the size of small set but logarithmic to the size of large set. A body of follow-up works achieved the same complexity for other functionalities. [CHLR18] proposed circuit-PSI, PSI-card and PSI-card-sum protocols based on generic 2PC technique, and then [SJ23, WY23] provided the associated implementations. [TCLZ23] created the first unbalanced PSU

[2] In the context of PSO, strict linear complexity means that the complexity grows linearly only to the sets sizes.

protocol by tweaking the technique due to [CLR17]. Another line of research extended PSO to multi-party settings: [KMP+17,NTY21] for PSI, [CDGB22] for PSI-card(-sum), and [LG23] for PSU.

1.5 Roadmap

In Sect. 2 we recall the standard definitions of MPC and PSO. In Sect. 3 we introduce the ingredients we use to build the PSO framework. In Sect. 4 and Sect. 5 we give two generic constructions of mqRPMT from newly formalized cwPRF and newly introduced permuted OPRF, respectively. In Sect. 6 we show how to build the PSO framework from mqRPMT, and also present a modular construction of private-ID from distributed OPRF and PSU. In Sect. 7 we provide a performance analysis of our implementation, and compare our experimental results to the related state-of-the-art protocols. Due to space limit, we defer all the security proofs and additional results to the full version of this paper https://eprint.iacr.org/2022/652.

2 Preliminaries

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

2.1 MPC in the Semi-honest Model

We use the standard notion of security in the presence of semi-honest adversaries. Let Π be a two-party protocol for computing the function $f(x_1, x_2)$, where party P_i has input x_i, and $\mathsf{output}(x_1, x_2)$ be the output of both parties in the protocol. For each party P_i where $i \in \{1, 2\}$, let $\mathrm{View}_{P_i}(x_1, x_2)$ denote the view of party P_i during an honest execution of Π on inputs x_1 and x_2, which consists of P_i's input, random tape, and all messages P_i received in the protocol.

Definition 1. *Two-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 $i \in \{1, 2\}$:*

$$\{View_{P_i}(x_1, x_2), \mathsf{output}(x_1, x_2)\} \approx_c \{\mathsf{Sim}(i, x_i, f(x_1, x_2)), f(x_1, x_2)\}$$

Roughly speaking, a protocol is secure if P_i 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. We call the two parties engaging in PSO the *sender* and the *receiver*. The sender holds a set X of size n_1, and the receiver holds a set Y of size n_2 (we set $n_1 = n_2 = n$ in the balanced setting). Figure 1 formally defines the ideal functionality for PSO that computes the intersection, union, cardinality, intersection sum with cardinality and intersection secret-sharing with cardinality.

Parameters: The receiver P_1's input size n_1 and the sender P_2's input size n_2.
Inputs: The receiver P_1 inputs a set of elements $Y = \{y_1, \ldots, y_{n_1}\}$ where $y_i \in \{0,1\}^\ell$. The sender P_2 inputs a set of elements $X = \{x_1, \ldots, x_{n_2}\}$ where $x_i \in \{0,1\}^\ell$ and possibly a set of values $V = \{v_1, \ldots, v_{n_2}\}$ where $v_i \in \mathbb{Z}_p$ for some integer modular p.
Output:

- **intersection:** The receiver P_1 gets $X \cap Y$.
- **union:** The receiver P_1 gets $X \cup Y$.
- **union*:** The receiver P_1 gets $X \cup Y$. The sender P_2 gets $|X \cap Y|$.
- **card:** The receiver P_1 gets $|X \cap Y|$.
- **card-sum:** The receiver P_1 gets $|X \cap Y|$. The sender P_2 gets $|X \cap Y|$ and $S = \sum_{i:x_i \in Y} v_i$.
- **card-secret-sharing:** The receiver P_1 gets $|X \cap Y|$ and $\{z_i^1\}_{i \in [n_2]}$. The sender P_2 gets $\{z_i^2\}_{i \in [n_2]}$. For each (z_i^1, z_i^2), $z_i^1 \oplus z_i^2 = x_i$ if $x_i \in Y$ and $z_i^1 \oplus z_i^2 = 0$ otherwise.

Fig. 1. Ideal functionality $\mathcal{F}_{\mathsf{PSO}}$ for PSO

In this work, we restrict ourselves to two-party PSO with semi-honest security in the balanced setting.

3 Protocol Building Blocks

3.1 Oblivious Transfer

Oblivious Transfer (OT) [Rab05] is a central cryptographic primitive in the area of secure computation. In the most common 1-out-of-2 OT, a sender with two input strings (m_0, m_1) interacts with a receiver with an input choice bit $b \in \{0,1\}$, and finally the receiver only learns m_b while the sender learns nothing. In some cases, it suffices to use a "one-sided" version of OT, which conditionally transfers the only item of the sender or nothing to the receiver depending on the choice bit. Though expensive public-key operations are unavoidable for a single OT, a powerful technique called OT extension [IKNP03, KK13, ALSZ15] allows one to carry out n OTs by only performing $O(\kappa)$ public-key operations and $O(n)$ fast symmetric-key operations.

3.2 Multi-query Reverse Private Membership Test

Multi-query reverse private membership test (mqRPMT) [ZCL+23] is a protocol where the client with input vector $(x_1, \ldots, x_n)$ interacts with a server holding a set Y. As a result, the server learns only a bit vector $(e_1, \ldots, e_n)$ in which e_i indicates that whether $x_i \in Y$. Figure 2 formally defines the ideal functionality for mqRPMT. We also consider a relaxed version of mqRPMT called mqRPMT*, in which the client is also given $|X \cap Y|$.

Parameters: The server P_1's set size n_1 and number of RPMT queries n_2 by the client P_2.
Inputs: The server P_1 inputs a set $Y = (y_1, \ldots, y_{n_1})$, where $y_i \in \{0,1\}^\ell$. The client P_2 inputs a set $X = (x_1, \ldots, x_{n_2})$ (should be interpreted as a vector), where $x_i \in \{0,1\}^\ell$.
Output: The server P_1 gets a vector $\vec{e} = (e_1, \ldots, e_{n_2}) \in \{0,1\}^{n_2}$, where $e_i = 1$ if $x_i \in Y$ and $e_i = 0$ otherwise. The client P_2 gets nothing.

Fig. 2. Ideal functionality $\mathcal{F}_{\mathsf{mqRPMT}}$ for multi-query RPMT

4 The First Generic Construction of mqRPMT

4.1 Definition of Commutative Weak PRF

We first formally define two standard properties for keyed functions.

Composable. For a family of keyed functions $F : K \times D \to R$, 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 functions, we say it is commutative if: $\forall k_1, k_2 \in K, \forall x \in D : F_{k_1}(F_{k_2}(x)) = F_{k_2}(F_{k_1}(x))$.

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 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. 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 2 (Commutative Weak PRF). *Let F be a family of keyed functions $K \times D \to D$. F is called commutative weak PRF if it satisfies weak pseudorandomness and commutative property simultaneously. If F is a permutation, we say F is a commutative weak pseudorandom permutation (cwPRP).*

Remark 1 (cwPRF vs. Commutative Encryption). We note that our notion of cwPRF is similar to but strictly weaker than a previous notion called commutative encryption [AES03]. The difference is that cwPRF neither requires F_k be a permutation nor F_k^{-1} be efficiently computable.

4.2 Construction of Commutative Weak PRF

We observe that the classic weak PRF based on the DDH assumption already satisfies commutative property. This gives us a simple cwPRF construction from the DDH-like assumption. It is still interesting to know if cwPRF can be built from other assumptions. Note that cwPRF naturally yields a non-interactive key exchange (NIKE) protocol, while the recent result of Guo et al. [GKRS22] indicated that it would be difficult to construct NIKE from lattice-based assumptions. Therefore, giving lattice-based cwPRF or proving impossibility will lead to progress on some other well-studied questions in cryptography.

4.3 mqRPMT from Commutative Weak PRF

In Fig. 3, 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: The server P_1's set size n_1 and the client P_2's set size n_2, cwPRF $F : K \times D \to D$, and hash function $\mathsf{H} : \{0,1\}^\ell \to D$.
Inputs: The server P_1 inputs a set $Y = \{y_1, \ldots, y_{n_1}\}$, where $y_i \in \{0,1\}^\ell$. The client P_2 inputs a set $X = \{x_1, \ldots, x_{n_2}\}$ (should be interpreted as a vector), where $x_i \in \{0,1\}^\ell$.
Protocol:

1. P_1 picks $k_1 \xleftarrow{R} K$, then computes and sends $\{F_{k_1}(\mathsf{H}(y_i))\}_{i\in[n_1]}$ to P_2.
2. P_2 picks $k_2 \xleftarrow{R} K$, then:
   (a) computes and sends $\{F_{k_2}(\mathsf{H}(x_i))\}_{i\in[n_2]}$ to P_1.
   (b) computes $\{F_{k_2}(F_{k_1}(\mathsf{H}(y_i)))\}_{i\in[n_1]}$, picks a random permutation π over $[n_1]$, then sends $\{F_{k_2}(F_{k_1}(\mathsf{H}(y_{\pi(i)})))\}_{i\in[n_1]}$ to P_1. Instead of explicit shuffling, an alternative choice is inserting $\{F_{k_2}(F_{k_1}(\mathsf{H}(y_i)))\}_{i\in[n_1]}$ to a Bloom filter and then sending the filter to P_1. We slightly abuse the notation, and still use Ω to denote the Bloom filter.
3. P_1 computes $\{F_{k_1}(F_{k_2}(\mathsf{H}(x_i)))\}_{i\in[n_2]}$, sets $e_i = 1$ iff $F_{k_1}(F_{k_2}(\mathsf{H}(x_i))) \in \Omega$.

Fig. 3. Multi-query RPMT from commutative weak PRF

Correctness. The protocol is correct except the event 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. In what follows, we fix a tuple (x, y) such that $x \neq y$. Let E_0 be the event $\mathsf{H}(x) = \mathsf{H}(y)$. By the collision resistance of H, we have $\Pr[E_0] = 2^{-\kappa}$. Let E_1 be the event 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}-$F_{k_2}(\mathsf{H}(x)) \neq F_{k_2}(\mathsf{H}(y)) \wedge F_{k_1}(F_{k_2}(\mathsf{H}(x))) = F_{k_1}(F_{k_2}(\mathsf{H}(y)))$. By the weak pseudorandomness of F, we have $\Pr[E_{10}] = (1 - \Pr[E_0]) \cdot 1/|D|$, and $\Pr[E_{11}] = (1 - \Pr[E_0]) \cdot (1 - 1/|D|) \cdot 1/|D|$. If $|D| = \omega(\kappa)$, then both $\Pr[E_0]$, $\Pr[E_{10}]$ and $\Pr[E_{11}]$ are negligible in κ. Therefore, by union bound we have $\Pr[E] \leq n_1 n_2 \cdot (\Pr[E_0] + \Pr[E_{10}] + \Pr[E_{11}]) = \mathsf{negl}(\kappa)$.

Theorem 1. *The mqRPMT protocol described in Fig. 3 is secure in the semi-honest model assuming* H *is a random oracle and* F *is a family of cwPRFs.*

Complexity Analysis. We now analyze the complexity of the above (n_1, n_2)-mqRPMT protocol. Simple calculation shows that the total computation cost is $(n_1 + n_2)$ hashings, $(2n_1 + 2n_2)$ evaluations of cwPRF F, n_2 lookups whose complexity is $O(1)$, and one shuffling whose complexity is $O(n_1)$, while the total communication cost is $(2n_1 + n_2)$ elements in range D. In summary, both the computation and communication complexity are strictly linear in set sizes.

Optimization. The protocol can be further improved by inserting the elements $\{F_{k_2}(F_{k_1}(\mathsf{H}(y_i)))\}_{i\in[n_1]}$ to a Bloom filter instead of explicitly shuffling them in the last move. In this way, the length of last message can be reduced from to n_1 group elements to $1.44\lambda \cdot n_1$ bits (with false positive probability $2^{-\lambda}$), where λ is the statistical security parameter and the typical choice is 40.

It is worth to highlight that our usage of Bloom filter is novel here since we additionally exploit its order-hiding property to ensure security[3]. To the best of knowledge, Bloom filter merely serves as a space-efficient data structure in previous works [KLS+17, RA18] to reduce communication cost.

5 The Second Generic Construction of mqRPMT

5.1 Definition of Permuted OPRF

An oblivious pseudorandom function (OPRF) [FIPR05] is a two-party protocol in which the server learns (or chooses) a PRF key k and the client learns $F_k(x_1), \ldots, F_k(x_n)$, where F is a pseudorandom function (PRF) and $(x_1, \ldots, x_n)$ are the client's inputs. Nothing about the client's inputs is revealed to the server and nothing more about the key k is revealed to the client.

We consider an extension of OPRF called permuted OPRF (pOPRF). Roughly speaking, the server additionally picks a random permutation π over $[n]$, and the client learns its PRF values in permuted order, namely, $y_i = F_k(x_{\pi(i)})$. Figure 4 formally defines the ideal functionality for pOPRF.

[3] Formally, order-hiding property means that the data structure does not reveal the adding order of elements. Recall that an empty Bloom filter is a bit array of m bits (all set to 0), and adding an element x is done by setting the bits at positions $\{h_1(x), \ldots, h_k(x)\}$. to be 1, where $\{h_i\}_{i=1}^k$ are k distinct hash functions. Clearly, Bloom filter satisfies order-hiding property since the resulting Bloom filter is independent of the adding order. We also stress that the choice of Bloom filter is not arbitrary here, cause other filters such as Cuckoo filter and Vacuum filter do not satisfy order-hiding property.

Parameters: Number of OPRF queries n.
Inputs: The server P_1 inputs nothing. The client P_2 inputs a set $X = (x_1, \ldots, x_n)$, where $x_i \in \{0,1\}^\ell$.
Output: The server P_1 gets a random PRF key k and a random permutation π over $[n]$. The client P_2 gets $y_i = F_k(x_{\pi(i)})$.

Fig. 4. Ideal functionality $\mathcal{F}_{\mathsf{pOPRF}}$ for permuted OPRF

5.2 Construction of Permuted OPRF

As we sketched in the introduction part, we can create a permuted OPRF from cwPRP F with the help of random oracle. At a high level, the universal masking procedure is done by applying a weak PRF $F_s(\cdot)$ to $\mathsf{H}(x)$, and the unmasking process is enabled by the commutative property of F and the fact that $F_s(\cdot)$ is an efficiently invertible permutation. We depict the construction in Fig. 5.

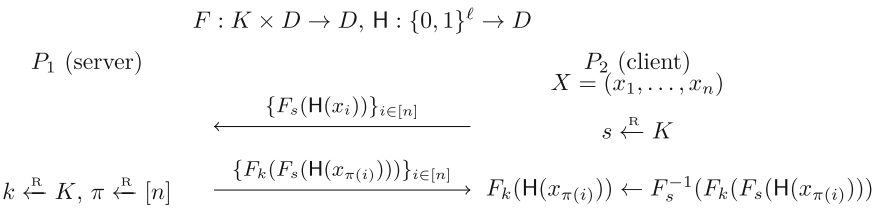

Fig. 5. Permuted OPRF from cwPRP

Theorem 2. *The above permuted OPRF protocol is secure in the semi-honest model assuming* H *is a random oracle and* F *is a family of cwPRPs.*

5.3 mqRPMT from Permuted OPRF

In Fig. 6, we show how to build mqRPMT from permuted OPRF for $F : K \times D \to R$. For simplicity, we assume that $\{0,1\}^\ell \subseteq D$. Otherwise, we can always map $\{0,1\}^\ell$ to D using a collision resistant hash function.

Parameters: The server P_1's set size n_1 and the client P_2's set size n_2, a permuted OPRF for $F : K \times D \to R$.
Inputs: The server P_1 inputs a set $Y = \{y_1, \ldots, y_{n_1}\}$, where $y_i \in \{0,1\}^\ell$. The client P_2 inputs a set $X = \{x_1, \ldots, x_{n_2}\}$, where $x_i \in \{0,1\}^\ell$.
Protocol:

1. P_1 with inputs $Y = \{y_1, \ldots, y_{n_1}\}$ and P_2 engage in a permuted OPRF protocol. P_1 acts as the pOPRF's client, while P_2 acts as the pOPRF's server. At the end of the protocol, P_1 obtains $\{F_k(y_{\pi(i)})\}_{i\in[n_1]}$, P_2 obtains a PRF key k and a random permutation π over $[n_1]$.
2. P_2 computes and sends $(F_k(x_1), \ldots, F_k(x_{n_1}))$ to P_1.
3. P_1 sets $e_i = 1$ iff $F_k(x_i) \in \{F_k(y_{\pi(i)})\}_{i\in[n_1]}$.

Fig. 6. mqRPMT from permuted OPRF

Correctness. The above protocol is correct except the event $E = \vee_{i,j} E_{ij}$ occurs, where E_{ij} denotes $F_k(x_i) = F_k(y_j)$ but $x_i \neq y_j$. By pseudorandomness of F, we have $\Pr[E_{ij}] = 2^{-\ell}$. Apply the union bound, we have $\Pr[E] \leq n_1 n_2 \cdot \Pr[E_{ij}] = n_1 n_2 / 2^\ell = \mathsf{negl}(\lambda)$.

Theorem 3. *The above mqRPMT protocol described in Fig. 6 is secure in the semi-honest model assuming the security of permuted OPRF.*

Performance Analysis. We now analyze the performance the above (n_1, n_2)-mqRPMT protocol based on the cwPRP-based permuted OPRF. Simple calculation shows that the total computation cost is $(n_1 + n_2)$ hashings, $2n_1 + n_2$ evaluations, n_2 inversions, n_2 lookups whose complexity is $O(1)$, and one shuffling whose complexity is $O(n_1)$, while the total communication cost is $(2n_1 + n_2)$ group elements in range D. In summary, both the computation and communication complexities are strictly linear in set sizes.

Comparison of the Two Constructions of mqRPMT. We have presented two generic constructions of mqRPMT, the first is from cwPRF, while the second is from pOPRF. We summarize their differences as below.

- cwPRF-based construction admits fast implementation from the NIKE protocol called X25519 based on `Curve25519` (since X25519 implies a cwPRF), and can further utilize the Bloom filter to reduce communication cost as well as improve efficiency.
- Compared with the cwPRF-based construction, the pOPRF-based construction does not admit fast implementation from X25519 anymore (since X25519 does not imply a cwPRP and thus further a pOPRF), and the Bloom filter optimization is not applicable (since the set is already fixed in pOPRF phase). Nevertheless, the pOPRF-based mqRPMT construction can be viewed as a counterpart of OPRF-based mqPMT construction, and thus is more of theoretical interest. So far, we only know how to build pOPRF based on

assumptions with nice algebra structure, but not from efficient symmetric-key primitives. This somehow explains the efficiency gap between mqPMT and mqRPMT.

6 Applications of mqRPMT

6.1 PSO Framework from mqRPMT

In Fig. 7, we show how to build a PSO framework centering around mqRPMT.

Parameters: The receiver P_1's set size n_1 and the client P_2's set size n_2.
Inputs: The receiver P_1 inputs a set $Y = \{y_1, \ldots, y_{n_1}\}$, where $y_i \in \{0,1\}^\ell$. The sender P_2 inputs a set $X = \{x_1, \ldots, x_{n_2}\}$ and $V = \{v_1, \ldots, v_{n_2}\}$, where $x_i \in \{0,1\}^\ell$ and $v_i \in \mathbb{Z}_p$. Let q be a big integer greater than $n_2 \cdot p$.
Protocol:

0. P_2 shuffles the set $\{x_1, \ldots, x_{n_2}\}$ and $\{v_1, \ldots, v_{n_2}\}$ using the same random permutation over $[n_2]$. For simplicity, we still use the original notation to denote the resulting vectors after permutation.
1. P_1 (playing the role of server) with Y and P_2 (playing the role of client) with $X = \{x_1, \ldots, x_{n_2}\}$ 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_1 holding e_i and P_2 holding $(\perp, x_i)$ invoke one-sided OT n_2 times. P_1 learns $\{x_i \mid e_i = 1\}_{i \in [n_2]} = X \cap Y$.
   - **union:** P_1 holding e_i and P_2 holding $(x_i, \perp)$ invoke one-sided OT n_2 times. P_1 learns $\{x_i \mid e_i = 0\}_{i \in [n_2]} = X \backslash Y$, and outputs $\{X \backslash Y\} \cup Y = X \cup Y$.
   - **card-sum:** P_2 randomly picks $r_i \in \mathbb{Z}_q$ and computes $r' = \sum_{i=1}^{n_2} r_i \bmod q$. Subsequently, P_1 holding e_i and P_2 holding $(r_i, r_i + v_i)$ invoke 1-out-of-2 OT n_2 times. P_1 learns $S' = \sum_{i=1}^{n_2} r_i + e_i \cdot v_i \bmod q$, then sends S' and the Hamming weight of $\vec{e}$ to P_2. P_2 computes $S = (S' - r') \bmod q$.
   - **card-secret-sharing:** P_2 randomly picks $r_i \in \{0,1\}^\ell$. Subsequently, P_1 holding e_i and P_2 holding $(r_i, r_i \oplus x_i)$ invoke 1-out-of-2 OT n_2 times. P_1 learns $\{z_i\}_{i \in [n_2]}$, and thus $\{(z_i, r_i \oplus x_i)\}_{e_i = 1}$ constitutes the shares of intersection values.

Fig. 7. PSO from mqRPMT

We prove the security of the above PSO framework by the case of PSU. The security proof of other functionality is similar.

Theorem 4. *The PSU derived from the above framework is semi-honest secure by assuming the semi-honest security of mqRPMT and OT.*

In what follows, we compare the protocols derived from our framework to existing protocols with focus on conceptual differences, and defer the performance comparisons to Sect. 7.

We first compare our PSU protocol to prior PSU protocols. [KS05,Fri07, DC17] proposed the first three PSU protocols from public-key techniques, with the complexity gradually dropping from quadratic to linear. Later, [KRTW19, GMR+21,JSZ+22] proposed three PSU protocols from symmetric-key techniques. Despite their protocols achieve much better performance than previous ones based on public-key techniques, all of them require superlinear complexity. Recently, Zhang et al. [ZCL+23] created a more efficient PSU protocol with linear complexity. Both our protocol and their protocol are derived from the same core protocol—mqRPMT, but with different instantiations. Our concrete mqRPMT protocols are much simpler and efficient, yielding the first PSU protocols with strict linear complexity.

We then discuss the relationship between our PSI-card protocol and prior related protocols. Huberman et al. [HFH99] proposed the first PSI-card protocol but did not provided security proof. Agrawal et al. [AES03] explained and proved the classic protocol via the notion of "commutative encryption". Later, De Cristofaro et al. [CGT12] gave a close variant of the classic protocol. Our PSI-card protocol is generically derived from the more abstract mqRPMT, which in turn can be built from cwPRF or pOPRF. By instantiating the underlying cwPRF and pOPRF from the DDH assumption, we recover the PSI-card protocols in [HFH99,CGT12] respectively. In a nutshell, our generic mqRPMT-based PSI-card construction not only encompasses existing concrete protocols at a high level, but also readily profits from the possible improvements on the underlying mqRPMT (e.g., Bloom filter optimization and post-quantum secure realization based on the EGA assumption).

We continue to compare our PSI-card-sum protocol with closely related protocols [IKN+20,GMR+21]. As mentioned in the introduction part, the PSI-card-sum protocols presented in [IKN+20] are built from concrete primitives (e.g. DH-protocol, ROT-protocol, Phasing+OPPRF etc.) with generic 2PC techniques or AHE schemes. Compared to [IKN+20], our protocol is built from mqRPMT and lightweight OT, which is more general and efficient. The protocol presented in [GMR+21] is built from permuted characteristic (permuted mqRPMT under our terminology) and secret sharing. Compared to [GMR+21], our protocol has the following differences: (i) mqRPMT underlying our protocol is conceptually simpler than permuted characteristic. More importantly, mqRPMT admits instantiations with optimal linear complexity, while the current best instantiation of permuted characteristic requires superlinear complexity. (ii) The protocol in [GMR+21] deviates from the standard functionality (as mentioned earlier in the introduction part), while our protocol meets the standard functionality of PSI-card-sum as defined in [IKN+20]. We do so by removing the constraint $\sum_{i=1}^{n} r_i = 0$ on the receiver side (as did in [GMR+21]), and having the sender send back the masked sum value to the receiver, and the receiver finally recovers the intersection sum by unmasking.

Finally, we compare our PSI card-secret-sharing protocol to the closely related circuit-PSI [HEK12,PSTY19,RS21]. The only difference on functionality is that our protocol additionally leaks the cardinality to the receiver. Nevertheless, as pointed out in [GMR+21], in many applications of interest the functions that need to be computed already contain such leakage. Garimella et al. [GMR+21] proposed a similar functionality called secret-shared intersection, in which the parties get the shares of intersection elements. As a result, their functionality leaks the cardinality to both the sender and the receiver.

6.2 Private-ID

Recently, Buddhavarapu et al. [BKM+20] proposed a two-party functionality called private-ID, which assigns two parties, each holding a set of items, a truly random identifier per item (where identical items receive the same identifier). As a result, each party obtains identifiers to his own set, as well as identifiers associated with the union of their input sets. With private-ID, two parties can sort their private set with respect to a global set of identifiers, and then can proceed any desired private computation item by item, being assured that identical items are aligned. Buddhavarapu et al. [BKM+20] also gave a concrete DDH-based private-ID protocol. Garimella et al. [GMR+21] showed how to build private-ID from OPRF and PSU. Roughly speaking, their approach proceeds in two phases. In phase 1, P_1 holding X and P_2 holding Y run an OPRF twice by switching the roles, so that first P_1 learns k_1 and P_2 learns $F_{k_1}(y_i)$, and second P_2 learns k_2 and P_1 learns $F_{k_2}(x_i)$. The random identifier of an item z is thus defined as $id_z = F_{k_1}(z) \oplus F_{k_2}(z)$. After phase 1, both parties can compute identifiers for their own items. In phase 2, they simply engage a PSU protocol on their sets $id(X)$ and $id(Y)$ to finish private-ID.

Our method is largely inspired by the approach presented in [GMR+21]. We first observe that in phase 1, two parties essentially need to engage a *distributed* OPRF protocol, as we formally depict in Fig. 8. The random identifier of an item z is defined as $G_{k_1,k_2}(z)$, where G is a PRF determined by key (k_1, k_2). Furthermore, note that $id(X)$ and $id(Y)$ are pseudorandom, which means in phase 2 a *distributional* PSU protocol suffices, whose semi-honest security is additionally defined over the input distribution. Such relaxation may lead to remarkable efficiency improvement.

In this work, we instantiate the generic private-ID construction as below: (1) realize the distributed OPRF protocol by running currently the most efficient multi-point OPRF of [RR22] built from VOLE and improved OKVS twice in reverse order; (2) run the PSU protocol from cwPRF-based mqRPMT with the obtained two sets of pseudorandom identifiers as inputs to fulfill the private-ID functionality.

Distributional PSU. Standard security notions for MPC are defined w.r.t. any private inputs. This treatment facilitates secure composition of different protocols. We find that in certain settings it is meaningful to consider a weaker security notion by allowing the real-ideal indistinguishability to also base on the

Parameters: PRF $G : K \times D \to R$, where $K = K_1 \times K_2$.
Inputs: P_1 inputs a set $X = \{x_1, \ldots, x_{n_1}\}$, where $x_i \in D$. P_2 inputs a set $Y = \{y_1, \ldots, y_{n_2}\}$, where $y_i \in D$.
Output: P_1 gets $\{G_{k_1,k_2}(x_i)\}_{i\in[n_1]}$ and k_1. P_2 gets $\{G_{k_1,k_2}(y_i)\}_{i\in[n_2]}$ and k_2.

Fig. 8. Ideal functionality for distributed OPRF

distribution of private inputs. This is because such relaxed security suffices if the protocol's input is another protocol's output which obeys some distribution, and the relaxation may admit efficiency improvement. Suppose choosing the DDH-based distributed OPRF and DDH-based PSU in the same elliptic curve (EC) group as ingredients, faithful implementation according to the above recipe requires $4n$ hash-to-point operations. Observe that the output of distributed DDH-based OPRF are already pseudorandom EC points. In this case, it suffices to use distributional DDH-based PSU instead, and thus can save $2n$ hash-to-point operations, which are costly in the real-world implementation.

7 Performance

We describe details of our implementation and report the performance of the following set operations: (1) **psi:** intersection of the sets; (2) **psi-card:** cardinality of the intersection; (3) **psi-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.

7.1 Implementation Details

Our protocols are written in C++, which can be found at https://github.com/yuchen1024/Kunlun/mpc. The code is organized in a modular and unified fashion in consistent with our paper: first implement the core mqRPMT protocol, then build various PSO protocols upon it. Besides, it only requires OpenSSL [Opea] as the main 3rd party library, and can smoothly run on both Linux and x86_64 MacOS platforms.

7.2 Experimental Setup

We run all our protocols and related protocols on Ubuntu 20.04 with a single Intel i7-11700 2.50 GHz CPU (8 physical cores) and 16 GB RAM. We simulate the network connection using Linux `tc` command. In the LAN setting, the bandwidth is set to be 10 Gbps with 0.1 ms RTT latency. In the WAN setting, the bandwidth is set to be 50 Mbps with 80 ms RTT latency. We use `iptables` command to calculate the communication cost, and use running time (the maximal time from

protocol begin to end in the sender and the receiver side, including messages transmission time) to measure the computation cost. For a fair comparison, we stick to the following setting for all protocols being evaluated:

- We set the computational security parameter $\kappa = 128$ and the statistical security parameter $\lambda = 40$.
- We test the balanced scenario by setting the input set size $n_1 = n_2$ (our implementation supports unbalanced scenario as well), and randomly generate two input sets with 128 bits item length conditioned on the intersection size being roughly $0.5n$. The exception is the implementation of protocol in [GMR+21], whose item length is set as 61 bits in default and cannot exceed 64 bits since each element is represented as a `uint64_t` integer.
- The PSI-card-sum protocol [IKN+20] and the private-id protocol [BKM+20] are two of the related works we are going to compare. The former implementation is built upon `NIST P-256` (also known as `secp256r1` and `prime256v1`), while the latter implementation is built upon `Curve25519`. For a comprehensive comparison, our implementation supports flexible switching between standard elliptic curve `NIST P-256` and special elliptic curve `Curve25519`. For protocols based on `NIST P-256`, we denote the ones not using or using point compression technique with ♦ and ▼ respectively. For protocols based on `Curve25519`, we denote them with ★.

7.3 Evaluation of mqRPMT

We first report the performance of our cwPRF-based mqRPMT protocol (optimized with Bloom filter) described in Sect. 4.3, which dominates the communication and computation overheads of its enabling PSO protocols. We test our protocol up to 4 threads, since both the server and the client run on a single CPU with 8 physical cores. Our cwPRF-based mqRPMT achieves optimal linear complexity, and thus is scalable, which is demonstrated by the experimental results in Table 1. Moreover, the computation tasks on both sides in our cwPRF-based mqRPMT are highly parallelable, thus we can effortlessly using OpenMP [Opeb] to make the program multi-threaded.

7.4 Benchmark Comparison of PSO Protocols

We derive all kinds of PSO protocols from cwPRF-based mqRPMT protocol, and compare them with the state-of-the-art related protocols. We report the performances for three input sizes $n = \{2^{12}, 2^{16}, 2^{20}\}$ all executed over a single thread in LAN and WAN settings. When testing the PSI-card, PSI-card-sum and PSU protocols in [GMR+21], we set the number of mega-bins as $\{1305, 16130, 210255\}$ and the number of items in each mega-bin as $\{51, 62, 72\}$ for set sizes $n = \{2^{12}, 2^{16}, 2^{20}\}$ respectively. These parameter choices have been tested to be much more optimal than their default ones.

PSI. We first compare our mqRPMT-based PSI protocol to the classical DH-PSI protocol reported in [PRTY19] and the one re-implemented by ourselves.

Table 1. Communication cost and running time of mqRPMT.

Protocol	T	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}
mqRPMT$^{\blacklozenge}$	1	0.50	7.20	114.16	1.39	9.68	136.27	0.52	8.35	133.6
	2	0.31	3.89	62.09	1.14	6.54	86.60			
	4	0.22	2.37	40.41	1.11	5.08	62.77			
Speedup		1.6–2.3 ×	1.9–3.0 ×	1.8–2.8 ×	1.2–1.3 ×	1.5–1.9 ×	1.6-2.2 ×	–	–	–
mqRPMT$^{\blacktriangledown}$	1	0.50	8.00	128.00	1.35	10.15	141.52	0.27	4.35	69.6
	2	0.32	5.05	80.69	1.18	7.11	94.19			
	4	0.23	3.54	58.40	1.08	5.54	71.26			
Speedup		1.6-2.2 ×	1.6-2.3 ×	1.6-2.2 ×	1.1-1.3×	1.4-1.8 ×	1.5-2 ×	–	–	–
mqRPMT$^{\bigstar}$	1	0.26	3.51	54.85	0.81	5.41	68.68	0.26	4.23	67.66
	2	0.15	1.79	28.24	0.75	3.83	41.38			
	4	0.10	1.07	15.32	0.72	3.09	28.31			
Speedup		1.7-2.6 ×	2.0-3.3 ×	1.9-3.6 ×	1.1-1.1 ×	1.4-1.8 ×	1.7-2.4 ×	–	–	–

We remark that the PSI protocols in comparison are not competitive to the state-of-the-art PSI protocol. We include them merely for illustrative purpose and completeness. PSI protocols build upon public-key techniques are used to be thought inefficient, but our experiment results demonstrate that they could be practical by leveraging modern crypto library and carefully choosing optimized parameters. By using fast elliptic curve operations provided by OpenSSL, our mqRPMT-based PSI protocol is $3.4 - 10.5\times$ faster than the DH-PSI protocol[4] implemented in [PRTY19]. By further exploiting the features of `Curve25519` in important ways (see Sect. 7.5 in details), our re-implemented DH-PSI protocol (denoted by DH-PSI$^{\bigstar}$) achieves a $6.3 - 26.1\times$ speedup, which is arguably the most efficient DH-PSI implementation known to date (Table 2).

Recently, Rosulek and Trieu [RT21] proposed a PSI protocol based on Diffie-Hellman key agreement, which requires the least time and communication of any known PSI protocols for small sets. Somewhat surprisingly, Table 3 shows that for small sets our mqRPMT-based PSI protocol is faster than their protocol in the LAN setting, and our re-implemented DH-PSI protocol is much faster than their protocol in all settings with marginally larger communication cost.

PSI-card. We compare our mqRPMT-based PSI-card protocol to the PSI-card protocol in [GMR+21]. Table 4 shows that our protocol achieves a $2.4 - 10.5\times$ speedup, and reduces the communication cost by a factor of $10.9 - 14.8\times$.

[4] We remark that except inefficiency, their implementation also has a severe security issue. More precisely, they realize the hash-to-point function $\{0,1\}^* \to \mathbb{G}$ as $x \mapsto g^{\mathsf{H}(x)}$, where H is some cryptographic hash function. However, such hash-to-point function cannot be modeled as random oracle anymore since it exposes the algebra structure of output in the clear, and hence totally compromise security. Similar issue also appears in `libPSI`.

Table 2. Communication cost and running time of PSI protocol.

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}
[PRTY19]*	5.51	88.64	1418.20	5.82	90.79	1498.67	0.30	4.74	76.60
Our PSI$^\blacklozenge$	**0.50**	**7.24**	**114.66**	**1.71**	**10.50**	**142.45**	**0.67**	**10.38**	**165.77**
Our PSI$^\blacktriangledown$	**0.55**	**8.04**	**128.18**	**1.73**	**11.02**	**148.18**	**0.41**	**6.38**	**101.63**
Our PSI*	**0.29**	**3.56**	**55.11**	**1.19**	**6.38**	**75.56**	**0.40**	**6.25**	**99.71**
DH-PSI*	**0.22**	**3.39**	**54.79**	**0.92**	**5.57**	**69.31**	**0.28**	**4.57**	**74.1**

Table 3. Communication cost and running time of PSI protocol on small sets.

PSI	Running time (ms)						Comm. (KB)		
	LAN			WAN			total		
	2^8	2^9	2^{10}	2^8	2^9	2^{10}	2^8	2^9	2^{10}
[RT21]*	50.0	71.0	147.3	224.1	260.2	457.9	17.9	34.1	66.3
Our PSI*	**41.9**	**69.5**	**99.3**	**577.0**	**582.9**	**646.1**	**38.6**	**63.5**	**113.3**
DH-PSI*	**16.49**	**31.80**	**56.91**	**210.42**	**227.33**	**252.32**	**18.48**	**36.68**	**72.8**

Table 4. Communication cost and running time of PSI-card protocol.

PSI-card	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}
[GMR+21]	1.00	8.41	126.01	8.60	27.46	323.52	2.93	55.49	1030
Our PSI-card$^\blacklozenge$	**0.49**	**7.20**	**114.31**	**1.30**	**9.68**	**136.06**	**0.53**	**8.59**	**137.31**
Our PSI-card$^\blacktriangledown$	**0.53**	**8.00**	**128.00**	**1.35**	**10.16**	**141.31**	**0.28**	**4.58**	**73.20**
Our PSI-card*	**0.27**	**3.51**	**54.89**	**0.82**	**5.42**	**68.31**	**0.27**	**4.46**	**71.30**

PSI-card-sum. We compare our mqRPMT-based PSI-card-sum protocol to the PSI-card-sum protocol (the most efficient and also the deployed one based on DH-protocol+Paillier) in [IKN+20].[5] As shown in Table 5, compared with the

[5] We do not compare the protocol described in [GMR+21] since its functionality is not the standard one, as we elaborated in the introduction. Putting aside the functionality difference, our protocol is still more advantageous than the protocol of [GMR+21] since our random masking trick is much simpler and efficient than the AHE-based technique adopted by the latter. In more detail, the upper bound of intersection sum in [GMR+21] is closely tied to the AHE scheme in use, which requires sophisticated parameter tuning and ciphertext packing techniques. Whereas in our protocol, the upper bound of intersection sum can be flexibly adjusted according to applications.

protocol presented in [IKN+20], our protocol achieves a $28.5 - 76.3\times$ improvement in running time and a $7.4\times$ reduction in communication cost.

Table 5. Communication cost and running time of PSI-card-sum protocol.

PSI-card-sum	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}
[IKN+20]$^\blacktriangledown$ (deployed)	23.64	176.34	–	30.10	186.29	–	2.72	43.24	–
Our PSI-card-sum$^\blacklozenge$	**0.51**	**7.22**	**113.66**	**1.46**	**9.68**	**136.27**	**0.65**	**10.12**	**161.40**
Our PSI-card-sum$^\blacktriangledown$	**0.57**	**8.12**	**129.66**	**1.94**	**11.83**	**157.66**	**0.39**	**6.10**	**97.34**
Our PSI-card-sum$^\bigstar$	**0.31**	**3.73**	**57.44**	**1.36**	**6.53**	**76.16**	**0.37**	**5.75**	**95.30**

We assume each associated value is a non-negative integer in $[0, 2^{32})$ conditioned on the upper bound of intersection sum being 2^{32}. We note that the implementation of [IKN+20] only works in our environment at set sizes 2^{12} and 2^{16}. For set size 2^{20}, we encounter a run time error reported in [Pri] that has not been fixed yet. The corresponding cells are marked with "–".

PSU. We compare our mqRPMT-based PSU protocol to the state-of-the-art PSU protocols in [GMR+21, ZCL+23, JSZ+22]. [ZCL+23] provides two PSU protocols from public-key and symmetric-key respectively. [JSZ+22] also provides two PSU protocols called PSU-S and PSU-R. We choose the most efficient PKE-PSU [ZCL+23] and PSU-R [JSZ+22] for comparison. Among all the mentioned PSU protocols, only our PSU protocol achieves strict linear communication and computation complexity. The experimental results in Table 6 indicate that our PSU protocol is the most superior one. Comparing to the state-of-the-art PSU protocol of [ZCL+23][6], our protocol achieves a $2.7 - 17\times$ improvement in running time and a $2\times$ reduction in communication cost.

Private-ID. We compare our private-ID protocol described in Sect. 6.2 to the state-of-the-art protocols in [BKM+20, GMR+21]. As shown in Table 7, our private-ID protocol achieves a $2.7 - 4.8\times$ speedup comparing to the current most computation efficient private-ID protocol [GMR+21], while requires $1.3\times$ less communication for sufficiently large sets[7] than the current most communication efficient private-ID protocol [BKM+20]. Hence, our private-ID protocol is arguably the most computation and communication efficient one to date.

[6] A recent work [BPSY23] proposed a new construction of OKVS and used it to improve the performance of the PSU protocol in [ZCL+23] by approximately 30%. However, if suitable parameters of the new OKVS construction exist when set sizes are less than 2^{10} is unclear. Our PSU protocol still performs the best even comparing with their optimized protocol.

[7] We note that our protocol requires more communication for sets of size 2^{12}. This is because the underlying multi-point OPRF [RR22] is built using VOLE, which has noticeable startup cost, arising relatively large constant terms in the computation and communication complexities of multi-point OPRF.

Table 6. Communication cost and running time of PSU protocol.

PSU	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}
[GMR+21]	1.16	10.06	151.34	10.34	38.52	349.43	3.85	67.38	1155
[ZCL+23]$^{\blacklozenge}$	4.87	12.19	141.38	5.78	15.75	182.88	1.35	21.41	342.38
[ZCL+23]$^{\blacktriangledown}$	5.10	15.13	187.29	5.82	17.37	210.06	0.77	12.20	195.17
[JSZ+22]	2.29	8.50	516.04	5.33	27.00	736.30	3.59	70.37	1341.55
Our PSU$^{\blacklozenge}$	**0.52**	**7.27**	**114.44**	**1.70**	**10.56**	**143.29**	**0.69**	**10.61**	**169.37**
Our PSU$^{\blacktriangledown}$	**0.57**	**8.04**	**128.20**	**1.76**	**10.92**	**148.15**	**0.42**	**6.61**	**105.23**
Our PSU$^{\bigstar}$	**0.30**	**3.55**	**55.48**	**1.19**	**6.38**	**74.96**	**0.41**	**6.48**	**103.31**

Table 7. Communication cost and running time of private-ID protocol.

Private-ID	Running time (ms)						Comm. (MB)		
	LAN			WAN			total		
	2^{12}	2^{16}	2^{20}	2^{12}	2^{16}	2^{20}	2^{12}	2^{16}	2^{20}
[GMR+21]	1.65	11.023	158.76	13.82	43.00	385.12	4.43	76.57	1293
[BKM+20]$^{\bigstar}$	2.21	37.56	671.75	7.98	46.97	710.94	1.00	15.97	226.70
Our Private-ID$^{\blacklozenge}$	**0.55**	**7.28**	**115.63**	**5.34**	**14.83**	**163.43**	**3.12**	**16.91**	**237.55**
Our Private-ID$^{\blacktriangledown}$	**0.65**	**8.43**	**134.16**	**5.69**	**15.68**	**169.05**	**2.85**	**12.91**	**173.50**
Our Private-ID$^{\bigstar}$	**0.34**	**3.78**	**59.76**	**5.04**	**10.87**	**94.89**	**2.82**	**12.74**	**171.54**

7.5 Tips for ECC-Based Implementations

In what follows, we summarize the lessons we learned during the implementation of ECC-based protocols, with the hope to uncover some dark details and correct imprecise impressions.

We first highlight the following two caveats when implementing with standard elliptic curves:

Pros and cons of point compression technique. Point compression is a standard trick in elliptic-curve cryptography (ECC), which can roughly reduce the storage cost of EC point by half, at the cost of performing decompression when needed. Point decompression was empirically thought to be cheap, but experiment indicates that it could be as expensive as scalar multiplication. Our perspective is that point compression offers a natural trade-offs between communication and computation. The above experimental results demonstrate that the total running time gives a large weight to communication cost in bandwidth constrained scenarios. Therefore, in the WAN setting (involving parties cannot be co-located) we recommend not to apply point compression trick, while in the LAN setting (involving parties are co-located) we recommend to apply point compression

trick. A quick take-away is that point compression trick pays off in the setting where communication is much more expensive than computation.

Tricky hash-to-point operation. The hash to point operation is very tricky in ECC. So far, there is no universal method to securely map arbitrary bit strings to points on elliptic curves. Here, the vague term "securely" indicates the hash function could be modeled as a random oracle. A folklore method is the "try-and-increment" algorithm [BLS01], which is also the method adopted in this work. Nevertheless, even such simple hash-to-point operation could be as expensive as scalar multiplication, which should be avoided if possible.

Regarding the two caveats discussed above, the following questions arise: (1) is it possible to get the best of two worlds of point compression; (2) could the hash-to-point operation be cheaper. Luckily, the answers are affirmative under some circumstances.

With the aim to avoid ASIACRYPT potential implementation pitfalls, Bernstein [Ber06] built a Montgomery curve called `Curve25519` in 2006, in which 25519 indicates that the characteristic of the base field is $2^{255} - 19$. Due to many efficiency/security advantages, `Curve25519` has been widely deployed in numerous applications and has become the *de facto* alternative to `NIST P-256`. Here, we highlight two notable features of `Curve25519`: (i) one can perform *somewhat* scalar multiplication using only X coordinate; (ii) by design, any 32-byte bit array corresponds the X coordinate of a valid EC point. Please refer to [Kle21] for more technique details. Exactly by leveraging these two features, Bernstein constructed a non-interactive key exchange (NIKE) protocol called X25519 based on `Curve25519`, which outperforms other EC NIKE protocols since it only depends on the X coordinate of the EC point.

Recall that our cwPRF-based mqRPMT protocol can be realized from any EC NIKE protocol and associated hash-to-point function. Compared with standard EC curves like `NIST P-256`, `Curve25519` is particularly beneficial for the implementation of our protocol. More precisely, feature (i) brings us the best of two worlds of point compression (without making trade-off anymore), while feature (ii) makes the hash-to-point function almost free, simply hashing the input to a 32-byte bit array via standard cryptographic hash function. To the best of our knowledge, this is the first time that `Curve25519` fully unleashes its power in the area of private set operations. In general, `Curve25519` is a perfect match for schemes/protocols enabled by cwPRF.

Public-key operations are always rashly thought to be much expensive than symmetric-key operations, and thus the design discipline of many practical protocols opts to avoid public-key operations as much as possible. Our experimental results indicate that this impression is not precise anymore after rapid advances on ECC-based cryptography in recent years. By leveraging optimized implementation, public-key operations could be as efficient as symmetric-key operations. As a concrete example, in EC group with 128 bit security level one EC point scalar operation takes 0.026 ms and one EC point addition takes 0.00028 ms on a laptop.

8 Summary and Perspective

This work demonstrates that mqRPMT protocol is complete for most private set operations. In particular, we created a unified PSO framework from mqRPMT, which is rather attractive given its conceptual simplicity and modular nature. The high level abstraction is useful for allowing us to interpret various PSO protocols through the lens of mqRPMT, and helps to greatly reduces the deployment and maintenance costs of PSO in the real world. We also presented two generic constructions of mqRPMT and instantiated them from the DDH assumption, yielding a family of PSO protocols with optimal asymptotic complexity and good concrete efficiency that are superior or competitive to existing ones. In summary, we regard the PSO framework from mqRPMT together with its efficient implementations as the main contribution of this work. We emphasize that our framework does not intend to fully cover the current state of the art, which is a rapidly moving target. Instead, it mainly aims to distill common principles and clean abstractions that can apply broadly and systematically.

Along the way of constructing mqRPMT, we introduced cwPRF and pOPRF. The notion of cwPRF can be viewed as the right cryptographic abstraction of the celebrated DH functions, demonstrating the versatility of the DDH assumption. The notion of pOPRF is of independent interest. It enriches the OPRF family, and helps us to understand which OPRF-based PSI protocols can (or cannot) be adapted to PCSI/PSU protocols. We left more applications and efficient constructions of pOPRF as an interesting problem.

Acknowledgement. We thank the anonymous reviewers for their valuable comments on this paper. We thank Yilei Chen for helpful discussions on the post-quantum constructions of cwPRF. This work was supported by the National Key Research and Development Program of China (Grant No. 2021YFA1000600), the National Natural Science Foundation of China (Grant No. 62272269 and No. 61932019), Taishan Scholar Program of Shandong Province, and Major Programs of the National Social Science Foundation of China (Grant No. 22&ZD147).

References

[AES03] Agrawal, R., Evfimievski, A.V., Srikant, R.: Information sharing across private databases. In: ACM SIGMOD 2003, pp. 86–97 (2003)

[ALSZ15] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_26

[Ber06] Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_14

[BKM+20] Buddhavarapu, P., Knox, A., Mohassel, P., Sengupta, S., Taubeneck, E., Vlaskin, V.: Private matching for computer (2020). https://eprint.iacr.org/2020/599

[Blo70] Burton, H.: Bloom. Commun. ACM **13**(7), 422–426 (1970)
[BLS01] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) Short signatures from the weil pairing. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_30
[BPSY23] Bienstock, A., Patel, S., Seo, J.Y., Yeo, K.: Near-optimal oblivious key-value stores for efficient PSI, PSU and volume-hiding multi-maps. In: USENIX Security 2023, pp. 301–318 (2023)
[CDGB22] Chen, Y., Ding, N., Dawu, G., Bian, Y.: Practical multi-party private set intersection cardinality and intersection-sum under arbitrary collusion. In: Deng, Y., Yung, M. (eds.) Information Security and Cryptology. Inscrypt 2022. LNCS, vol. 13837, pp. 169–191. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-26553-2_9
[CGT12] De Cristofaro, E., Gasti, P., Tsudik, G.: Fast and private computation of cardinality of set intersection and union. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 218–231. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35404-5_17
[CHLR18] Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: ACM CCS 2018, pp. 1223–1237 (2018)
[CLR17] Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: ACM CCS 2017, pp. 1243–1255 (2017)
[CM20] Chase, M., Miao, P.: Private set intersection in the internet setting from lightweight oblivious PRF. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 34–63. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_2
[CMdG+21] Cong, K., et al.: Labeled PSI from Homomorphic Encryption with Reduced Computation and Communication. In: ACM CCS 2021, pp. 1135–1150. ACM (2021)
[DC17] Davidson, A., Cid, C.: An efficient toolkit for computing private set operations. In: ACISP 2017 (2017)
[DCW13] Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: ACM CCS 2013, pp. 789–800 (2013)
[DRRT18] Demmler, D., Rindal, P., Rosulek, M., Trieu, N.: PIR-PSI: scaling private contact discovery. Proc. Priv. Enhanc. Technol. **2018**(4), 159–178 (2018)
[FIPR05] Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: TCC 2005, pp. 303–324 (2005)
[Fri07] Frikken, K.: Privacy-preserving set union. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 237–252. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_16
[GKRS22] Guo, S., Kamath, P., Rosen, A., Sotiraki, K.: Limits on the efficiency of (ring) LWE-based non-interactive key exchange. J. Cryptol. **35**, 1 (2022)
[GMR+21] Garimella, G., Mohassel, P., Rosulek, M., Sadeghian, S., Singh, J.: Private set operations from oblivious switching. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 591–617. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75248-4_21
[GPR+21] Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 395–425. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_14

[HEK12] Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS 2012 (2012)

[HFH99] Huberman, B.A., Franklin, M.K., Hogg, T.: Enhancing privacy and trust in electronic communities. In: ACM Conference on Electronic Commerce, pp. 78–86 (1999)

[HLS+16] Hogan, K., et al.: Secure multiparty computation for cooperative cyber risk assessment. In: IEEE Cybersecurity Development, SecDev 2016, pp. 75–76 (2016)

[HN10] Hazay, C., Nissim, K.: Efficient set operations in the presence of malicious adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_19

[IKN+20] Ion, M., et al.: On deploying secure computing: private intersection-sum-with-cardinality. In: IEEE EuroS&P 2020, pp. 370–389 (2020)

[IKNP03] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9

[JSZ+22] Jia, Y., Sun, S.-F., Zhou, H.-S., Du, J., Gu, D.: Shuffle-based private set union: faster and more secure. In: USENIX 2022 (2022)

[KK13] Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_4

[KKRT16] Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient batched oblivious PRF with applications to private set intersection. In: ACM CCS 2016, pp. 818–829 (2016)

[Kle21] Kleppmann, M.: Implementing curve25519/x25519: a tutorial on elliptic curve cryptography (2021). https://www.cl.cam.ac.uk/teaching/2122/Crypto/curve25519.pdf

[KLS+17] Kiss, Á., Liu, J., Schneider, T., Asokan, N., Pinkas, B.: Private set intersection for unequal set sizes with mobile applications. Proc. Priv. Enhanc. Technol. **4**, 177–197 (2017)

[KMP+17] Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., Trieu, N.: Practical multi-party private set intersection from symmetric-key techniques. In: ACM CCS 2017, pp. 1257–1272 (2017)

[KRTW19] Kolesnikov, V., Rosulek, M., Trieu, N., Wang, X.: Scalable private set union from symmetric-key techniques. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11922, pp. 636–666. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_23

[KS05] Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_15

[LG23] Liu, X., Gao, Y.: Scalable multi-party private set union from multi-query secret-shared private membership test. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology. ASIACRYPT 2023. LNCS, vol. 14438, pp. 237–271. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8721-4_8

[LV04] Lenstra, A., Voss, T.: Information security risk assessment, aggregation, and mitigation. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 391–401. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_34

[Mea86] Meadows, C.A.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: IEEE Symposium on Security and Privacy, pp. 134–137 (1986)

[MPR+20] Miao, P., Patel, S., Raykova, M., Seth, K., Yung, M.: Two-sided malicious security for private intersection-sum with cardinality. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 3–33. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_1

[NTL+11] Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D.: Location privacy via private proximity testing. In: NDSS 2011 (2011)

[NTY21] Nevo, O., Trieu, N., Yanai, A.: Simple, fast malicious multiparty private set intersection. In: ACM CCS 2021, pp. 1151–1165 (2021)

[Opea] https://github.com/openssl

[Opeb] https://www.openmp.org/resources/openmp-compilers-tools/

[Pri] https://github.com/google/private-join-and-compute/issues/16

[PRTY19] Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: SpOT-light: lightweight private set intersection from sparse OT extension. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 401–431. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_13

[PSTY19] Pinkas, B., Schneider, T., Tkachenko, O., Yanai, A.: Efficient circuit-based PSI with linear communication. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 122–153. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_5

[PSZ14] Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: USENIX 2014, pp. 797–812 (2014)

[PSZ18] Pinkas, B., Schneider, T., Zohner, M.: Scalable private set intersection based on OT extension. ACM Trans. Priv. Secur. **21**(2), 7:1-7:35 (2018)

[RA18] Resende, A.C.D., Aranha, D.F.: Faster unbalanced private set intersection. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 203–221. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_11

[Rab05] Rabin, M.O.: How to exchange secrets with oblivious transfer (2005). https://eprint.iacr.org/2005/187

[RR17] Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 235–259. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_9

[RR22] Raghuraman, S., Rindal, P.: Blazing fast PSI from improved OKVS and subfield VOLE. In: ACM CCS 2022 (2022)

[RS21] Rindal, P., Schoppmann, P.: VOLE-PSI: fast OPRF and circuit-PSI from vector-OLE. In: Canteaut, A., Standaert, F.-X. (eds.) EUROCRYPT 2021. LNCS, vol. 12697, pp. 901–930. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77886-6_31

[RT21] Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. In: ACM CCS 2021, pp. 1166–1181 (2021)

[Sha80] Shamir, A.: On the power of commutativity in cryptography. In: de Bakker, J., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 582–595. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10003-2_100

[SJ23] Son, Y., Jeong, J.: PSI with computation or circuit-psi for unbalanced sets from homomorphic encryption. In: ASIA CCS 2023, pp. 342–356. ACM (2023)

[TCLZ23] Tu, B., Chen, Y., Liu, Q., Zhang, C.: Fast unbalanced private set union from fully homomorphic encryption (2023)

[TKC07] Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.U.: Privacy preserving error resilient DNA searching through oblivious automata. In: ACM CCS 2007, pp. 519–528 (2007)

[WY23] Wu, M., Yuen, T.H.: Efficient unbalanced private set intersection cardinality and user-friendly privacy-preserving contact tracing. In: USENIX Security 2023 (2023)

[ZCL+23] Zhang, C., Chen, Y., Liu,W., Zhang, M., Lin, D.: Optimal private set union from multi-query reverse private membership test. In: USENIX 2023 (2023). https://eprint.iacr.org/2022/358