Efficient Differentially Private Set Intersection

Xinyu Peng*†, Yufei Wang‡(✉), Weiran Liu†, Liqiang Peng†, Feng Han†, Zhen Gu‡, Jianling Sun*, Yuan Hong§
*Zhejiang University, †Alibaba Group,
‡DAMO Academy, Alibaba Group, Hupan Lab, §University of Connecticut
*sunjl@zju.edu.cn, †{steven.pengxy, weiran.lwr, plq270998, fengdi.hf}@alibaba-inc.com,
‡{wangyufei.wyf, guzhen.gz}@alibaba-inc.com, §yuan.hong@uconn.edu

***Abstract*—Private Set Intersection (PSI) enables a sender and a receiver to jointly compute the intersection of their sets without disclosing other information about items not in the intersection. However, in many cases of joint data analysis, it is not just the items outside the intersection that are sensitive but the items within it. To protect such sensitive information, prior work presents a Differentially Private version of PSI (DPSI) based on a circuit-PSI using Fully Homomorphic Encryption. However, their concrete protocol is somewhat inefficient compared with the state-of-the-art (SOTA) circuit-PSI.**

In this paper, we revisit the DPSI definition and formalize its ideal functionality. We identify the key desiderata required by PSI-related tools to construct DPSI and propose two frameworks to construct efficient DPSI protocols. The first one generalizes the idea of existing DPSI, showing that any circuit-PSI can be used to construct DPSI. We obtain a more efficient DPSI protocol by plugging the SOTA circuit-PSI protocol in the framework. The second one helps to obtain a more efficient DPSI protocol based on the multi-query Reverse Private Membership Test (mqRPMT) that was previously used to construct Private Set Operation (PSO). However, mqRPMT additionally leaks the intersection size to the sender. We bound such leakage using differential privacy by padding random dummy items in input sets. We implement numerous constructions based on our frameworks. Experiments show that our protocols significantly outperform the existing DPSI construction, 2.5-22.6× more communication efficient and up to 110.5-151.8× faster. Our work also shows a new use case for mqRPMT besides obtaining PSO.

***Index Terms*—private set intersection, differential privacy**

I. Introduction

Private Set Intersection (PSI) allows two parties, a sender P_1 and a receiver P_2, holding sets X and Y, respectively, to identify the intersection $X \cap Y$ without revealing any information about items not in the intersection. PSI can help to achieve data minimization as no data is shared beyond what each party has in common [1]. As one of the best-studied applications of secure computation, PSI has been widely recognized as a highly valuable tool with numerous important applications in practice.

Although PSI preserves privacy for items outside the intersection, in many cases of joint data analysis, the intersection itself is also sensitive. Consider the following Example. A card-payment company P_1 wants to present advertisements to some individuals on an ads platform P_2. To this end, P_1 and P_2 run PSI to ensure that P_2 can only identify individuals who are common targets for ads between both entities. In this way, individuals outside the intersection are kept secret for P_2, preventing unnecessary information leakage. However, all

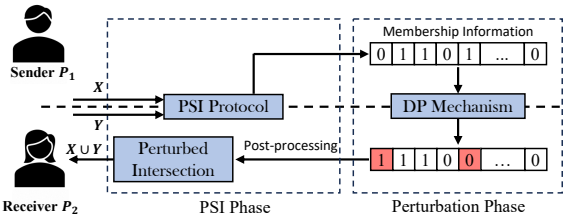


Fig. 1: The key desiderata of PSI to obtain DPSI

users in the intersection have an implicit feature that they are P_1's users. Therefore, P_2 can directly identify that each user in the intersection must also be registered with P_1. This can be seen as a data privacy problem when some users might be reluctant to disclose such information to the ads company. It can also be seen as a data secrecy problem because it allows the ads platform to help another rival card-payment company, e.g., P_1', present ads for customers in this intersection, winning customers and gaining advantage for P_1' directly from P_1. Unfortunately, since the PSI functionality requires outputting the correct intersection, this privacy problem inherently cannot be handled by PSI.

From the privacy perspective, one can think of PSI as the sender P_1 (holding the input set X) telling the receiver P_2 (holding the input set Y) a Boolean vector with size $|Y|$, where each bit indicates whether the corresponding item in Y is in the intersection, leaking one-bit information for each item in Y. In this way, the problem can be formalized as a privacy leakage in which Differential Privacy (DP) comes in. DP has been increasingly regarded as a *de facto* standard for protecting individual privacy that has been studied extensively in both academia [2]–[9] and industry [10]–[14]. DP guarantees that the presence or absence of any particular individual's record has a negligible impact on the likelihood that a particular result is returned to a query. Thus, an adversary cannot make meaningful inferences about any individual's record value or even whether the record is present.

The natural idea is to combine DP with PSI to further protect the Boolean information corresponding to each item. Targeting this problem, Kacsmar et al. [15] defined the notion of Differentially Private Set Intersection (DPSI) and proposed a DPSI protocol based on a variant of PSI, namely circuit-PSI [16]–[18], that uses Fully Homomorphic Encryption (FHE) [19], [20].

The solution from Kacsmar et al. [15] offers a promising

approach to this problem, while there is a critical limitation: although their protocol has linear computation complexity, the use of FHE results in poor performance in practice. The performance report in [15] shows that even for small item size $n = 2^{16}$, the running time reaches about 8 minutes, which is inefficient compared to typical PSI-based applications that require several minutes for database size about $n = 2^{20}$ [21], [22]. Circuit-PSI and related tools have been extensively studied and concretely more efficient, i.e., several minutes for item size $n = 2^{20}$ [17], [18], [23], the natural question is to close the gap between DPSI and PSI-related tools and utilize these tools to construct more efficient DPSI protocols. Since PSI is a very active research area, and more PSI-related tools have been proposed, a deeper understanding of the relationship between DPSI and PSI-related tools helps us construct DPSI using a broader range of (future) tools.

Our Contributions. In this paper, we revisit the existing definition of DPSI [15] and formalize its ideal functionality. By considering the privacy and security concerns in the ideal functionality, we figure out the key desiderata of PSI for obtaining its corresponding DPSI protocol. The details are illustrated in Fig. 1. Briefly speaking, for a sender who holds X and a receiver who holds Y, any PSI-related tool that enables the sender to perturb the membership information about Y *without knowing the correct intersection* can be used to construct DPSI. We identify two such PSI-related tools: circuit-PSI [16]–[18] and multi-query Reverse Private Membership Test (mqRPMT) [24], [25].

Following the key desiderata, we propose two general frameworks for constructing DPSI. The first one is circuit-DPSI, which is based on circuit-PSI. In this framework, the parties first get a shared Boolean vector of the intersection by *employing any circuit-PSI*. Then, the sender obliviously perturbs his shared Boolean vector via a DP mechanism. Finally, the receiver can get the DP intersection by XORing the sender's perturbed Boolean vector with its own one. We note that existing FHE-based DPSI [15] follows this framework. By plugging in more efficient circuit-PSI instances, we obtain more efficient DPSI compared with the protocol by Kacsmar et al. [15].

The second framework, which we call mqRPMT-DPSI, helps obtain even more efficient DPSI. As the name suggests, it leverages a recently proposed tool named multi-query Reverse Private Membership Test (mqRPMT). In mqRPMT, a sender holding a vector $X = (x_1, ..., x_n)$ interacts with a receiver holding a set Y, and eventually, the sender learns only a bit vector $(e_1, ..., e_n)$ indicating whether $x_i \in Y$ without learning the value of Y, while the receiver learns nothing. This tool is previously used to construct the Private Set Operation (PSO) [24], [25]. In our second framework, two parties first invoke mqRPMT with their input sets X and Y, and the sender gets the indication bit vector, which contains the membership information of Y. Then, the sender perturbs the indication bit vector via a DP mechanism and sends it to the receiver, who gets the final DPSI output. Thanks to the efficiency of the SOTA mqRPMT protocol [25], the second framework would be a plausible choice to obtain a more efficient DPSI.

However, there is a remaining challenge when leveraging mqRPMT to construct DPSI. According to the functionality of mqRPMT, the indication bit vector will unavoidably leak the cardinality of the intersection set to the sender, which may lead to the attacks [26], [27] that de-anonymize specific items in the input sets. We propose a solution to alleviate this privacy leakage. Briefly speaking, we ask both parties to insert dummy items into their sets via a Random Dummy Item Padding mechanism ahead of time and take the padded X' and Y' as mqRPMT inputs. Random Dummy Item Padding guarantees that the leaked cardinality satisfies (ϵ, δ)-DP. By plugging in the SOTA mqRMPT protocol [25], we finally obtain an even more efficient DPSI where the additional cardinality leakage is bounded by differential privacy. Interestingly, our work shows a new use case for mqRPMT besides obtaining PSO.

The main contributions are summarized as follows:

- We revisit the current DPSI definition and formalize its ideal functionality. We then figure out the key desiderata of PSI for obtaining its corresponding DPSI protocols.
- We propose a general DPSI framework satisfying ϵ-DPSI based on circuit-PSI. We then present an mqRMPT-based DPSI framework to further improve efficiency, which additionally reveals the cardinality of the intersection set with (ϵ, δ)-DP guarantee as a trade-off.
- We implement numerous constructions based on the proposed DPSI frameworks. Experiments show that they are arguably the most computation- and communication-efficient DPSI protocols available to date, and their efficiency is comparable to that of the most efficient PSI protocols.

Organizations. The remainder of this paper is organized as follows. Section II reviews the related works. Section III provides the preliminaries. Section IV revisits the definition of DPSI. Section V and Section VI give the details of the circuit-DPSI framework and the mqRPMT-DPSI framework, respectively. Section VII demonstrates our experimental results. Finally, Section VIII concludes our work.

II. Related Works

PSI. PSI dates back to the 1980s, and the first PSI protocol was initially based on the Diffie-Hellman (DH) key agreement [28]. Since then, constructions employing DH key agreement [29]–[31], Oblivious Transfers (OT), FHE [32] and Garbled Circuits (GC) [33] have been proposed. These proposals offer different trade-offs in terms of their computation costs and communication overheads. Among them, OT-based protocols are typically faster than other variants. Specifically, the first OT-based PSI protocol was proposed by Pinkas et al. [34], which employs OT extension [35]–[37] as the technical core of their protocol. Kolesnikov et al. [38] proposed the fastest OT-based PSI protocol by coming up with an efficient primitive called batched Oblivious Pseudo-Random Functions (batched

OPRF). However, all these OT-based PSI protocols offer improved efficiency at the expense of increased communication. To reduce the communication overheads, Pinkas et al. [23] proposed a PSI protocol based on sparse OT extension. Their scheme achieves a better balance between computation costs and communication overheads. Furthermore, Pinkas et al. [39] extended their work by introducing a new data structure called Oblivious Key-Value Stores (OKVS) and obtained more efficient PSI protocols under different security models. Shortly after, Rindal et al. [17] proposed an even more computationally efficient PSI protocol by combining the OKVS data structure with Vector Oblivious Linear Evaluation (VOLE) [40], [41]. Garimella et al. [42] introduced a way of compressing OKVS, improving the communication overhead of [39]. By employing the improved OKVS into [17], Raghuraman et al. [43] further reduced the communication complexity of PSI. Chen et al. [25] presented a unified framework by taking mqRPMT [24] as the primitive and using the framework to obtain PSI and more general PSO.

Circuit-PSI. PSI protocols mentioned above are specific solutions that only output the intersection itself. For supporting arbitrary secure computations over the intersection of private sets, Huang et al. [44] introduced the notion of circuit-PSI, where both parties receive shares of the set intersection instead of the intersection in the clear. The work introduced several Boolean circuits for PSI and evaluated these circuits using Yao's GC [33]. This type of PSI is then referred to as circuit-PSI. Targeting the same problem, Chen et al. [45], [46] constructed circuit-PSI protocols that achieve low communication overheads by employing FHE [32]. However, since FHE introduces a significant computational overhead, these circuit-PSI protocols are inefficient in practice. By invoking the Oblivious Programmable Pseudo-Random Function (OPPRF), Pinkas et al. [16] proposed the first circuit-PSI protocol that achieves linear communication complexity without FHE. Chandran et al. [18] constructed a new circuit-PSI protocol with linear communication and computation costs by invoking Relaxed Batch Oblivious Programmable Pseudo-Random Functions (RB-OPPRF).

Composing DP and PSO. There are a series of works that protect sensitive data in private sets by composing DP and PSO. Numerous works employ DP as a complementary technology for preventing information leakage during the use of secure computation protocols. In order to prevent side-channel attacks while answering federated queries, Narayan et al. [47] proposed a primitive for obtaining PSI cardinality (PSI-CA) in a differentially private manner. Addressing the same problem, Bater et al. [48] introduced a private data federation called Shrinkwrap, which utilizes computational DP to minimize padding in intermediate query results. DP can also be adopted as an orthogonal technology to protect the query results, which cannot be protected by secure computation. Along those lines, He et al. [49] formulated the problem of Differentially Private Record Linkage (DPRL) and developed corresponding protocols by employing a specific privacy definition called Output Constrained DP. In 2020, for the first time, Kacsmar et al. [15] attempted to construct DPSI by combining FHE-based PSI with DP. However, the use of FHE results in somewhat inefficient protocol in practice. Besides these work, Groce et al. [50] consider the problem of improving the performance of secure 2-party computation protocols by leveraging DP. They show that if differentially private leakage is allowed, the cost of Rindal et al.s' [51] PSI protocol can be reduced by up to 63%.

III. Preliminaries

A. Notations and Security Model

The computational security parameter and the statistical security parameter are denoted by κ and σ, respectively. We denote $[a, b]$ with $a, b \in \mathbb{N}$ and $a \leq b$ to the set $\{a, a+1, \ldots, b\}$ and $[b]$ as a shorthand for $[1, b]$. We denote the upper letter $A = (a_1, \ldots, a_n)$ as a vector with length $|A| = n$.

Two parties in DPSI are P_1 (the sender) and P_2 (the receiver). They have input sets $X = \{x_i\}_{i\in[n]} \in \mathcal{V}^n$ and $Y = \{y_j\}_{j\in[m]} \in \mathcal{V}^m$, respectively. All items in X and Y are from the same item domain $\mathcal{V}$. Based on the context, we sometimes use the notation $X[i]$ or x_i to denote the i-th item in the set X. We assume that both parties wish to securely compute a differentially private version of $X \cap Y$ without disclosing data outside the intersection to each other.

In this work, we consider the semi-honest model, where an adversary tries to learn as much information as possible from a given protocol execution but is not able to deviate from the protocol procedures. We remark that all cryptographic tools used in this work have been proven secure in the semi-honest model under the standard real-ideal paradigm [52].

B. Standard Differential Privacy Definition

DP is an appealing choice to allow multiple releases of statistics while bounding the information leakage of the individual records in the output. DP requires the outputs of algorithms to be approximately the same if any individual's record in the input is added or removed. Formally, DP is defined as follows [53].

Definition 1 ((ϵ, δ)-Differential Privacy): A randomization mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{F}$ is (ϵ, δ)-differential privacy if for all neighboring inputs $D, D' \in \mathcal{D}$ such that D and D' differ by adding or removing a record, and all subsets $F \in \mathcal{F}$,

$$\Pr[\mathcal{M}(D) \in F] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in F] + \delta$$

where the probability is taken over the randomness of $\mathcal{M}$.

In Definition 1, ϵ is the privacy budget, and δ can be treated as the failure probability of DP protection. We say that $\mathcal{M}$ satisfies ϵ-DP if $\delta = 0$. When obtaining multiple DP outputs, the sequential composition (Theorem 1) and parallel composition (Theorem 2) are used to measure the total privacy budget.

Theorem 1 (Sequential Composition): Given k number of (ϵ_i, δ_i)-DP mechanisms $\mathcal{M}_i$ that access the same input D with $1 \leq i \leq k$, the combination of their outputs satisfies (ϵ, δ)-DP, where $\epsilon = \sum_{i=1}^{k} \epsilon_i$ and $\delta = \sum_{i=1}^{k} \delta_i$.

Theorem 2 (Parallel Composition): Given k number of (ϵ_i, δ_i)-DP mechanisms $\mathcal{M}_i$ that access disjoint inputs D_i with $1 \leq i \leq k$, the combination of their outputs satisfies (ϵ, δ)-DP, where $\epsilon = \max(\epsilon_1, \ldots, \epsilon_k)$ and $\delta = \max(\delta_1, \ldots, \delta_k)$.

In this work, we follow the standard DP definition to consider DPSI by defining neighboring datasets and measuring the ratio of output distribution by taking all possible pairs of the neighboring datasets as inputs. The details are shown in Section IV.

C. Cryptographic Primitives

Cuckoo Hashing. Cuckoo hashing [54] uses $d > 1$ universal hash function $h_1, \ldots, h_d : \{0,1\}^* \to [\beta]$ to map n items to $\beta > n$ bins in the hash table HT, where each bin is restricted to accommodate at most one item. It iteratively inserts a sequence of items $(e_1, \ldots, e_n)$ into HT as follows. Given the item e_i, if one of $HT[h_1(e_i)], ..., HT[h_d(e_i)]$ bins is empty, then insert e_i in the first empty bin. Otherwise, sample $i \in [d]$ uniformly at random, evict the item e_i' present in $HT[h_i(e_i)]$, place e_i in bin $HT[h_i(e_i)]$, and recursively try to insert the evicted item e_i' until the number of attempts reaches a certain threshold and then fails. In this way, the item e_i could be placed into one bin among $h_1(e_i), \ldots, h_d(e_i)$.

Oblivious Programmable Pseudo-Random Function. Let $F : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\ell$ be a Pseudo-Random Function (PRF) that maps an input to a pseudo-random output under a PRF key. Programmable PRF (PPRF) is a variant of PRF that further allows to "program" some PRF outputs, i.e., produce assigned PRF outputs for some specific given inputs.

PPRF is usually defined for handling multiple PRF keys $K = (k_1, \ldots k_\beta)$ in a batch. An (ℓ, β)-PPRF consists of a pair of algorithms $\hat{F} = (Hint, F)$. Given a set of uniformly random and independent PRF keys $K = (k_1, \ldots, k_\beta)$, the disjoint input sets $(X_1, \ldots, X_\beta)$, and the target multi-sets $(T_1, \ldots, T_\beta)$ satisfying $|T_j| = |X_j|$ for all $j \in [\beta]$ and $T_j[i] \in \{0,1\}^\ell$ for all $i \in [|T_j|]$, $Hint(K, X, T)$ outputs $hint$. Later, given the key $k_j \in \{0,1\}^\kappa$, $hint$, and an input $x \in \{0,1\}^*$, $F(k_j, hint, x)$ outputs a PRF value $y \in \{0,1\}^\ell$. It holds that if $x = X_j[i]$ for some $i \in [|X_i|]$, then $y = T_j[i]$. Otherwise, y looks random.

Oblivious PPRF (OPPRF) is a two-party protocol that allows the sender and the receiver to obliviously run PPRF. In OPPRF, the sender takes inputs as the programmed input sets $X = (X_1, \ldots, X_\beta)$ and the target sets $T = (T_1, \ldots, T_\beta)$, while the receiver takes inputs as the batch queries $(q_1, \ldots, q_\beta)$. OPPRF samples PPRF keys $K = (k_1, \ldots, k_\beta)$ for an (ℓ, β)-PPRF to the sender, and gives $hint$ and the PPRF outputs $\{F(k_j, hint, q_j)\}_{j \in [\beta]}$ to the receiver.

Private Set Membership. Private Set Membership (PSM) [18] is a two-party protocol that takes input as a set $X = \{x_i\}_{i \in [n]} \in \mathcal{V}^n$ from the sender and an item $y \in \mathcal{V}$ from the receiver. Define the Boolean value a such that $a = 1$ if $y \in X$ and $a = 0$ otherwise. The PSM functionality outputs Boolean shares of a to the sender and the receiver, i.e., random bits $\langle a \rangle_1$ and $\langle a \rangle_2$ to both parties, respectively such that $\langle a \rangle_1 \oplus \langle a \rangle_2 = a$.

Parameters: The size of sender's set n. The size of the receiver's set m. $H_s : \mathcal{V}^m \to HT_2$, which on input a size-$m$ set outputs a hash table with size $\beta = O(m)$ and each element is assigned to a distinct bin of HT_2.
Input: The sender P_1's set of items $X = \{x_i\}_{i \in [n]}$, and the receiver P_2's set of items $Y = \{y_i\}_{i \in [m]}$.
Output: The sender P_1 gets a shared Boolean vector $\{\langle a_j \rangle_1\}_{j \in [\beta]}$. The receiver P_2 gets a hash table $HT_2 = H_s(Y)$ with size $\beta = O(m)$ storing items in Y, and a shared Boolean vector $\{\langle a_j \rangle_2\}_{j \in [\beta]}$, where $a_j = \langle a_j \rangle_1 \oplus \langle a_j \rangle_2$ indicates if the item in $HT_2[j]$ is in the intersection.

Fig. 2: Ideal functionality $\mathcal{F}_{CPSI}$ for circuit-PSI

Circuit-PSI. Circuit-PSI is a two-party protocol that allows the sender and receiver to securely compute arbitrary symmetric functions over the intersection of their private sets by utilizing the secure circuit evaluation. In circuit-PSI, both the sender and the receiver take their sets $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_m\}$ as inputs and receive the Boolean shares of the set intersection $Z = X \cap Y$. The functionality of circuit-PSI is formally defined in Fig. 2.

The first circuit-PSI protocol that achieves linear communication complexity is presented by Pinkas et al. [16] by utilizing OPPRF. In their protocol, the receiver P_2 runs Cuckoo hashing with d hash functions that map Y into the hash table HT_2, while the sender P_1 simply hashes items in X using all d hash functions into the hash table HT_1. Then, both parties invoke $\mathcal{F}_{OPPRF}$ functionality to map their items to PPRF values, where all items in HT_1 are mapped to values in the target set $T = \{T_1, \ldots, T_\beta\}$ for $T_j = \{t_j\}$ randomly sampled by P_1. After running $\mathcal{F}_{OPPRF}$, P_2 receives the PPRF outputs $(y_1^*, \ldots, y_\beta^*)$. Finally, P_1 and P_2 invoke the $\mathcal{F}_{PSM}$ functionality to check whether y_j^* lies in $T_j = \{t_j\}$ and learn Boolean shares of membership, respectively. Since there is only one t_j in each T_j, $\mathcal{F}_{PSM}$ can be done very efficiently with only linear communication complexity, so does the resulting circuit-PSI. The protocol proposed by Pinkas et al. [16] can be viewed as a prototype for constructing linear circuit-PSI, and all other efficient circuit-PSI protocols are developed by replacing the OPPRF with alternative primitives, e.g., the OKVS data structure [17], [39] and RB-OPPRF [18].

Multi-query Reverse Private Membership Test. Multi-query Reverse Private Membership Test (mqRPMT) [24], [25] is a two-party protocol that is initially used as a primitive for constructing PSO. In mqRPMT, a sender with a set X interacts with a receiver holding a set $Y = (y_1, \ldots, y_m)$, and eventually the sender learns only a bit vector $(e_1, \ldots, e_m)$ indicating whether $y_i \in X$ without learning the value of y_i, while the receiver learns nothing[1] The functionality of mqRPMT is formally defined in Fig. 3. Among the protocols that instantiate mqRPMT, the cwPRF-based mqRPMT protocol is the most efficient one that achieves strict lin-

[1]For ease of description, we define mqRPMT in the reversed roles.

Parameters: The size of sender's set n. The size of receiver's set m.
Input: The sender P_1's set of items $X = \{x_i\}_{i\in[n]}$, and the receiver P_2's set of items $Y = \{y_i\}_{i\in[m]}$.
Output: The sender P_1 gets a vector $(e_1, \ldots, e_m)$, where $e_i = 1$ if $y_i \in X$ and $e_i = 0$ otherwise. The receiver P_2 gets nothing.

Fig. 3: Ideal functionality $\mathcal{F}_{mqRPMT}$ for mqRPMT

ear computation and communication complexity. Specifically, cwPRF is a family of keyed functions $K \times D \to D$ that satisfy weak pseudom-randomness and commutative property simultaneously. Specifically, commutative property means that for all $k_1, k_2 \in K$ and $x \in D$, a family of keyed functions satisfy $F_{k_1}(F_{k_2}(x)) = F_{k_2}(F_{k_1}(x))$.

One can easily obtain cwPRF using a finite group $\mathbb{G}$ with order p where the Decisional Diffie-Hellman (DDH) assumption holds, by setting $k_1, k_2 \in \mathbb{Z}_p$, selecting a hash function $H : \{0,1\}^* \to \mathbb{G}$, and defining $F_{k_1}(x) = H(x)^{k_1}, F_{k_2}(x) = H(x)^{k_2}$. The finite group under the Elliptic Curve (EC) is a good candidate choice for $\mathbb{G}$.

IV. Differentially Private Set Intersection

In this section, we first revisit the DPSI definition and formulate the ideal functionality of DPSI. Based on the privacy and security concerns in the ideal functionality, we identify the key desiderata for PSI-related tools that can be used to construct DPSI protocol.

Before going into the DPSI definition, we first consider the difference between DPSI and classical DP. Classical DP is in the centralized setting where the data owner has the data in the clear and answers queries from the aggregated data with DP guarantee by adding noise on their own. Therefore, classical DP usually considers the mechanism outputs an aggregation result. However, PSI computes the intersection of two sets, which is not an aggregation procedure. Therefore, in PSI setting, we may need to consider DPSI mechanisms as a way of randomizing the presence or absence of any possible items in the input sets. Such randomization procedure lies more similar to the Local DP (LDP) setting [55] by treating each item in the inputs as an individual's data and adding noise (randomization) on items by individuals themselves. However, we still consider DPSI in the DP setting because the data owner (i.e., the sender and the receiver) have the data *in the clear* and add noise in the centralized setting.

A. Security Gap in the Existing DPSI

We first revisit the DPSI definition proposed by Kacsmar et al. [15] by following the classical DP definition (shown in Definition 1).

The key point to define DPSI is to precisely define the neighboring datasets and the output range. There are two inputs in PSI: the set X from the sender and the set Y from the receiver. As the DP definition suggests, one needs to consider the output distribution for all neighboring datasets D' and D that differ in any single item. Kacsmar et al. [15] consider a one-sided PSI where the sender learns nothing about the receiver's input set Y, and only the receiver learns a differentially private version of the intersection. For defining the neighboring datasets corresponding to one-sided DPSI, we can consider the one-sided PSI functionality from a different perspective. Think of the sender's input X being the database to be queried, and the receiver asking a query with the input data Y. That is, the query is of the form:

Which items in your database are also in Y?

On one hand, this can be analogous to the standard counting query "How many people in the database with age greater than 20?" with the input data 20. Note that DP contemplates a static query structure, wherein altering the input from 20 to a different value constitutes a modification of the query itself, thereby contravening the foundational prerequisites of DP. Similarly, by treating the input data Y as a query description rather than the data, it is reasonable to consider DPSI with the fixed query that contains Y as parts of the query description. On the other hand, since the PSI functionality only allows the receiver to get the intersection $X \cap Y$, this means that PSI only reveals X's information to the receiver. Therefore, we only need to consider privacy from the sender's input X by defining the neighboring dataset $D' = (X', Y)$ for a fixed Y, where $X' = X \cup \{v_t\}$ for $v_t \notin X$ or $X' = X/\{v_r\}$ for $v_r \in X$.

Although the neighboring dataset can be any dataset by adding or removing one item in X, the output intersection Z must be a subset of the fixed Y. Therefore, the output range can be defined as a set description with the capability of including all items in Y. Kacsmar et al. [15] chooses the output range in the most compressed form, that is, a Boolean vector $\mathcal{F} = \{0,1\}^m$ in which the Boolean value f_i indicates whether $y_i \in Z$. Combining it with the definition of the neighboring dataset leads to the DPSI definition described as follows.

Definition 2 (ϵ-Differentially Private Set Intersection [15]): The randomized two-party mechanism $\mathcal{M} : \mathcal{V}^n \times \mathcal{V}^m \to \{0,1\}^m$ that takes sender's set X and receiver's set Y as inputs and returns a Boolean vector $F \in \{0,1\}^m$ to the receiver achieves ϵ-DPSI if and only if for all $F \in \{0,1\}^m$ and all neighboring datasets $D = (X, Y)$ and $D' = (X', Y)$ where $X' = X \cup \{v_t\}$ or $X' = X/\{v_r\}$, $v_t, v_r \in \mathcal{V}$,

$$\Pr[\mathcal{M}(D) = F] \leq e^{\epsilon} \Pr[\mathcal{M}(D') = F]$$

where the probability is taken over the randomness of $\mathcal{M}$.

The above definition clarifies the privacy concerns of DPSI formally. However, since a DPSI protocol inherently relates to both data security and data privacy, a definition from the security aspect is also required. We formulate the ideal functionality $\mathcal{F}_{DPSI}$ in Fig. 4, which takes X from P_1 and Y from P_2 as inputs and only outputs a perturbed Boolean vector to P_2. It is parameterized by a differential private mechanism $\mathcal{M}_{dp}$ constraint with the hamming distance $HM(F, F') = 1$ between neighboring dataset F and F'. The output of $\mathcal{F}_{DPSI}$

Parameters: The size of the sender's set n. The size of the receiver's set m. The privacy budget ϵ. Differential private mechanism $\mathcal{M}_{dp}$ satisfying $\Pr[\mathcal{M}_{dp}(F) = O] \leq e^{\epsilon} \Pr[\mathcal{M}_{dp}(F') = O]$ for all $F, F', O \in \{0,1\}^m$ with $HM(F, F') = 1$.
Input: The sender P_1's set of items $X = \{x_i\}_{i\in[n]} \in \mathcal{V}^n$, and the receiver P_2's set of items $Y = \{y_i\}_{i\in[m]} \in \mathcal{V}^m$.
Functionality: Upon receiving X from P_1, Y from P_2. Compute $F' \in \{0,1\}^m$, where f'_i indicates if $y_i \in X$. Apply $\mathcal{M}_{dp}$ to F' and send the result F to P_2.

Fig. 4: Ideal functionality $\mathcal{F}_{DPSI}$.

clearly achieves ϵ-DPSI. Further, a protocol provides ϵ-SIM-CDPSI (simulation-based computational differentially private PSI) if it securely realizes $\mathcal{F}_{DPSI}$ parameterized by a certain DP mechanism $\mathcal{M}_{dp}$ against a semi-honest non-uniform probabilistic polynomial time adversary under the simulation-base security paradigm. We refer to [15] for the definition of ϵ-SIM-CDPSI and omit it here. Note that this form of ideal functionality is not the only one that satisfies ϵ-DPSI. For example, a function that randomly samples $F \in \{0,1\}^m$ and outputs F to the receiver is also ϵ-DPSI. However, the above function is meaningless since the output F' is independent of the intersection. Therefore, we present $\mathcal{F}_{DPSI}$ by elucidating the relation between the result vector and the intersection.

B. PSI-related Tools for Constructing DPSI

After revisiting the definition of DPSI and formally providing the ideal functionality, we explore the scope of PSI-related tools that can be modified to the DPSI protocol. According to $\mathcal{F}_{DPSI}$, we can easily draw the conclusion that any PSI primitives that can be combined with $\mathcal{M}_{dp}$ to perturb the output before sending it to the receiver can be used for construction DPSI protocols. Although the conclusion is intuitive, finding suitable PSI tools is not easy. Since $\mathcal{F}_{DPSI}$ indicates that, in the absence of a trusted third party, the perturbation can only occur on the sender's side, yet the sender, paradoxically, cannot directly obtain the output.

Recall that the output range in DPSI definition (Definition 2) is $\mathcal{F} = \{0,1\}^m$. The output range indicates that any item in Y has the possibility of appearing in the output intersection Z regardless of whether the same item is contained in X or not. To this end, the randomization mechanism $\mathcal{M}$ for DPSI must be able to perturb (its unknown) items in Y/X to items in the output intersection Z.

Consequently, we identify the key desiderata required by PSI-related tools to construct a DPSI protocol: the sender must get the membership information of Y and can map the items in his/her set to items in Y via a randomized algorithm. To our best knowledge, two PSI-related tools satisfy this requirement: circuit-PSI [16]–[18] and mqRPMT [24]. In circuit-PSI, both parties get a shared Boolean vector of length $\beta = O(|Y|)$ obliviously indicating whether or not an item is in the intersection. Note that Kacsmar et al.'s DPSI

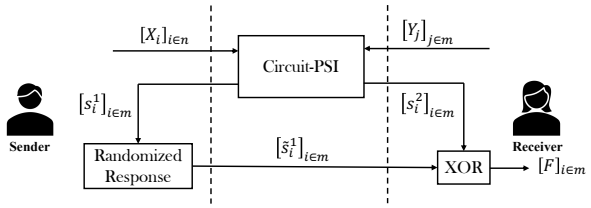


Fig. 5: The framework of circuit-DPSI.

protocol also takes a circuit-PSI as primitive which supports arbitrary subsequent circuits. In their protocol, the sender acquires the membership information of Y in the form of FHE's ciphertext. In mqRPMT, the sender gets a bit vector $(e_1, \ldots, e_n)$ indicating whether $x_i \in Y$ or not without learning the value of x_i. These two PSI-related tools create a potential for the sender to randomly perturb the final binary output using a DP mechanism. The remaining procedure is straightforward. We can leverage $\mathcal{M}_{RR}$ [56], which is a basic DP mechanism for binary data, to do the perturbation. Take a binary input $v \in \{0,1\}$, $\mathcal{M}_{RR}$ outputs $\tilde{v} = v$ with probability p and $\tilde{v} = 1 - v$ with probability q. Then, $\mathcal{M}_{RR}$ is formally defined as below.

$$\Pr[\tilde{v} = 1] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+1} & \text{if } v = 1 \\ q = \frac{1}{e^{\epsilon}+1} & \text{if } v = 0 \end{cases}. \tag{1}$$

Based on these eligible tools, we present two generic frameworks for constructing DPSI. As indicated by their respective designations, the first framework operates on the circuit-PSI, while the second framework is founded on the mqRPMT. Although both can achieve linear computation and communication complexity, they differ in terms of efficiency and security. The circuit-DPSI framework can be viewed as a generalization of Kacsmar et al.'s FHE-based protocol. In this framework, the sender learns nothing from the additive shares, but can perturb the output indirectly by flipping the bits in these shares, which rigorously satisfies $\mathcal{F}_{DPSI}$. In contrast, the mqRPMT-DPSI framework offers improved efficiency at the price of additionally leaking the intersection cardinality. To make the leakage acceptable without completely undermining the idea of computation, we further introduce a method to bound such leakage with (ϵ, δ)-DP.

C. The Practical Significance and Limitations

As a combination of DP and PSI, our DPSI framework ensures that inferences about the presence or absence of any specific item in the output intersection are limited, while the intersection as a whole remains sufficiently accurate for substantive usage. The utility-privacy trade-off for employing our frameworks manifests in the form of false positives and negatives within the intersection. Nonetheless, such a trade-off can be tailored to suit the specific requirements of the relevant application, as illustrated in the subsequent case.

Recall that the case of a card-payment company and an ads platform identify the set of customers that they have in common. By using the constructions of our DPSI frameworks, these two parties can agree on a differentially private set

Parameters: The size of parties' set n. A randomized response mechanism $\mathcal{M}_{RR}$.
Input: The sender P_1's set of items $X = \{x_i\}_{i\in[n]}$, and the receiver P_2's set of items $Y = \{y_i\}_{i\in[n]}$.
Protocol:

1. **(Circuit-PSI preprocessing)** The sender P_1 and the receiver P_2 invoke a circuit-PSI protocol with inputs X and Y and get the secret sharing $\langle a_j\rangle_1, \langle a_j\rangle_2 \in \{0,1\}$ as output, respectively.
2. **(Secret Sharing Perturbation)** P_1 invokes $\mathcal{M}_{RR}$ with input $\langle a_1\rangle_1, ..., \langle a_\beta\rangle_1$ and gets the perturbed secret sharing $\langle \tilde{a_1}\rangle_1, ..., \langle \tilde{a_\beta}\rangle_1$ as output.
3. **(Intersection Computation)** P_2 receives the perturbed secret sharing $\langle \tilde{a_1}\rangle_1, ..., \langle \tilde{a_\beta}\rangle_1$ from P_1 and computes $\mathcal{O} = \{\langle \tilde{a_j}\rangle_1 \oplus \langle a_j\rangle_2\}_{j\in[\beta]}$.
4. P_2 gets the intersection vector $F \in \{0,1\}^m$ by deleting the bits that correspond to the random dummy value from $\mathcal{O}$.

Fig. 6: Efficient circuit-DPSI protocol.

intersection that protects any item in the set from the card-payment company while preserving the effectiveness of the ads. Depending on their privacy preference, the card-payment company may choose a different utility-privacy trade-off. If the card payment company is privacy-sensitive, it may choose to minimize potential inferences about individual items by accepting a higher rate of false negatives and false positives in the intersection. If the card-payment company prioritizes the effectiveness of the ads, they may ask for a lower rate of false negatives and false positives in the intersection.

Meanwhile, it is important to note that if the same DPSI protocol is used repeatedly to compute the intersection of the same two sets, the probability of item leakage in the intersection will increase. Such leakage is actually the DPSI functionality: each time the DPSI protocol is used to compute an intersection, it will consume a portion of the privacy budget, denoted as ϵ. According to the parallel composition described in Theorem 2, the combination of all computed intersections satisfies $\sum_{i=1}^{n} \epsilon$-DPSI, where n represents the number of times the protocol is reused.

V. Circuit-DPSI Framework

A. Circuit-DPSI

We first show the flow of the circuit-DPSI in Fig. 5 and give the details in Fig. 6. In the circuit-DPSI, the sender P_1 and receiver P_2 first invoke a construction of circuit-PSI with inputs X and Y. Then, each of P_1 and P_2 gets a piece of Boolean shares $\{\langle a_j\rangle_i\}_{j\in[\beta]} \in \{0,1\}$, $i \in \{1,2\}$, as the output. The Boolean shares have $\beta = (1+\alpha)m$ bits, which contain not only part of information about the membership of the items in Y but also that of some random dummy values added during the Cuckoo Hashing of circuit-PSI protocol. After getting $\{\langle a_j\rangle_1\}_{j\in[\beta]}$, P_1 perturbs it bit by bit via the RR mechanism and sends the perturbed Boolean shares $\{\langle \tilde{a}_j\rangle_1\}_{j\in[\beta]}$ to P_2. Finally, P_2 receives $\{\langle \tilde{a}_j\rangle_1\}_{j\in[\beta]}$ and computes a Boolean vector $\mathcal{O} = \{\langle \tilde{a_j}\rangle_1 \oplus \langle a_j\rangle_2\}_{j\in[\beta]}$, which indicates the membership of the items in Y and the random dummy values. By deleting the bits that correspond to the random dummy value from $\mathcal{O}$, P_2 obtains $F = \{0,1\}^m$, a set where each position i contains a one if the item corresponding to Y_i is part of the differentially private intersection, and zeros in all other positions.

It is worth noting that within the circuit-DPSI framework, the integration of the circuit-PSI protocol is designed to be modular. By selecting and incorporating an appropriate circuit-PSI protocol, it is possible to achieve a more efficient circuit-based DPSI protocol, adaptable to varying network and computational environments.

We observed that the FHE-based DPSI protocol proposed by Kacsmar et al. [15] can be regarded as a variant of the circuit-DPSI construction. In this protocol, the receiver encrypts their items using the BGV [32] scheme and sends the ciphertext to the sender. After receiving the ciphertext from the receiver, the sender also encrypts their items and computes encryption of a vector containing ones in positions where the corresponding items are found in the intersection. Moreover, the sender perturbs the encrypted vector using the RR mechanism and sends the results to the receiver. Finally, the receiver decrypts the vector and obtains the perturbed intersection. It is evident that the workflow of this FHE-based DPSI protocol is similar to our circuit-DPSI framework, as both involve obtaining a vector containing membership information and perturbing it using the RR mechanism.

B. Security & Privacy Analysis of Circuit-DPSI

Theorem 3: $\mathcal{P}_{CDPSI}$ in Fig. 6 satisfies ϵ-SIM-DPSI.

We prove the above theorem in two steps. First, we present the ideal functionality of our Circuit-DPSI protocol and prove it correctly realizes ϵ-DPSI. Then, we prove our protocol securely realizes the functionality. By combining them, the theorem is proved.

Lemma 1: $\mathcal{F}_{DPSI}$ in Fig. 4 with $\mathcal{M}_{RR}$ parameterized by ϵ satisfies ϵ-DPSI.

Proof: Given the inputs X from the sender and Y from the receiver, a neighboring input $X' = X \cup \{v_t\}$ or $X' = X/\{v_r\}$, for any output $F \in \{0,1\}^m$ and the privacy budget ϵ, we need to prove that $\left|\frac{\Pr[\mathcal{F}_{CDPSI}(X,Y)=F]}{\Pr[\mathcal{F}_{CDPSI}(X',Y)=F]}\right| \leq e^\epsilon$.

We prove it by considering four different cases: (1) $X' = X \cup \{v_t\}$, $v_t \notin Y$; (2) $X' = X \cup \{v_t\}$, $v_t \in Y$; (3) $X' = X/\{v_r\}$, $v_r \notin Y$; (4) $X' = X/\{v_r\}$, $v_r \in Y$.

For the first case, when $v_t \notin Y$, v_t must not be in either $X \cap Y$ or $X' \cap Y$. As a result, all other items in the input are the same, we have $\Pr[\mathcal{F}_{CDPSI}(X,Y) = F]/Pr[\mathcal{F}_{CDPSI}(X',Y) = F] = 1 \leq e^\epsilon$.

For the third case, when $v_r \notin Y$, v_r must not be in either $X \cap Y$ or $X' \cap Y$. As a result, all other items in the input are the same, we have $\Pr[\mathcal{F}_{CDPSI}(X,Y) = F]/\Pr[\mathcal{F}_{CDPSI}(X',Y) = F] = 1 \leq e^\epsilon$.

For the second and fourth cases, since $v_t \in Y$, W.L.O.G., we assume that the t-th bit b_t in F indicates v_t is in the

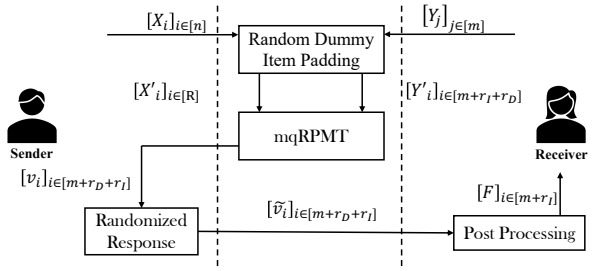


Fig. 7: The framework of mqRPMT-DPSI

intersection if $b_t = 1$, and v_t is not in the intersection if $b_t = 0$. By representing the t-th bit in sender's Boolean share for input X and X' as $\langle \tilde{a}_t \rangle_1^X$ and $\langle \tilde{a}_t \rangle_1^{X'}$, respectively, we have

$$\begin{aligned} &\left| \frac{\Pr[\mathcal{F}_{CDPSI}(X,Y) = F]}{\Pr[\mathcal{F}_{CDPSI}(X',Y) = F]} \right| \\ &= \prod_{i\in[m]} \left| \frac{\Pr[\mathcal{F}^i_{CDPSI}(X,Y) = b_i]}{\Pr[\mathcal{F}^i_{CDPSI}(X',Y) = b_i]} \right| \\ &= \left| \frac{\Pr[\mathcal{F}^t_{CDPSI}(X,Y) = b_t]}{\Pr[\mathcal{F}^t_{CDPSI}(X',Y) = b_t]} \right| \\ &= \left| \frac{\Pr[\mathcal{M}_{RR}(\langle \tilde{a}_t \rangle_1^X) = b_t]}{\Pr[\mathcal{M}_{RR}(\langle \tilde{a}_t \rangle_1^{X'}) = b_t]} \right| \leq \frac{p}{q} = e^{\epsilon}. \end{aligned} \tag{2}$$

Therefore, $\mathcal{F}_{CDPSI}$ with $\mathcal{M}_{RR}$ parametered by ϵ satisfies ϵ-DPSI. ∎

Lemma 2: The protocol $\mathcal{P}_{CDPSI}$ in Fig. 6 securely realizes the functionality in Fig. 4 using $\mathcal{M}_{RR}$ against a semi-honest adversary in the $\mathcal{F}_{CPSI}$-hybrid model.

Proof: The correctness of our protocol is obvious, which follows the correctness of the underlying protocol realizing $\mathcal{F}_{CPSI}$. Then we prove the security by constructing simulators. For the corrupt semi-honest receiver, we exhibit the simulator Sim_R as follows.

1) Sim_R receives P_2's input Y, invokes $\mathcal{F}_{DPSI}$ to obtain the output F', and appends them to the view.
2) Sim_R uses H_s and Y to honestly compute HT_2 with size β, and selects random bits $\langle a'_j \rangle_2 \leftarrow \{0,1\}$ for $j \in [\beta]$. Then, Sim_R invokes the simulator of $\mathcal{F}_{CPSI}$ with input $(Y, \{\langle a'_j \rangle_2\}_{j\in[\beta]})$ and appends the outputs to the view.
3) Sim_R computes a length-β bit vector b. For $i \in [\beta]$: If there is an element in the i-th bin in HT_2, say y_j, then $b_i = \langle a'_i \rangle_2 \oplus F'_j$. Otherwise, b_i equals the output of $\mathcal{M}_{RR}$ with input $\langle a'_i \rangle_2$.

The real view of P_2 in an execution can be written as: $\texttt{view}_2^{\pi} = \{Y, HT_2, \{\langle a_i \rangle_2\}_{i\in[\beta]}, \{\langle \tilde{a}_i \rangle_1\}_{i\in[\beta]}, F\}$, and the output view of Sim_R is: $S_2 = \{Y, HT_2, \{\langle a'_i \rangle_2\}_{i\in[\beta]}, \{b_i\}_{i\in[\beta]}, F'\}$. Two views are statically indistinguishable, which directly follows the security of underlying protocol realizing $\mathcal{F}_{CPSI}$.

Also, the simulator of the corrupt receiver can be easily constructed by simulating β random bits and invoking the simulator of $\mathcal{F}_{CPSI}$. Thus, we conclude the proof. ∎

VI. MQRPMT-BASED DPSI FRAMEWORK

By removing the dependency on FHE, the circuit-DPSI framework is much more efficient compared to the existing construction of DPSI. However, the constructions of this framework still utilize Garbled Circuit as a fundamental component, which accounts for 96% of the overall computational overhead of the circuit-PSI primitive. Therefore, we intuitively consider whether we can improve the efficiency of DPSI by performing the membership test without relying on GC. To this end, we propose the mqRPMT-DPSI framework. Different from the circuit-PSI where both parties learn the secret sharing of the membership vector based on GC, mqRPMT allows the sender to learn the complete membership vector efficiently.

A. mqRPMT-DPSI

The mqRPMT-DPSI framework offers improved efficiency at the price of relaxing security by allowing the protocol to leak some "extra" information. The functionality of mqRPMT allows the sender to obtain the membership vector v in plaintext. Although a permutation operator performed by the receiver can make sure the membership bits are of the permuted order unknown to the sender, the sender can still learn the cardinality of intersection (PSI-CA) by counting the number of "1" in v, or the size of $Y - X \bigcap Y$ (PSD-CA) by counting the number of "0" in v. Recent studies [26], [27] have found that adversaries can leverage an efficient attack on the size-revealing PSIs, using a binary-search-like strategy, to de-anonymize specific data records in the input sets, thereby leaking membership information about the input sets. To take advantage of mqRPMT without completely undermining the idea of secure computation, a natural candidate is to leak only DP information about the PSI-CA and PSD-CA.

A straightforward strategy to ensure that the leakage learned by the sender satisfies DP is to generate a random integer from a truncated discrete Laplace distribution and then add the corresponding number of dummy items to both the intersection and difference sets before the mqRPMT process.

However, in our two-party scenario without a trusted third party, the dummy items cannot be added to the intersection directly and the alternative strategy is that both parties generate a random integer from a truncated discrete Laplace distribution, respectively, and add the corresponding number of dummy items to their own private set. Unfortunately, this alternative strategy fails to satisfy DP. Note that a dummy item can be found in the intersection if and only if both the sender and receiver possess this dummy item. Consequently, the maximum number of dummy items in the intersection is inherently limited by the number of dummy items added by the sender locally. Consider a worst-case scenario in which the sender draws a random integer 0 from a truncated discrete Laplace distribution and adds 0 dummy items into his/her local set, the sender can learn the true PSI-CA regardless of the number of dummy items added by the receiver.

Definition 3 (Truncated Discrete Laplace Mechanism): Given a query $\mathbf{c} : \mathcal{D} \rightarrow \mathbb{N}$, the truncated discrete Laplace mechanism $TLap(D)$ outputs a integer $\max(\mathbf{c}(D)+\eta, \mathbf{c}(D))$,

where $\eta \in \mathbb{Z}$ follows a distribution, denoted by $L(\epsilon, \delta, \Delta\mathbf{c})$ that has a probability density function $\Pr[\eta = x] = p \cdot e^{-(\epsilon/\Delta\mathbf{c})|x-\eta^0|}$, where $p = \frac{e^{\epsilon/\Delta\mathbf{c}}-1}{e^{\epsilon/\Delta\mathbf{c}}+1}$, $\eta^0 = \Delta\mathbf{c} - 1 + \frac{\ln[(e^\epsilon+1)(1-\delta)]}{\epsilon}$, $\Delta\mathbf{c}$ is the sensitivity parameter for the query $\mathbf{c}$, ϵ is the privacy budget and δ is the error probability.

Lemma 3 (Discrete Laplace Mechanism Utility [57]): For $\eta \in \mathbb{Z}$ that follows $L(\epsilon, \delta, \Delta\mathbf{c})$ and any $m \in \mathbb{N}$, we have $\Pr[\eta \geq m] = \frac{e^{-\epsilon(m-\eta^0-1)}}{e^\epsilon+1}$.

Algorithm 1: Random Dummy Item Padding $\mathcal{M}_{rd}$

Parameters: Statistically security parameter σ. DP parameter ϵ_I, ϵ_D.
Input: Sender's input $X = \{x_1, x_2, ...x_n\}$. Receiver's input $Y = \{y_1, y_2, ...y_m\}$.
/* Sender side */
Compute R: the minimum number satisfying $Pr(TLap(\frac{1}{\epsilon_i}) > R) < 2^{-\sigma}$;
Initialize an empty set X';
Add padded real item $\{0|x\}_{x\in X}$ into X';
Add dummy items $\{0|i|0\}_{i\in R}$ into X';
Return X';
/* Receiver side */
$r_I = L(\epsilon_I, \delta, 1)$, $r_D = TLap(\epsilon_D, \delta, 1)$;
Initialize an empty set Y';
Add padded real item $\{0|y\}_{y\in X}$ into Y';
Add items $\{0|i|0\}_{i\in r_I}$ into Y' ; // Dummy items should be in X'
Add items $\{0|i|1\}_{i\in r_D}$ into Y' ; // Dummy items should not be in X'
Return Y'.

To this end, we modify the above strategy as follows, which is shown in Algorithm 1. In our strategy, the sender and receiver first independently pad the items in their own sets to bit strings with "0" at the highest position. Next, to ensure that the number of dummy items in the intersection remains oblivious to the sender, the sender adds R dummy items $\{1|i|0\}_{i\in[R]}$ into X', where R is typically a large integer leading to enough redundant dummy items in Y'. As a result, the number of dummy items added to the intersection is only determined by the dummy items generated by the receiver, which is oblivious to the sender. Simultaneously, the receiver draws two random integer r_I, r_D from two truncated discrete Laplace distributions, which is described in Definition 3, and adds r_I dummy items $\{1|i|0\}_{i\in[r]}$ and r_D dummy items $\{1|i|1\}_{i\in[r]}$ into X'. Finally, the sender and receiver exchange $|Y'| = n + R$ and $|X'| = m + r_I + r_D$ with each other. Consequently, the overall strategy bounds the leakage to P_1 with $(\epsilon_I + \epsilon_D, 2\delta)$-DP if and only if the truncated discrete Laplace mechanism satisfies (ϵ', δ)-DP, where $\epsilon' \in \{\epsilon_I, \epsilon_D\}$.

In the Random Dummy Item Padding Mechanism, determining the size of R is an important issue, as the magnitude of R simultaneously affects the computation complexity of the entire framework and the inherent error rate of the padding mechanism. In our implementation, we set R as the minimum number satisfying $\Pr[r_I > R] \leq 2^{-\sigma}$, where $\sigma = 40$ is the statistical security parameter in MPC. It means the number of dummy items added into the intersection satisfying (ϵ_I, δ)-DP with a high enough probability. The result of our experiment shows that: with fixed $\delta = 10^{-5}$, $\epsilon_I = 0.01, R = 2774$; $\epsilon_I = 0.1, R = 279$; $\epsilon_I = 1.0, R = 29$; $\epsilon_I = 10.0, R = 4$.

Parameters: The size of sender's set n. The DP parameters $\epsilon, \epsilon_i, \epsilon_d$. A random dummy item padding mechanism $\mathcal{M}_{rd}$. A randomized response mechanism $\mathcal{M}_{RR}$.
Input: The sender P_1's set $X = \{x_i\}_{i\in[n]}$, and the receiver P_2's set $Y = \{y_i\}_{i\in[m]}$.
Protocol:

1. **(Dummy Item Padding)** P_1 invokes $\mathcal{M}_{rd}$ with X as input and output a padded set $X' = \{x'_i\}_{i\in[n+R]}$, which means there are R items padded.
2. P_2 invokes $\mathcal{M}_{rd}$ with Y as input and output a padded set $Y' = \{y'_i\}_{i\in[M]}$, where $M = m + r_I + r_D$ and r_I, r_D are random integers drawn from a discrete Laplace distribution. Then, P_2 permute the Y' with the randomly generated permutation ϕ and obtain Y^*, where $y^*_i = y'_{\phi(i)}$.
3. **(mqRPMT Processing)** P_1 and P_2 invoke a mqRMPT protocol with inputs X' and Y^*. As the result of mqRPMT, P_1 gets an indication bit vector $\{v_i\}_{i\in[M]}$ such that $v_i = 1$ if and only if $y^*_i \in X'$ but without knowing y^*_i.
4. **(Membership vector Perturbation)** P_1 invokes $\mathcal{M}_{RR}$ with input v and gets the perturbed membership vector $\tilde{v}$ as output.
5. **(Intersection Computation)** P_2 receives the perturbed membership vector $\tilde{v}$ from P_1, permutes $\tilde{v}$ with ϕ^{-1} to obtain $\tilde{v}'$, and get the intersection vector $F \in \{0,1\}^m$ by deleting the bits that corresponding with the dummy item from $\tilde{v}'$.

Fig. 8: Efficient mqRPMT-based DPSI protocol.

We show the flow of the mqRPMT-DPSI protocol in Fig. 7 and give the details in Fig. 8. In the mqRPMT-DPSI, the sender P_1 and receiver P_2 first insert some dummy items into their private sets X and Y via a randomization mechanism called Random Dummy Item Padding Mechanism and get the padded sets X' and Y', respectively. Next, P_1 and P_2 invoke a construction of mqRPMT with inputs X' and Y'. As the functionality of mqRPMT, P_1 gets an indication bit vector $\{v_i\}_{i\in[M]}$ such that $v_i = 1$ if and only if $y'_i \in X'$ but without knowing y'_i. After getting $\{\langle v_i\rangle\}_{i\in[M]}$, P_1 perturbs it bit by bit via the RR mechanism and sends the perturbed boolean shares $\{\langle\tilde{v}_i\rangle\}_{i\in[M]}$ to P_2. Finally, P_2 receives $\{\langle\tilde{v}_i\rangle\}_{i\in[M]}$ and learns $F = \{0,1\}^m$ by deleting the bits corresponding to the dummy items from $\{\langle\tilde{v}_i\rangle\}_{i\in[M]}$.

Similar to the circuit-DPSI framework, the mqRPMT protocols invoked in mqRPMT-DPSI are pluggable and can be compatible with any future mqRPMT protocols.

Parameters: The size of the P_1's set n. The privacy budget $\epsilon_I, \epsilon_D, \epsilon$.
Input: The sender P_1's set $X = \{x_i\}_{i\in[n]} \in \mathcal{V}^n$. The receiver P_2's set $Y = \{y_i\}_{i\in[m]} \in \mathcal{V}^m$.
Functionality: Upon receiving X from P_1, Y from P_2.

1) Compute $F' \in \{0,1\}^m$, where f'_i indicates if $y_i \in X$. Apply $\mathcal{M}_{RR}$ with parameter ϵ to F' and send the result F to P_2.
2) **DP Leakage**: Sample r_I, r_D with $L(\epsilon_I, \delta, 1)$ and $L(\epsilon_D, \delta, 1)$. Compute and send $n_I = |X \bigcap Y| + r_I$, $n_D = Y - |X \bigcap Y| + r_D$ to P_1.

Fig. 9: Ideal functionality $\mathcal{F}_{rDPSI}$ with privacy leakage.

B. Security & Privacy Analysis of mqRPMT-DPSI

Theorem 4: The protocol in Fig. 8 satisfies ϵ-SIM-DPSI with the information leakage to P_1 bounded by $(\epsilon_I + \epsilon_D, 2\delta)$-DP.

Since both circuit-DPSI and mqRPMT-DPSI perturb the intersection in the same way using $\mathcal{M}_{RR}$, we omit the analysis of the DP property of the receiver's output and prove that the information leaked to the sender in Fig. 9 is bounded by $(\epsilon_I+\epsilon_D, 2\delta)$-DP. Also, according to the sequential composition theorem of DP, we only need to prove lemma 4.

Lemma 4: The truncated discrete Laplace mechanism satisfies (ϵ, δ)-differential privacy.

Proof: For any neighboring database D_1, D_2, let $c_1 = \mathbf{c}(D_1) \geq 0$ and $c_2 = \mathbf{c}(D_2) \geq 0$. W.L.O.G. we consider $c_2 \geq c_1$. It is easy to see that $\Pr[TLap(D_1) \in (-\infty, c_1)] = 0$ and $\Pr[TLap(D_2) \in (-\infty, c_2)] = 0$. For any $o \geq c_2 \geq c_1$, it is true that

$$\begin{aligned}
&\frac{\Pr[TLap(D_2) = o]}{\Pr[TLap(D_1) = o]} = \frac{p \cdot \exp\left(-(\epsilon/\Delta\mathbf{c})|o - c_1 - \eta^0|\right)}{p \cdot \exp\left(-(\epsilon/\Delta\mathbf{c})|o - c_2 - \eta^0|\right)} \\
&= \exp\left(\frac{\epsilon(|o - c_2 - \eta^0| - |o - c_1 - \eta^0|)}{\Delta\mathbf{c}}\right) \\
&\leq \exp\left(\frac{\epsilon|c_1 - c_2|}{\Delta\mathbf{c}}\right) \leq \exp\left(\frac{\epsilon \cdot \Delta\mathbf{c}}{\Delta\mathbf{c}}\right) = \exp(\epsilon).
\end{aligned} \tag{3}$$

However, when the output $o \in [c_1, c_2)$, $\Pr[TLap(D_2) = o] = 0$, $\Pr[TLap(D_1) = o] > 0$, making the ratio of probabilities unbounded. Nevertheless, we can bound $\Pr[TLap(D_1) \in [c_1, c_2)]$ by δ as shown below.

Let $O^* = (-\infty, c_2)$. Then, we can show that for any output set O of query $\mathbf{c}()$, we have

$$\begin{aligned}
&\Pr[TLap(D_1) \in O] \\
&= \Pr[TLap(D_1) \in (O \cap O^*)] + \Pr[TLap(D_1) \in (O - O^*)] \\
&\leq \Pr[TLap(D_1) \in [c_1, c_2)] + e^\epsilon \Pr[TLap(D_1) \in (O - O^*)] \\
&= \Pr[\eta_1 < \Delta\mathbf{c}] + e^\epsilon \Pr[TLap(D_1) \in O] \\
&= 1 - \Pr[\eta_1 \geq \Delta\mathbf{c}] + e^\epsilon \Pr[TLap(D_1) \in O] \\
&= 1 - \frac{e^{-\epsilon(\Delta\mathbf{c} - \eta^0 - 1)}}{e^\epsilon + 1} + e^\epsilon \Pr[TLap(D_1) \in O] \\
&= \delta + e^\epsilon \Pr[TLap(D_1) \in O].
\end{aligned} \tag{4}$$

Therefore, this mechanism satisfies (ϵ, δ)-DP. ■

Lemma 5: The protocol in Fig. 8 securely realizes the functionality in Fig. 9 against a semi-honest adversary in the $\mathcal{F}_{mqRPMT}$-hybrid model.

We can prove the above theorem by constructing the simulators similar to the proof of Lemma 2 and omit the details here due to space limitation.

VII. Experiments

In this section, we describe the details of our implementation and evaluate the performance of the proposed DPSI constructions.

A. Implementation Details

We implement three circuit-DPSI protocols and an mqRPMT-DPSI protocol by plugging the SOTA circuit-PSI protocols [16]–[18] and mqRPMT protocol [25] into the proposed DPSI frameworks, respectively. Moreover, we also implement all circuit-PSI protocols and mqRPMT-based PSI protocol above for comparison. Specifically, all these protocols mentioned above are written in JAVA and the source code is available at the module mpc4j-work-dpsi of the root project mpc4j (https://github.com/alibaba-edu/mpc4j).

We denote the circuit-PSI protocols and mqRPMT-based PSI protocol, which are serving as primitives, as PSTY'19, RS'21, CGS'22, and CZZ'22. The DPSI protocols that are built upon them are denoted as PSTY'19*, RS'21*, CGS'22*, and CZZ'22*. Moreover, we also list the running time and communication cost of the FHE-based circuit-DPSI protocol mentioned in [15] and denote it as KKL+'20. Note that the experimental results of KKL+'20 are copied from [15] since this protocol is obviously inefficient even though it is written in C/C++, and the further detailed comparison between this protocol with our protocols is unnecessary.

B. Datasets

We conduct different experiments on both synthetic and real datasets. For evaluating the runtime and communication cost of implemented protocols, which are only affected by the size of dataset, we construct synthetic datasets separately for both parties, each containing distinct keywords with some overlap. For evaluating the utility of PSI-CA and PSI-SUM, we use the widely applied real-world dataset TPC-H. We take the ORDERKEY and LINEKEY attributes from the LINEITEM table of TPC-H as keywords, the DISCOUNT attribute as the value, and split this table for both parties.

C. Experimental Setup

For all implemented protocols, we set the computational security parameter $\kappa = 128$ and the statistical security parameter $\sigma = 40$. To evaluate the performance of implemented protocols, we conduct a series of experiments with different sizes of items: $n = m = 2^{14}, 2^{16}, 2^{18}, 2^{20}$, as well as different network environments, including LAN and WAN. In addition, we set the corresponding maximum size of items $N = M = 2^{15}, 2^{17}, 2^{19}, 2^{21}$. For the DP mechanism used in the implemented DPSI protocols, we use the following default values unless specifically stated: $\epsilon = 1.0, \delta = \frac{1}{n}$.

TABLE I: Runtime of DPSI protocols on $n = 2^{14}, 2^{16}, 2^{18}, 2^{20}$ set sizes under LAN and WAN network settings. The results for KKL+'20 are copied from [15]. The best results are marked as green. The best results under the circuit-PSI framework are marked as blue.

Primitive	Protocol	Runtime(s)							
		LAN				WAN			
		2^{14}	2^{16}	2^{18}	2^{20}	2^{14}	2^{16}	2^{18}	2^{20}
Circuit-PSI	PSTY'19	4.26	19.57	96.97	457.74	8.52	23.95	107.06	474.90
	RS'21	5.35	20.98	99.96	474.68	8.67	27.36	111.93	488.63
	CGS'22	5.29	21.81	102.67	485.09	8.47	26.25	110.51	501.74
Circuit-DPSI	PSTY'19⋆	4.14	19.15	94.93	451.62	5.11	19.90	99.92	488.06
	RS'21⋆	4.95	19.67	97.21	465.22	5.46	21.19	99.11	479.51
	CGS'22⋆	4.48	20.75	102.25	476.95	9.44	26.03	111.79	503.86
	KKL+'20	-	531.62	-	11719.4	-	-	-	-
mqRPMT-PSI	CZZ'22	1.34	5.40	21.40	85.33	1.67	6.01	22.51	88.33
mqRPMT-DPSI	CZZ'22⋆	1.19	4.81	19.14	77.22	1.61	5.76	21.55	85.37

TABLE II: Communication cost of DPSI protocols on $n = 2^{14}, 2^{16}, 2^{18}, 2^{20}$ set sizes. The results for KKL+'20 are copied from [15]. The best results are marked as green.

Primitive	Protocol	Total Comm.(MB)			
		2^{14}	2^{16}	2^{18}	2^{20}
PSI	PSTY'19	2.63	8.82	32.08	125.93
	RS'21	3.27	9.56	33.03	127.08
	CGS'22	4.73	17.15	65.97	261.39
	CZZ'22	1.88	7.5	31	124
Circuit-DPSI	PSTY'19⋆	2.64	9.04	32.16	126.25
	RS'21⋆	3.27	9.58	33.11	127.4
	CGS'22⋆	5.74	17.17	66.05	261.76
	KKL+'22	-	133	-	232
mqRPMT-DPSI	CZZ'22⋆	1.41	5.88	23.5	94

We run all our protocols and related protocols on a single Intel Core i9-9900K with 3.6GHz and 128GB RAM. We simulate the network connection using Linux tc command. For the WAN setting, we set the average RTT to be 80 ms and bandwidth to be 100 Mbps. We use iptables command to calculate the communication cost, and use the running time to compute the computation complexity, which is the maximum time from the protocol beginning to the end, including the messages transmission time.

D. Running time of Implemented Protocols

Table I shows the running time of all the implemented protocols, and we can draw some interesting conclusions. First, we can find that the running time of all DPSI protocols increases linearly with the growth of the data size. Secondly, it is obvious that, among the PSI and DPSI protocols we evaluated, the mqRPMT-DPSI protocol CZZ'22⋆ consistently outperformed the others in all network environments. Specifically, the CZZ'22⋆ is approximately 3.2-5.8× faster than the other circuit-DPSI protocols. This is consistent with our analyses that mqRPMT-DPSI protocols will be faster than the circuit-DSPI protocols since the latter uses GC as a fundamental technology. Third, we observe that the PSTY'19⋆ is the fastest among all circuit-DPSI protocols over different sizes of data in the WAN setting and when $n = 2^{14}, 2^{16}$ in the LAN setting. The reason is that we implement the PSTY'19⋆ by combining the circuit-PSI protocol in [16] with the PSM protocol in [18]. Finally, surprisingly, by comparing the running time of DPSI protocols with their corresponding PSI protocols which are used as primitives, we find that the running time of all circuit-DPSI protocols is roughly the same as their corresponding circuit-PSI protocols. However, the running time of the mqRPMT-DPSI protocol is even shorter than the mqRPMT-based PSI protocol. The reason is that, compared to the original circuit-PSI protocols, the corresponding circuit-DPSI protocols only add a secret sharing perturbation phase, which utilizes an efficient RR mechanism. Therefore, the computational overhead is almost negligible. On the other hand, compared to the mqRPMT-based PSI protocols, although the mqRPMT-DPSI protocol needs to add a few dummy items into both parties' sets in advance, it eliminates the step of sending the receiver the mapping of perturbed membership vector with the items in sender's set via OT. As a result, CZZ'22⋆ is slightly faster than CZZ'22.

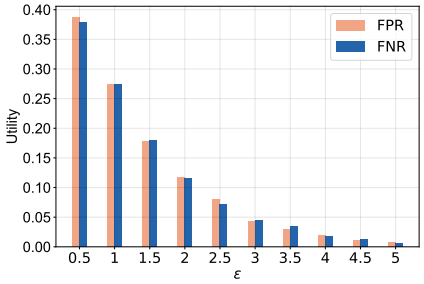


Fig. 10: The FPR and FNR of mqRPMT-DPSI protocol

E. Communication Cost of Implemented Protocols

Table II shows the total communication cost of all implemented protocols. According to the results, we can find that all DPSI protocols achieve strict linear communication complexity. Specifically, except for the CGS'22⋆, the communication overhead of the other DPSI protocols is relatively consistent. Among them, the mqRPMT-DPSI protocol has the lowest communication overhead. Moreover, compared with the constructions of the corresponding PSI primitives, all circuit-DPSI protocols have similar communication overheads. This is because the perturbation operations involved in the DPSI protocols are performed locally and do not incur additional communication overhead. Furthermore, CZZ'22⋆ achieves a reduction in communication costs by eliminating the OT step.

F. Utility of Implemented DPSI Protocols

1) FPR and FNR: The output of the DPSI protocols is a differentially private set, which contains both false negatives and false positives. According to the Equation 1, both the false negative rate (FNR) and false positive rate (FPR) are theoretically $q = \frac{1}{1+e^{\epsilon}}$, which indicates that the FNR and FPR are determined by the privacy budget ϵ: a larger ϵ will lead to smaller FNR and FPR at the expense of weaker privacy guarantee, while a smaller ϵ will result in lower utility. In practice, the parties need to negotiate and define a consensus ϵ in advance to strike a balance between privacy and utility that aligns with their specific needs.

To verify our theoretical analysis ofw the utility of DPSI protocols, we additionally design a series of experiments, where we measure the FPR and FNR of the chosen implemented DPSI protocol while varying the private budget ϵ from 0.5 to 5.0. Specifically, since the utility of all DPSI protocols is the same, we chose the mqRPMT-DPSI protocol as a representative protocol and conducted experiments to evaluate its FPR and FNR under different privacy budgets. The results in Fig. 10 support and validate our analyses: as privacy budget ϵ increases, the FPR and FNR will decrease gradually. It indicates that the parties can strike a balance between privacy and utility by choosing a proper ϵ.

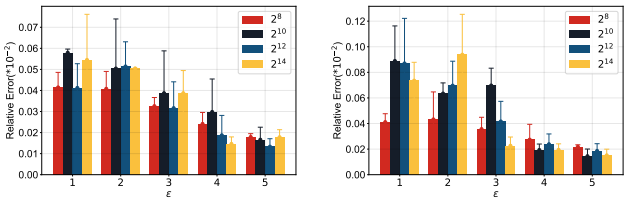
2) PSI-CA and PSI-SUM: Besides the FPR and FNR, the utility of DPSI protocols can also be evaluated by measuring the accuracy of PSI-CA and PSI-SUM, where former counts the cardinality of the intersection and the latter sums the measures associated with the items in the intersection. Theoretically speaking, the results of PSI-CA and PSI-SUM over the output of our DPSI protocols are unbiased estimations of the true PSI-CA and PSI-SUM, the accuracy of which is determined by the RR mechanism used in the perturbation phase of the DPSI protocols, whose variance is $Var[\tilde{c}] = \frac{e^{\epsilon}}{n\cdot(e^{\epsilon}-1)^2}$. For evaluating the accuracy of the DPSI protocols' outputs, we conduct a series of experiments and measure the Relative Error $RE = \frac{|c-c'|}{c}$ of the outputs, where c and c' are the cardinality of the intersection and the cardinality of the output of DPSI protocol, respectively. Specifically, we measure the REs of CZZ'22* while varying the size of sets n, m and the privacy budget ϵ in the LAN setting.

The results are shown in Fig. 11, where the bars represent the RE and the vertical lines over the bars represents the variance. By observing Fig. 11(a) or Fig. 11(b), we can find that the results are consistent with the theoretical analyses above. On the one hand, as the privacy budget ϵ increases, the REs and the variance of the results exhibit a gradual reduction. On the other hand, the variance of REs also decreases as the size of each party's sets increase. Additionally, by comparing Fig. 11(a) with Fig. 11(b), we can find that the variance of PSI-CA is less than that of PSI-SUM in most cases. It can be explained as follows. In PSI-SUM, the items have different measures, which means that even if the cardinality of the intersection remains unchanged, perturbing different items will also affect the final results, leading to greater variance.

Notice that, besides the DPSI protocols, there are also a series of works are designed for computing differentially private PSI-CA or PSI-SUM with high utility. However, we do not compare these works with out DPSI protocols here since these works are designed for different scenarios: the results of DPSI can be directly released to the client before being aggregated whereas the existing works cannot. This means that the DPSI protocols can provide stronger privacy protection, which inevitably comes at the cost of utility degrade.

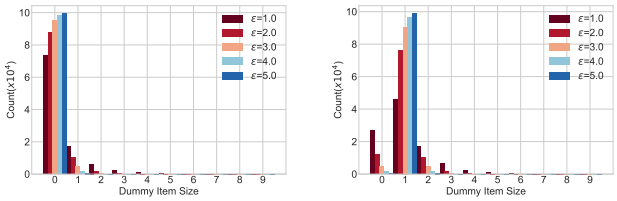
G. Effectiveness of Random Dummy Item Padding Mechanism

Finally, we evaluate the effectiveness of the Random Dummy Item Padding Mechanism, which is the key building block in the mqRPMT-DPSI framework. We count the distribution of Random Dummy Item Padding Mechanism's outputs in a different setting. Specifically, we run Random Dummy Item Padding Mechanism for 10^5 times while varying the privacy budget ϵ from 1.0 to 5.0 under the setting of $\delta = 0.1$ and $\delta = 0.0001$. In all experiments, we use the default values $n = m = 2^9$, and show the results in Fig. 12. It demonstrates that when $\delta = 0.0001$, which is a much more practical setting in the above experiments, as the privacy budget ϵ increases, the distribution of the outputs becomes centralized gradually. It means that a smaller ϵ can lead to a stronger privacy-preserving capability. In contrast, when $\delta = 0.1$, the output is not adequately randomized, implying a higher probability of accidental information leakage.



(a) PSI-CA (b) PSI-SUM

Fig. 11: The utility of PSI-CA in mqRPMT-DPSI protocol



(a) $\delta = 0.1$ (b) $\delta = 0.0001$

Fig. 12: The distribution of dummy items' size

VIII. Conclusion

In this paper, we revisit the definitions of DPSI and propose two DPSI frameworks. The first one is a generalized circuit-DPSI framework which is secure and satisfies ϵ-DP. The other one is an mqRPMT-DPSI framework which offers improved efficiency at the expense of privacy compromise. Specifically, in the mqRPMT-DPSI framework, we come up with a Random Dummy Item Padding Mechanism to prevent the exact PSI-CA from being leaked. We prove that the released PSI-CA satisfies (ϵ, δ)-DP and the output of mqRPMT-DPSI framework is secure and satisfies ϵ-DP. We implement numerous of constructions based on proposed DPSI frameworks and compare them with the state-of-the-art PSI. Experiments show that our constructions are arguably the most computation and communication efficient DPSI protocol to date, and they are even comparable to the most efficient PSI protocols.

REFERENCES

[1] Information Commissioner's Office, "Chapter 5: Privacy-enhancing technologies," 2022.

[2] A. Edmonds, A. Nikolov, and J. R. Ullman, "The power of factorization mechanisms in local and central differential privacy," in *STOC 2020*. ACM, 2020, pp. 425–438.

[3] M. Hardt, K. Ligett, and F. McSherry, "A simple and practical algorithm for differentially private data release," in *NeurIPS 2012*. NeurIPS Foundation, 2012, pp. 2348–2356.

[4] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 1021–1032, 2010.

[5] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *PODS 2010*. ACM, 2010, pp. 123–134.

[6] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi, "The matrix mechanism: optimizing linear counting queries under differential privacy," *Proc. VLDB Endow.*, vol. 24, no. 6, pp. 757–781, 2015.

[7] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *SIGMOD 2010*. ACM, 2010, pp. 735–746.

[8] G. Yuan, Y. Yang, Z. Zhang, and Z. Hao, "Convex optimization for linear query processing under approximate differential privacy," in *SIGKDD 2016*. ACM, 2016, pp. 2005–2014.

[9] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao, "Optimizing batch linear queries under exact and approximate differential privacy," *ACM Trans. Database Syst.*, vol. 40, no. 2, pp. 11:1–11:47, 2015.

[10] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong privacy for analytics in the crowd," in *SOSP 2017*. ACM, 2017, pp. 441–459.

[11] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *CCS 2014*. ACM, 2014, pp. 1054–1067.

[12] J. C. Duchi, M. I. Jordan, and M. J. Wainwrigh, "Local privacy and statistical minimax rates," in *FOCS 2013*, 2013.

[13] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory meets practice on the map," in *ICDE 2008*. IEEE Computer Society, 2008, pp. 277–286.

[14] S. Haney, A. Machanavajjhala, J. M. Abowd, M. Graham, M. Kutzbach, and L. Vilhuber, "Utility cost of formal privacy for releasing national employer-employee statistics," in *SIGMOD 2017*. ACM, 2017, pp. 1339–1354.

[15] B. Kacsmar, B. Khurram, N. Lukas, A. Norton, M. Shafieinejad, Z. Shang, Y. Baseri, M. Sepehri, S. Oya, and F. Kerschbaum, "Differentially private two-party set operations," in *EuroS&P 2020*. IEEE, 2020, pp. 390–404.

[16] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based psi with linear communication," in *Eurocrypto 2019*. Springer, 2019, pp. 122–153.

[17] P. Rindal and P. Schoppmann, "Vole-psi: fast oprf and circuit-psi from vector-ole," in *Eurocrypto 2021*. Springer, 2021, pp. 901–930.

[18] N. Chandran, D. Gupta, and A. Shah, "Circuit-psi with linear complexity via relaxed batch opprf," *Proceedings on Privacy Enhancing Technologies Symposium*, vol. 2022, no. 1, pp. 353–372, 2022.

[19] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC 2009*. ACM, 2009, pp. 169–178.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[21] J. Bater, Y. Park, X. He, X. Wang, and J. Rogers, "SAQE: practical privacy-preserving approximate query processing for data federations," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2691–2705, 2020.

[22] Y. Wang and K. Yi, "Secure yannakakis: Join-aggregate queries over private data," in *SIGMOD 2021*. ACM, 2021, pp. 1969–1981.

[23] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Spot-light: lightweight private set intersection from sparse ot extension," in *CRYPTO 2019*. Springer, 2019, pp. 401–431.

[24] C. Zhang, Y. Chen, W. Liu, M. Zhang, and D. Lin, "Linear private set union from {Multi-Query} reverse private membership test," in *USENIX Security 2023*. USENIX, 2023, pp. 337–354.

[25] Y. Chen, M. Zhang, C. Zhang, M. Dong, and W. Liu, "Private set operations from multi-query reverse private membership test," in *PKC 2024*. Springer Nature Switzerland, 2024, pp. 387–416.

[26] X. Guo, Y. Han, Z. Liu, D. Wang, Y. Jia, and J. Li, "Birds of a feather flock together: How set bias helps to deanonymize you via revealed intersection sizes," in *USENIX Security 2022*. USENIX, 2022, pp. 1487–1504.

[27] B. Jiang, J. Du, and Q. Yan, "Anonpsi: An anonymity assessment framework for PSI," *Cryptology ePrint Archive*, 2024.

[28] C. Meadows, "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party," in *S&P 1986*. IEEE, 1986, pp. 134–134.

[29] E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *FC 2010*. Springer, 2010, pp. 143–159.

[30] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute," *Cryptology ePrint Archive*, 2020. [Online]. Available: https://eprint.iacr.org/2020/599

[31] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *EuroS&P 2020*. IEEE, 2020, pp. 370–389.

[32] M. Yagisawa, "Fully homomorphic encryption without bootstrapping," *Cryptology ePrint Archive*, 2015. [Online]. Available: https://eprint.iacr.org/2011/277

[33] A. C.-C. Yao, "How to generate and exchange secrets," in *FOCS 1986*. IEEE, 1986, pp. 162–167.

[34] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *USENIX Security 2015*. USENIX, 2015, pp. 515–530.

[35] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Crypto 2003*. Springer, 2003, pp. 145–161.

[36] V. Kolesnikov and R. Kumaresan, "Improved ot extension for transferring short secrets," in *Crypto 2013*. Springer, 2013, pp. 54–70.

[37] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer extensions with security for malicious adversaries," in *Eurocrypto 2015*. Springer, 2015, pp. 673–701.

[38] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious prf with applications to private set intersection," in *CCS 2016*. ACM, 2016, pp. 818–829.

[39] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Psi from paxos: fast, malicious private set intersection," in *EUROCRYPT 2020*. Springer, 2020, pp. 739–767.

[40] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl, "Efficient two-round ot extension and silent non-interactive secure computation," in *CCS 2019*. ACM, 2019, pp. 291–308.

[41] G. Couteau, P. Rindal, and S. Raghuraman, "Silver: silent vole and oblivious transfer from hardness of decoding structured ldpc codes," in *Crypto 2021*. Springer, 2021, pp. 502–534.

[42] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Crypto 2021*. Springer, 2021, pp. 395–425.

[43] S. Raghuraman and P. Rindal, "Blazing fast psi from improved okvs and subfield vole," in *CCS 2022*. ACM, 2022, pp. 2505–2517.

[44] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS 2012*, 2012.

[45] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *CCS 2017*. ACM, 2017, pp. 1243–1255.

[46] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," in *CCS 2018*. ACM, 2018, pp. 1223–1237.

[47] A. Narayan and A. Haeberlen, "Djoin: Differentially private join queries over distributed databases," in *OSDI 2012*. USENIX, 2012, pp. 149–162.

[48] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers, "Shrinkwrap: efficient sql query processing in differentially private data federations," *Proceedings of the VLDB Endowment*, vol. 12, no. 3, 2018.

[49] X. He, A. Machanavajjhala, C. Flynn, and D. Srivastava, "Composing differential privacy and secure computation: A case study on scaling private record linkage," in *CCS 2017*. ACM, 2017, pp. 1389–1406.

[50] A. Groce, P. Rindal, and M. Rosulek, "Cheaper private set intersection via differentially private leakage," *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 6–25, 2019.

[51] P. Rindal and M. Rosulek, "Malicious-secure private set intersection via dual execution," in *CCS 2017*. ACM, 2017, pp. 1229–1242.
[52] O. Goldreich, *Foundations of cryptography: volume 2, Basic Applications*. Cambridge University Press, 2009.
[53] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
[54] R. Pagh and F. F. Rodler, "Cuckoo hashing," in *ESA 2001*. Springer, 2001, pp. 121–133.
[55] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS 2023*. IEEE, 2013, pp. 429–438.
[56] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
[57] C. L. Canonne, G. Kamath, and T. Steinke, "The discrete gaussian for differential privacy," in *NeurIPS 2020*. NeurIPS Foundation, 2020.