Security and Communication Networks

Multiple Private Set Intersection from Reusable Oblivious PRF

Qiang Liu,[1,2,3] Xiaojun Chen,[1,2] Weizhan Jing,[1,2,3] Yansong Zhang,[1,2,3] and Xudong Chen[1,2,3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China
[2] State Key Laboratory of Cyberspace Security Defense, Beijing 100085, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Correspondence should be addressed to Xiaojun Chen; chenxiaojun@iie.ac.cn

Abstract

Private set intersection is a technique that enables two or multiple parties to find their common elements while keeping all other data confidential and hidden. The research on two-party PSI has been well-developed. However, their dataset is often updated dynamically. When both parties add a new dataset and update their dataset, they need to calculate a new updatable intersection. As new datasets are continually added, there will be multiple PSI between the updated datasets. The research on existing PSI protocols has paid little attention to this scenario. We build upon this and design an efficient multiple PSI protocol ensuring semi-honest security under the plain model. Overall, based on the oblivious pseudorandom functions (OPRF) of protocol (*Chase et al., Crypto 2020*), we build the streaming PSI firstly, it allows the OPRF from previous PSI to be reused in subsequent PSI. Then we combine it with some lightweight operations to construct our multiple PSI protocol. Furthermore, the idea of our protocol is applicable to many two-party PSI protocols, for which we provide a general framework. Compared to the state-of-the-art protocol (*Raghuraman et al., CCS 2022*), ours has the lowest communication and runtime in multiple PSI scenario. For instance, in the second PSI, when the dataset size is $2^{24} + 2^{16}$ in both parties, our protocol achieves a 50× (2.6×) improvement in terms of communication (runtime).

1. Introduction

In recent years, with the increasing awareness of data security among organizations and individuals, research on privacy-preserving technologies has been further developed. PSI is a research hotspot in the field of privacy preserving, which avoids additional information leakage while calculating the intersection.

Initial two-party Private set intersection (PSI) protocols rely on the Diffie-Hellman (DH) assumption [1–4], which results in low communication costs but significant computational complexity. However, with the introduction of oblivious transfer (OT) extension [5], some protocols [6–9] leveraging OPRF have been developed, where the sender obtains a PRF key and the receiver receives PRF values derived from its inputs, and these protocols achieve a good balance between computational and communication overhead. In particular, the protocols [10–13] have significantly improved overall performance in recent years. From a practical standpoint, two-party PSI has become feasible with rapid implementations across various scenarios, such as private contact discovery [14-16], kinship testing [17] and privacy-preserving contact tracing [18, 19]. However, in these scenarios, the above protocols do not take into account the dynamic updates of participant datasets, when both parties add a new dataset and update their dataset, they need to calculate a new updatable intersection.

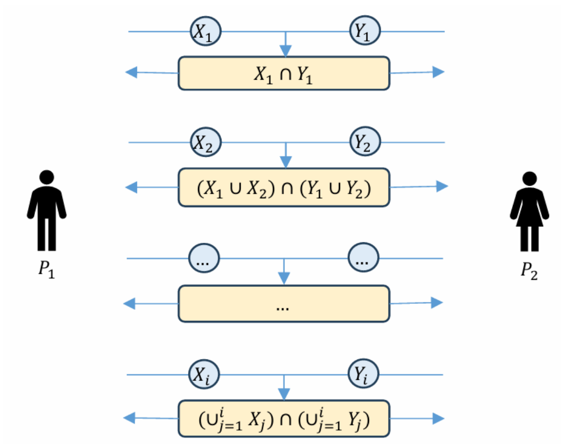

Figure 1: The multiple PSI scenario.

We focus on the multiple PSI scenario in which the dataset is updated dynamically, and we depict it in Figure 1. In this scenario, P_1 has initial dataset X_1 and newly added datasets $\{X_2, X_3, \cdots\}$, P_2 has initial dataset Y_1 and newly added datasets $\{Y_2, Y_3, \cdots\}$. In the initial PSI, they want to get the result of $\mathcal{F}_{PSI}(X_1, Y_1)$ (we use ideal function $\mathcal{F}_{PSI}(X, Y)$ to represent the functionality of computing $X \cap Y$, as illustrated in Figure 2). And in the i-th PSI, they want to get the result of $\mathcal{F}_{PSI}(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j)$. This is a beach that has been less explored before, and it exists in many two-party PSI scenarios. For instance, in private contact discovery [14-16], the user's contact database is matched with the application's database to discover private contacts. Since both the user's contact data and the application's database are dynamically updated, irregular PSI is required by both parties.

The existing solution is directly applicable to multiple PSI scenarios, but it incurs considerable overhead. Consider this: for $i > 1$, to obtain $\mathcal{F}_{PSI}(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j)$ in the i-th PSI, there are two naive methods to calculate the result below.

- *Protocol_All*. If we want to get the result of $\mathcal{F}_{PSI}(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j)$, a direct way is re-executing the $\mathcal{F}_{PSI}(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j)$ with all datasets from both parties.
- *Protocol_Split*. We can use the $(i-1)$-th PSI result to calculate the i-th PSI result with less overhead. Since $\mathcal{F}_{PSI}(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j) = \mathcal{F}_{PSI}(\cup_{j=1}^{i-1} X_j, \cup_{j=1}^{i-1} Y_j) \cup \mathcal{F}_{PSI}(\cup_{j=1}^{i-1} X_j, Y_i) \cup \mathcal{F}_{PSI}(X_i, \cup_{j=1}^{i} Y_j)$, only the $\mathcal{F}_{PSI}(\cup_{j=1}^{i-1} X_j, Y_i) \cup \mathcal{F}_{PSI}(X_i, \cup_{j=1}^{i} Y_j)$ needs to be calculated.

Clearly, the overall overhead is linearly related to the updated dataset $\cup_{j=1}^{i} X_j$ or $\cup_{j=1}^{i} Y_j$, as new datasets accumulate, the overhead will become increasingly unaffordable.

We find that if subsequent PSI reuses the OPRF from previous PSI rather than generating a new OPRF itself, it can significantly reduce overhead. Therefore, in our paper, we design the streaming PSI, which reuses the OPRF from the previous PSI. This approach realizes that the overall overhead is linearly related to the size of the newly added dataset, and we use this as a foundational building block and combine some lightweight operations to construct our multiple PSI protocol.

IDEAL FUNCTIONALITY $\mathcal{F}_{PSI}(X, Y)$
Input: P_1, P_2 input dataset X and Y, $|X| = N_X, |Y| = N_Y$.
Output: $X \cap Y$.

1) For all $i \in [1, N_X]$, P_1 inputs a set of elements $X = \{x_i\}$ where $x_i \in \{0,1\}^*$. For all $j \in [1, N_Y]$, P_2 inputs a set of elements $Y = \{y_j\}$ where $y_j \in \{0,1\}^*$.
2) Output the set intersection $I = X \cap Y$

Figure 2: Ideal functionality for $\mathcal{F}_{PSI}(X, Y)$.

1.1. Our Contribution

- Firstly, based on the OPRF from protocol [21], we introduce the streaming PSI that allows the reuse of OPRF in subsequent PSIs. This results in a linear relationship between the protocol's overhead and the newly added dataset.
- Secondly, to avoid significant overhead due to continuous data accumulation, we generate new reusable OPRF whenever the dataset grows beyond a certain threshold, significantly improving the efficiency of subsequent PSI. We also provide a formal simulator-based security proof for our multiple PSI protocol.
- Additionally, we provide a version of privacy-enhancing and a general framework that allows our protocol's approach to be applied to other OPRF-based PSI protocols [12,13,22], enabling a smooth transition to multiple PSI with minimal modifications.
- Finally, we implemented our protocol in C++, and the experiments show that our protocol has obvious advantages in these scenarios. Compared to the state-of-the-art protocols [12,13,21,22], our protocol achieves up to a 50 × improvement in communication overhead and a 2.6 × improvement in runtime.

1.2. Related work

The SOTA PSI protocols. There are many research works in the two-PSI field, including Diffie Hellman-based protocols [1, 23], circuit-based protocols [24-26], OT-based protocols [12,13,21,22]. Among them, OT-based PSI is currently optimal.

The semi-honest PSI protocol [22] enables efficient computation using only oblivious transfer (OT), hash function, symmetric-key, and bitwise operations, but comes with high communication overhead. The protocol [21] achieves a good balance between computation and communication through lightweight oblivious pseudorandom functions (OPRF). Subsequently, protocol [11] introduces the oblivious key value store (OKVS) to protect the key value associations. Protocol [12] shows that by combining the OKVS with vector oblivious linear evaluation [27, 28], an efficient PSI protocol can be achieved. Recently, protocol [13] has combined the VOLE from protocol [28] with the improved OKVS from protocol [10], resulting in the current optimal protocol.

Protocols similar to ours. Protocol [29] defines two specific settings to support dataset updates using DH and HE primitives. UPSI with addition is the first setting, parties can add new elements to their datasets every day. UPSI weak deletion is the second setting, parties can also delete their old elements every t days, then both parties' databases maintain a fixed size (approximately equivalent to the data of t days). This differs from our protocol, where both parties can continuously accumulate data until a certain limit is reached, at which point the protocol becomes invalid. Later, protocol [30] further extends the functionality of protocol [29] to include PSI-Cardinality and PSI-Sum. Additionally, the protocol constructs an ORAM tree using an oblivious root-to-leaf path, supporting dataset updates.

1.3. Organization

The remainder is organized as follows. We briefly review the background knowledge in Sections 2. Then we construct the streaming PSI and our multiple PSI protocol in Section 3. The experimental evaluation is in Section 4, and the conclusion is in Section 5.

2. Preliminaries

2.1. Notation

We designate $[n]$, $A[i]$ as the set $\{1,\cdots,n\}$ and a matrix A with i-th column, designate δ,λ as the parameters for statistical and computational security. For dataset X_i and $X_{[i,n]}$, we designate $N_{X_i}, X_{[i,n]}$ as the size $|X_i|$ and $\bigcup_{j=i}^{n} X_j$. The hamming weight between string x and 0 is denoted by $\|x\|_H$.

2.2. Oblivious Transfer

Oblivious Transfer (OT) [31] is an important cryptographic tool that is widely applied in privacy preserving computations. It can be described as a protocol where the sender sends two or more messages, and the receiver only receives one of them, remaining oblivious to the others. In this context, we consider the 1-out-of-2 OT, where the sender sends two messages, and the receiver receives one, with no knowledge of the other.

2.3. OPRF from Protocol [21]

Our protocol utilizes the Oblivious Pseudorandom Function (OPRF) [32] cryptographic primitive, which allows the sender to obtain the PRF key while the receiver obtains the corresponding PRF values. Finally, the sender transmits the values encrypted with the PRF key to the receiver. The receiver then compares the two sets to obtain the intersection, with no information leaked other than the intersection itself.

The protocol [21] is described in Figure 3. First, for each $y \in Y_1$, let $v = F_k\big(H_1(y)\big)$, then $D_i^{(1)}\big[v[i]\big] = 0$ for all $i \in [w]$, and hence $A_i^{(1)}\big[v[i]\big] = B_i^{(1)}\big[v[i]\big]$. After both parties run w OTs for $i \in [w]$, P_1 gets $C_i^{(1)}$ as the receiver. Finally, for each $x \in X_1$, $y \in Y_1$, P_2 compares $H_2(C_1^{(1)}\big[v[1]\big] \parallel \cdots \parallel C_w^{(1)}\big[v[w]\big])$ with $H_2(A_1^{(1)}\big[v[1]\big] \parallel \cdots \parallel A_w^{(1)}\big[v[w]\big])$ to get the result.

In a high level, the protocol [21] constructs an effective OPRF. P_1 gets the PRF key, which is consist of matrix $C^{(1)}$ with dimension $m \times w$. P_2 gets the PRF values Ψ_{Y_1}, they satisfy the following conditions: If $x \in Y_1$, then $\psi_x \in \Psi_{Y_1}$. If $x \notin Y_1$, the parameters m, w in the protocol are chosen such that there are at least d 1's in $\{D_1^{(1)}\big[v[1]\big], \cdots, D_w^{(1)}\big[v[w]\big]\}$, and the value $\psi_x = H_2(C_1^{(1)}\big[v[1]\big] \parallel \cdots \parallel C_w^{(1)}\big[v[w]\big])$ is pseudorandom to P_2 (in Definition II.1). We denote this protocol as $\Pi_{CM\_PSI}(X_1, Y_1)$.

PROTOCOL $\Pi_{CM\_PSI}(X_1, Y_1)$
Input: P_1, P_2 input dataset X_1 and Y_1, $|X_1| = N_{X_1}, |Y_1| = N_{Y_1}$.
Output: $X_1 \cap Y_1$.
P_1 and P_2 agree on security parameters λ, σ, protocol parameters m, w, ℓ_1, ℓ_2, hash function $H_1 : \{0,1\}^* \to \{0,1\}^{\ell_1}, H_2 : \{0,1\}^w \to \{0,1\}^{\ell_2}$, uniformly random key $k \xleftarrow{\$} \{0,1\}^\lambda$, pseudorandom function $F : \{0,1\}^{\ell_1} \times \{0,1\}^\lambda \to \{m\}^w$.

1) Precomputation
   - P_1 samples a random string $s \xleftarrow{\$} \{0,1\}^w$.
   - P_2 does the following:
     a) Initialize an $m \times w$ binary matrix $D^{(1)}$ to all $1's$. Denote its column vectors by $D_1^{(1)} = \cdots = D_w^{(1)} = 1^m$.
     b) For each $y \in Y_1$, computer $v = F_k(H_1(y))$. Set $D_i^{(1)}(v[i]) = 0$ for all $i \in [w]$.
2) Oblivious Transfer
   - P_2 randomly samples an $m \times w$ binary matrix $A^{(1)} \xleftarrow{\$} \{0,1\}^{m \times w}$. Compute Matrix $B^{(1)} = A^{(1)} \oplus D^{(1)}$.
   - P_1 and P_2 run w oblivious transfer where P_2 is the sender with inputs $\{A_i^{(1)}, B_i^{(1)}\}_{i \in [w]}$ and P_1 is the receiver with inputs $s[1], \cdots, s[w]$. As a result, P_1 obtains w OT number of m-bit strings as the column vectors of matrix $C^{(1)}$ (with dimension $m \times w$).
3) OPRF Evaluation
   - For each $x \in X_1$, P_1 computes $v = F_k(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]]||\cdots||C_w^{(1)}[v[w]])$, sends ψ_x to P_2.
   - For each $y \in Y_1$, P_2 computes $v = F_k(H_1(y))$ and its OPRF value $\psi_y = H_2(A_1^{(1)}[v[1]]||\cdots||A_w^{(1)}[v[w]])$. Let Ψ_{Y_1} be the set of $\{\psi_y\}$, and add y to the result set $X_1 \cap Y_1$ iff $\psi_x \in \Psi_{Y_1}$.

Figure 3: Private set intersection from protocol [21].

2.4. (Hamming) Correlation Robustness

Similar to protocol [6, 22, 33], our protocol is proved to be secure, relying on the correlation robustness property.

Definition II.1. For all $i \in [n]$, *the length of* a_i, b_i *is* ℓ, $\|b_i\|_H \geq d$, *with* s *being a random string of length* ℓ, *then* H *is* d*-Hamming correlation robust function with input length* ℓ, *the distribution is pseudorandom:*

$$H(a_1 \oplus [b_1 \cdot s]), \cdots, H(a_n \oplus [b_n \cdot s])$$

3. Our Protocol

3.1. Construction

P_1 and P_2 each have datasets $\{X_1, \cdots, X_n\}$ and $\{Y_1, \cdots, Y_n\}$, respectively. P_1 and P_2 set the maximum dataset sizes as $N_{X_{[1,n]}}$ and $N_{Y_{[1,n]}}$. Here, $N_{X_\mathrm{i}} = N_{Y_\mathrm{i}}$ for $i \in [1, n]$.

IDEAL FUNCTIONALITY $\mathcal{F}_{PSI}(X_{[1,n]}, Y_1)$
Input: Receiver input dataset Y_1 and sender input dataset streaming $\{X_1, \cdots, X_n\}$.
Output: Receiver obtains $X_i \cap Y_1$ for all $i \in [1, n]$.

Figure 4: Ideal functionality for streaming PSI.

The streaming PSI. The functionality of streaming PSI is in Figure 4, which means the sender's dataset streaming $\{X_1, X_2, \cdots, X_n\}$ can be matched with the receiver's dataset Y_1 in turn to obtain $X_i \cap Y_1$ for $i \in [1, n]$. Our purpose is making the OPRF from previous PSI be reused in subsequent PSI. We achieve it by making a small adjustment to the OPRF from protocol [21].

As for the OPRF in Section 2.3, P_1 has the key $C^{(1)}$, and P_2 has the PRF values Ψ_{Y_1}. In the $\mathcal{F}_{PSI}(X_1, Y_1)$, for each $x \in X_1$, P_1 computes $v = F_{k_1}(H_1(x))$ and sends its PRF value $\psi_x = H_2\left(C_1^{(1)}[v[1]] \parallel \cdots \parallel C_w^{(1)}[v[w]]\right)$ to P_2. P_2 compares ψ_x with its PRF values Ψ_{Y_1} to get the result. To calculate the $\mathcal{F}_{PSI}(X_2, Y_1)$, a natural idea for reusing key $C^{(1)}$ and PRF values Ψ_{Y_1} is: for each $x \in X_2$, P_1 computes $v = F_{k_1}(H_1(x))$ and sends its PRF value $\psi_x = H_2\left(C_1^{(1)}[v[1]] \parallel \cdots \parallel C_w^{(1)}[v[w]]\right)$ to P_2. P_2 compares ψ_x with its PRF values Ψ_{Y_1} to get the result. However, it may leak some additional information about the P_1 (the reason is put in Section 3.2). Fortunately, we can avoid it by adjusting the parameter m and w. A detailed analysis of the parameters is put in Section 3.2.

PROTOCOL $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$
Input: P_1, P_2 input dataset X_1, Y_1 and total size $N_{X_{[1,n]}}$, $N_{Y_{[1,n]}}$.
Output: $X_{[1,n]} \cap Y_1$ or $X_1 \cap Y_{[1,n]}$.

1) P_1 acts as the sender and P_2 acts as the receiver, executing step 1,2 of $\Pi_{CM\_PSI}(X_1, N_{Y_{[1,n]}})$ ($N_{Y_{[1,n]}}$ constrains the choice of m,w in Section 3.2). As a result, P_1 obtains the OPRF key $C^{(1)}$, P_2 obtains the OPRF values Ψ_{Y_1}.
   - For each $x \in X_1$, P_1 computes $v = F_{k_1}(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]]|| \cdots ||C_w^{(1)}[v[w]])$ and send ψ_x to P_2. Then P_2 add y to the result set $X_1 \cap Y_1$ iff $\psi_x \in \Psi_{Y_1}$.
   - In later PSI, for $i \in [2, n]$ and $x \in X_i$, P_1 computes $v = F_{k_1}(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]]|| \cdots ||C_w^{(1)}[v[w]])$ and send ψ_x to P_2. Then P_2 add y to the result set $X_i \cap Y_1$ iff $\psi_x \in \Psi_{Y_1}$.
2) P_2 acts as the sender and P_1 acts as the receiver, executing step 1,2 of $\Pi_{CM\_PSI}(Y_1, N_{X_{[1,n]}})$ ($N_{X_{[1,n]}}$ constrains the choice of m,w in Section 3.2). As a result, P_2 obtains the OPRF key $C^{(2)}$, P_1 obtains the OPRF values Ψ_{X_1}.
   - For each $y \in Y_1$, P_2 computes $v = F_{k_2}(H_1(y))$ and its OPRF value $\psi_y = H_2(C_1^{(2)}[v[1]]|| \cdots ||C_w^{(2)}[v[w]])$ and send ψ_y to P_1. Then P_1 add x to the result set $Y_1 \cap X_1$ iff $\psi_y \in \Psi_{X_1}$.
   - In later PSI, for $i \in [2, n]$ and $y \in Y_i$, P_2 computes $v = F_{k_2}(H_1(y))$ and its OPRF value $\psi_y = H_2(C_1^{(2)}[v[1]]|| \cdots ||C_w^{(2)}[v[w]])$ and send ψ_y to P_1. Then P_1 add x to the result set $Y_i \cap X_1$ iff $\psi_y \in \Psi_{X_1}$.

Figure 5: The design for our streaming PSI.

Input: P_1, P_2 input datasets $\{X_1, X_2, \cdots, X_n\}$, $\{Y_1, Y_2, \cdots, Y_n\}$.
Output: $X_{[1,i]} \cap Y_{[1,i]}$, for $i \in [1, n]$
P_1 and P_2 agree on parameters $\lambda, \sigma, m, w, \ell_1, \ell_2, k_1, k_2, H_1 : \{0,1\}^* \to \{0,1\}^{\ell_1}, H_2 : \{0,1\}^w \to \{0,1\}^{\ell_2}$, pseudorandom function $F : \{0,1\}^{\ell_1} \times \{0,1\}^\lambda \to \{m\}^w$. We assume $N_{X_{[2,a-1]}} \& N_{Y_{[2,a-1]}} < th$ and $N_{X_{[2,a]}} \& N_{Y_{[2,a]}} \geq th$.

1) The initial PSI
   - Calculate $\mathcal{F}_{PSI}(X_1, Y_1)$.
     – P_1 and P_2 invoke $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ to get the result set $X_1 \cap Y_1$. After it, P_1 have matrix $C^{(1)}$ and OPRF set Ψ_{X_1}. P_2 have matrix $C^{(2)}$ and OPRF set Ψ_{Y_1}.
2) The (a)-th PSI ($a > 1$)
   - if $N_{X_{[1,a]}} > N_{X_{[1,n]}}$ or $N_{Y_{[1,a]}} > N_{Y_{[1,n]}}$, the protocol is terminated, otherwise, proceed as follows:
     – Calculate $\mathcal{F}_{PSI}(X_a, Y_1)$.
       * For each $x \in X_a$, P_1 computes $v = F_{k_1}(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]]|| \cdots ||C_w^{(1)}[v[w]]$ and send ψ_x to P_2.
       * Let Ψ be the set of OPRF values received from P_1. P_2 compares Ψ to Ψ_{Y_1} and gets PSI result $X_a \cap Y_1$.
     – Calculate $\mathcal{F}_{PSI}(X_1, Y_a)$.
       * For each $y \in Y_a$, P_2 computes $v = F_{k_2}(H_1(y))$ and its OPRF value $\psi_y = H_2(C_1^{(2)}[v[1]]|| \cdots ||C_w^{(2)}[v[w]]$ and send ψ_y to P_1.
       * Let Ψ be the set of OPRF values received from P_2. P_1 compares Ψ to Ψ_{X_1} and gets PSI result $X_1 \cap Y_a$.
     – Calculate $\mathcal{F}_{PSI}(X_{[2,a]}, Y_{[2,a]})$ and Reconstruct our final result.
       * If $N_{X_{[2,a]}} \& N_{Y_{[2,a]}} < th$, to calculate $\Pi_{CM\_PSI}(X_{[2,a]}, Y_{[2,a]})$. Else to calculate $\Pi_{LC\_PSI}(X_{[2,a]}, N_{X_{[2,n]}}, Y_{[2,a]}, N_{Y_{[2,n]}})$. Then P_1 have matrix $C^{(3)}$ and set $\Psi_{X_{[2,a]}}$. P_2 have matrix $C^{(4)}$ and set $\Psi_{Y_{[2,a]}}$.
       * P_1 and P_2 reconstruct the final result $X_{[1,a]} \cap Y_{[1,a]}$ by the above results.

Figure 6: The design for our multiple PSI.

After choosing proper parameters, the key $C^{(1)}$ and values Ψ_{Y_1} can be reused. And we can get the streaming PSI $\mathcal{F}_{PSI}(X_i, Y_1)$ easily, for $i \in [1, n]$. Correspondingly, when P_1 is the receiver with input X_1, P_2 is the sender with input streaming Y_i, for $i \in [1, n]$, we can get the reusable $C^{(2)}$ and Ψ_{X_1}, and compute the other PSI $\mathcal{F}_{PSI}(X_1, Y_i)$ easily. Finally, we can get two streaming PSI, which reduce the computational and communication overhead significantly. We denote it as $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ and show it in Figure 5. In the second PSI, since $\mathcal{F}_{PSI}(X_1, Y_2)$ and $\mathcal{F}_{PSI}(X_2, Y_1)$ are low-cost, the main overhead is related to $\mathcal{F}_{PSI}(X_2, Y_2)$, which is linearly related to the newly added dataset size N_{X_2} and N_{Y_2}. And in the i-th PSI, both parties can calculate $\mathcal{F}_{PSI}(X_1, Y_i)$ and $\mathcal{F}_{PSI}(X_i, Y_1)$ in low-cost.

Our multiple PSI. We use the streaming PSI to construct our multiple PSI. Since $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ makes $\mathcal{F}_{PSI}(X_1, Y_i)$ and $\mathcal{F}_{PSI}(X_i, Y_1)$ low-cost for each $i \in [2, n]$. It is naturally thought that we execute the $\Pi_{LC\_PSI}(X_j, N_{X_{[j,n]}}, Y_j, N_{Y_{[j,n]}})$ in the j-th PSI for each $j \in [2, n]$, then $\mathcal{F}_{PSI}(X_j, Y_i)$ and $\mathcal{F}_{PSI}(X_i, Y_j)$ would be low-cost for each $i \in [j+1, n]$. However, due to its own considerable overhead, a number of Π_{LC_PSI} also cause large overhead. And to prevent this, we execute the Π_{LC_PSI} whenever the accumulated dataset reaches the threshold th. It means our protocol will generate $\lfloor 1 + \frac{N_{X_{[2,n]}}}{th} \rfloor$ or $\lfloor 1 + \frac{N_{Y_{[2,n]}}}{th} \rfloor$ Π_{LC_PSI}.

The full description of our multiple PSI is in Figure 6. In the initial PSI, P_1 and P_2 obtain $X_1 \cap Y_1$, $C^{(1)}$, Ψ_{X_1}, $C^{(2)}$ and Ψ_{Y_1}. In the a-th PSI (a>1), in order to get $\mathcal{F}_{PSI}(X_{[1,a]}, Y_{[1,a]})$, they need to calculate $\mathcal{F}_{PSI}(X_a, Y_1)$, $\mathcal{F}_{PSI}(X_a, Y_{[2,a]})$, $\mathcal{F}_{PSI}(X_1, Y_a)$, $\mathcal{F}_{PSI}(X_{[2,a]}, Y_a)$ (remember $\mathcal{F}_{PSI}(X_{[1,a-1]}, Y_{[1,a-1]})$ is calculated in $(a-1)$-th PSI). In order to reduce the number of PSI, we merge $\mathcal{F}_{PSI}(X_a, Y_{[2,a]})$ with $\mathcal{F}_{PSI}(X_{[2,a]}, Y_a)$ into $\mathcal{F}_{PSI}(X_{[2,a]}, Y_{[2,a]})$. Since $\mathcal{F}_{PSI}(X_a, Y_1)$ and $\mathcal{F}_{PSI}(X_1, Y_a)$ are low-cost, then the primary cost lies in the $\mathcal{F}_{PSI}(X_{[2,a]}, Y_{[2,a]})$. When $N_{X_{[2,a]}} \& N_{Y_{[2,a]}} < th$, the process of $\mathcal{F}_{PSI}(X_{[2,a]}, Y_{[2,a]})$ is $\Pi_{CM\_PSI}(X_{[2,a]}, Y_{[2,a]})$; otherwise, it is $\Pi_{LC\_PSI}(X_{[2,a]}, N_{X_{[2,n]}}, Y_{[2,a]}, N_{Y_{[2,n]}})$, then the subsequent PSIs can significantly reduce the protocol overhead by utilizing reusable matrices $C^{(3)}, C^{(4)}$, $\Psi_{X_{[2,a]}}$ and $\Psi_{Y_{[2,a]}}$. Note: the maximum dataset size for P_1 and P_2 is limited to$N_{X_{[1,n]}}$ and $N_{Y_{[1,n]}}$ respectively, if these limits are exceeded, the protocol will be terminated.

3.2. Parameter analysis

Choice of m, w. Let the current dataset sizes of P_1 and P_2 be N_{X_1}, N_{Y_1}, and the total sizes of estimated dataset are $N_{X_{[1,n]}}, N_{Y_{[1,n]}}$. Our purpose is to choose proper parameters m, w to make that no less than d 1 appear in $D_1^{(1)}[v[1]], \cdots, D_w^{(1)}[v[w]]$ for $x \in X_{[1,n]} \backslash I$ and $v = F(H_1(x))$, then the value ψ_x is pseudorandom (in Section 2.4). We now discuss how to achieve it.

Initially, the $m \times w$ matrix D is set to all 1's. For each element from the set Y_1, a pseudorandom function generates w random positions, denoted as$\{l_1, \ldots, l_{\mathrm{w}}\}$, and sets $D_1[l_{\mathrm{j}}] = 0$ for $j \in [1, w]$. This results in:

$$p = Pr\left[D_i^{(1)}[j] = 1\right] = \left(1 - \frac{1}{m}\right)^{N_{Y_1}}.$$

Subsequently, when an element $x \notin Y_1$, the probability that k 1s appear in matrix D is denoted as:

$$\binom{w}{k} p^k (1-p)^{w-k}.$$

In function $\Pi_{CM\_PSI}(X_1, Y_1)$, it must satisfy the condition that, for all $x \in X_1 \backslash I$, the probability of fewer than d 1s appearing in the matrix D is negligible. We can get:

$$N_{X_1} \cdot \sum_{k=0}^{d-1} \binom{w}{k} p^k (1-p)^{w-k} \leq negl(\sigma).$$

Then we can derive a proper w in protocol $\Pi_{CM\_PSI}(X_1, Y_1)$ when m is fixed. Obviously, if the dataset size is bigger than N_{X_1} from the P_1, then the probability is non-negligible for the element x to be derived.

In function $\Pi_{LC\_PSI}\left(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}}\right)$, we should make the streaming PSI $\mathcal{F}_{PSI}\left(X_{[1,n]}, Y_1\right)$ secure (and the streaming PSI $\mathcal{F}_{PSI}\left(X_1, Y_{[1,n]}\right)$ is same with this case). It must satisfy the condition that, for all $x \in X_{[1,n]} \backslash I$, the probability of fewer than d 1s appearing in the matrix D is negligible. We can get:

$$N_{X_{[1,n]}} \cdot \sum_{k=0}^{d-1} \binom{w}{k} p^k (1-p)^{w-k} \leq negl(\sigma).$$

Then we can derive a proper w in protocol $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ when m is fixed.

Choice of ℓ_1. To ensure the hash function H_1 resists collision and birthday attacks, its output length is set to $\ell_1 = 2\lambda$, and λ is a computational security parameter.

Choice of ℓ_2. To ensure the hash function H_2 resists collision, the output length ℓ_2 can be computed as $\ell_2 = \sigma + log\left(N_{X_1} N_{Y_1}\right)$ in function $\Pi_{CM\_PSI}(X_1, Y_1)$. And in function $\Pi_{LC\_PSI}\left(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}}\right)$, it can be calculated as $\ell_2 = \sigma + log\left(N_{X_1} N_{Y_{[1,n]}}\right)$ or $\sigma + log\left(N_{X_{[1,n]}} N_{Y_1}\right)$.

Choice of th. We set the size of data accumulation to a certain size th, and execute the function Π_{LC_PSI} for the accumulated dataset. In fact, if the value of th is too small, it will add a lot of streaming PSI, which will also bring a large overhead. If the value of th is too big, the overhead accumulated by subsequent PSI will not be reduced. We recommend that the th make a compromise between the new dataset size and the total estimated dataset size.

For instance, if the size of each added dataset is 2^{16}, and the size of estimated total dataset is up to 2^{24}, then we can set th to $2^{19}, 2^{20}, 2^{21}$.

Choice of $N_{X_{[1,n]}}, N_{Y_{[1,n]}}$. If the total size of accumulated dataset exceeds $N_{X_{[1,n]}}$ and $N_{Y_{[1,n]}}$, the protocol will be terminated. In practice, we need a bigger size for $N_{X_{[1,n]}}$ and $N_{Y_{[1,n]}}$ to keep protocol efficient and secure.

For instance, two companies need to cooperate on a project, and multiple PSI requests are required. Then two companies evaluate the upper limits of their dataset size $N_{X_{[1,n]}}'$ and $N_{Y_{[1,n]}}'$ in advance, and set two size factors u_1 and u_2 to an appropriate value, such as 1.5, 2, 2.5, etc. Finally, the total size can be set as follow:

$$N_{X_{[1,n]}} = N_{X_{[1,n]}}' \cdot u_1, N_{Y_{[1,n]}} = N_{Y_{[1,n]}}' \cdot u_2$$

3.3. Security proof

We defer the security proof of protocol in Figure 5 to Appendix A, defer the security proof of protocol in Figure 6 to Appendix B, and only state the theorem below.

Theorem III.1. If protocol Π_{CM_PSI} in Figure 3 is secure in the semi-honest model and parameters m, w, ℓ_1, ℓ_2 satisfy the constraints in Section 3.2, then, the protocol Π_{LC_PSI} is proven to be secure in the semi-honest model, as shown in Figure 5.

Theorem III.2. If protocols Π_{CM_PSI} and Π_{LC_PSI} in Figure 3 and 5 is secure in the semi-honest model, parameters m, w, ℓ_1, ℓ_2 satisfy the constraints in Section 3.2, then, the protocol is proven to be secure in the semi-honest model, as shown in Figure 6.

4. Other improvements

4.1. Privacy-enhancing

In fact, the location information of the element from intersection can also be sensitive. For instance, in the second PSI, participants know three intersections: $\mathcal{F}_{PSI}(X_1, Y_2)$, $\mathcal{F}_{PSI}(X_2, Y_1)$, and $\mathcal{F}_{PSI}(X_2, Y_2)$. For $x \in X_2$, P_1 can know whether the intersection element x exists in the set Y_1 or Y_2. For $y \in Y_2$, P_2 can know whether the intersection element y exists in the set X_1 or X_2. But in practice, it may also be sensitive information [29]. In order to prevent this information from being leaked, we should make P_1 only know $\mathcal{F}_{PSI}(X_1, Y_2)$ and $\mathcal{F}_{PSI}(X_2, Y_1 \cup Y_2)$, and P_2 only know $\mathcal{F}_{PSI}(X_2, Y_1)$ and $\mathcal{F}_{PSI}(X_1 \cup X_2, Y_2)$. Figure 7 shows our strategy to achieve it.

Assuming that P_1 is the sender and P_2 is the receiver, P_2 needs to get the intersection of $\mathcal{F}_{PSI}(X_1 \cup X_2, Y_2)$ securely. After the initial PSI, both parties execute $\mathcal{F}_{PSI}(Y_2, X_1)$, P_1 obtains the intersection $X_I = \{x_{old,1}, \cdots\}$, and then P_1 inserts the intersection X_I into the X_2, updates $\{X_2, X_3, \cdots\}$ to $\{X'_2, X'_3, \cdots\}$.

Input: P_1, P_2 input datasets $\{X_1, X_2, \cdots, X_n\}$, $\{Y_1, Y_2, \cdots, Y_n\}$.
Output: $X_{[1,i]} \cap Y_{[1,i]}$, for $i \in [2, n]$. In here, $i = 2$.
After the initial PSI, P_1 has matrix $C^{(1)}$, dataset $X_1 := \{x_{old,1}, x_{old,2}, \cdots\}$ and OPRF set Ψ_{X_1}. P_2 has matrix $C^{(2)}$,dataset $Y_1 := \{y_{old,1}, y_{old,2}, \cdots\}$ and OPRF set Ψ_{Y_1}. And P_1 and P_2 agree on parameters $\lambda, \sigma, m, w, \ell_1, \ell_2, k_1, k_2, H_1 : \{0,1\}^* \rightarrow \{0,1\}^{\ell_1}, H_2 : \{0,1\}^w \rightarrow \{0,1\}^{\ell_2}$, pseudorandom function $F : \{0,1\}^{\ell_1} \times \{0,1\}^{\lambda} \rightarrow \{m\}^w$. P_1 add new dataset $\{X_2, X_3, \cdots\}$ and P_2 add new dataset $\{Y_2, Y_3, \cdots\}$. Note: $\{X_2, X_3, \cdots; Y_2, Y_3, \cdots\} = \{(x_1, \cdots, x_t), (x_{t+1}, \cdots, x_{2t}), \cdots; (y_1, \cdots, y_t), (y_{t+1}, \cdots, y_{2t}), \cdots\}$. And we assume P_1 is the sender and P_2 is the receiver. $N_{X_2} \& N_{Y_2} < th$.

1) The next PSI
- Calculate $\mathcal{F}_{PSI}(X_1, Y_2)$ and update.
  - For each $y \in Y_2$, P_2 computes $v = F_{k_2}(H_1(y))$ and its OPRF value $\psi_y = H_2(C_1^{(2)}[v[1]] || \cdots || C_w^{(2)}[v[w]]$ and send ψ_y to P_1.
  - Let Ψ be the set of OPRF values received from P_2. P_1 compares Ψ to Ψ_{X_1} and gets PSI result X_I. And we assume the intersection dataset is $X_I := \{x_{old,1}, \cdots, x_{old,k}\}$.
  - P_1 update the $\{X'_2, X'_3, \cdots\} = \{(x_{old,1}, \cdots, x_{old,k}, x_1, \cdots, x_{t-k}), (x_{t-k+1}, \cdots, x_{2t-k}), \cdots\}$.
  - P_1 update the $X''_2 = \{x_1, \cdots, x_{t-k}\} \cup X_D$, X_D consists of dummy random elements and $|X_D| = k$.
- Calculate $\mathcal{F}_{PSI}(X''_2, Y_1)$.
  - For each $x \in X''_2$, P_1 computes $v = F_{k_1}(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]] || \cdots || C_w^{(1)}[v[w]]$ and send ψ_x to P_2.
  - Let Ψ be the set of OPRF values received from P_1. P_2 compares Ψ to Ψ_{Y_1} and gets the intersection dataset Y_I.
- Calculate $\mathcal{F}_{PSI}(X'_2, Y_2)$ and Output.
  - Because $N_{X'_2} \& N_{Y_2} < th$, to execute $\Pi_{CM\_PSI}(X'_2, Y_2)$. P_2 get the intersection dataset Y'_I.
  - Reconstruct PSI result $(X_1 \cup X'_2) \cap Y_{[1,2]}$, and send $Y_I \cup Y'_I$ to P_1.

Figure 7: Privacy-enhancing strategy.

We cannot execute $\mathcal{F}_{PSI}(X'_2, Y_1)$ directly, it will leak the number of $\mathcal{F}_{PSI}(Y_2, X_1)$. If we directly execute $\mathcal{F}_{PSI}(X'_2, Y_1)$, X_I will perform the following operations: for each $x \in X_I$, P_1 computes $v = F_{k_1}(H_1(x))$ and its OPRF value $\psi_x = H_2(C_1^{(1)}[v[1]] \parallel \cdots \parallel C_w^{(1)}[v[w]])$ and sends ψ_x to P_2. This process has also appeared in previous PSI, which is $\mathcal{F}_{PSI}(X_1, Y_1)$. Therefore, P_2 can get the two same ψ_x, and P_2 can know the number of $\mathcal{F}_{PSI}(Y_2, X_1)$.

To avoid additional information leakage, we generate a set of random values X_D, $|X_D| = |X_I|$, then update $X''_2 = X'_2 \backslash X_I \cup X_D$. Then both parties run the $\mathcal{F}_{PSI}(X''_2, Y_1)$, P_2 gets the intersection Y_I. Finally, P_1 acts as the sender, P_2 acts as the receiver, and both run $\Pi_{CM\_PSI}(X_2', Y_2)$. P_2 get the intersection $(X_1 \cup X'_2) \cap Y_2$, which is dataset Y_I'. For $y \in Y_I'$, P_2 can not know whether the intersection element y exists in set X_1 or $\{X'_2 \backslash X_I\}$. And the same as P_1. Furthermore, through the same method, we can get the privacy-enhancing $\mathcal{F}_{PSI}\left(\cup_{j=1}^{i} X_j, \cup_{j=1}^{i} Y_j\right)$ in the i-th PSI, for $i > 2$.

4.2. Generalizability

Our idea is applicable to other PSI protocols, and makes these protocols suitable for multiple PSI scenario. In theory, we can construct the streaming PSI from OPRF-based protocols [12, 13, 22], then the sender utilizes the reusable OPRF key and the receiver utilizes the reusable OPRF values to construct multiple PSI protocol. In Figure 8, we provide a generic framework for applying our idea to other protocols.

Input: P_1, P_2 input datasets $\{X_1, X_2, \cdots, X_n\}$, $\{Y_1, Y_2, \cdots, Y_n\}$.
Output: $X_{[1,i]} \cap Y_{[1,i]}$, for $i \in [1, n]$.

1) The initial PSI
   - P_1 and P_2 invoke the streaming PSI to get the result of $\mathcal{F}_{PSI}(X_1, Y_1)$. After it, P_1 has OPRF key κ_1 and OPRF set Ψ_{X_1}. P_2 have OPRF key κ_2 and OPRF set Ψ_{Y_1}.
2) The (a)-th PSI
   - if $N_{X_{[1,a]}} > N_{X_{[1,n]}}$ or $N_{Y_{[1,a]}} > N_{Y_{[1,n]}}$, the protocol is terminated, otherwise, proceed as follows:
     - For dataset X_a, P_1 computes their OPRF value Ψ_{X_a} by key κ_1, and P_2 compares Ψ_{X_a} with Ψ_{Y_1} to get the result of $\mathcal{F}_{PSI}(X_a, Y_1)$.
     - For dataset Y_a, P_2 computes their OPRF value Ψ_{Y_a} by key κ_2, and P_1 compares Ψ_{Y_a} with Ψ_{X_1} to get the result of $\mathcal{F}_{PSI}(X_1, Y_a)$.
     - If $N_{X_{[2,a]}} \& N_{Y_{[2,a]}} < th$, to calculate the result of $\mathcal{F}_{PSI}(X_{[2,a]}, Y_{[2,a]})$ with original protocol. Else to invoke two streaming PSI $\mathcal{F}_{PSI}(X_{[2,a]}, N_{Y_{[2,n]}})$ and $\mathcal{F}_{PSI}(N_{X_{[2,n]}}, Y_{[2,a]})$.
     - Reconstruct the result of $\mathcal{F}_{PSI}(X_{[1,a]}, Y_{[1,a]})$.

Figure 8: A generic framework for our idea.

5. Performance Evaluation

5.1. Preparation

Implement. We tested our protocol using C++ on a computer equipped with a Ryzen 7 5800H processor (3.2 GHz), 8 physical cores, and 16 GB of RAM. The benchmarks were conducted in a local area network (LAN) environment with a 1 Gbps connection and sub-millisecond latency. Our code is implemented at https://github.com/GreenEli/Multiple_PSI.

Parameters. In our experimental setup, we set the computational and statistical security parameter $\lambda = 128$ and $\sigma = 40$. $\ell_1 = 2\lambda = 256, d = \lambda = 128$. Additional parameters are detailed in Table 1. We choose $m = N_{X_1} = N_{Y_1}$, as this choice nearly minimizes the communication overhead and also allows for optimal computational overhead.

Comparison protocols. We compare our PSI protocol with the current optimal protocols [22, 21, 12, 13], which are [KKRT16, CM20, RS21, RR22]. In our experiments, in order to allow these protocols to be applied to multiple PSI scenario, we use the two methods (*Protocol_All* and *Protocol_Split*) mentioned above.

Table 1: Parameters for set size N_{X_1}, N_{Y_1} matrix height m, matrix width w, and output length ℓ_2 of hash function H_2 for semi-honest security, and communication overhead of every bit.

Protocol	N_{X_1}&N_{Y_1}	$N_{X_{[1,n]}}$&$N_{Y_{[1,n]}}$	m	w	ℓ_2
$\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$	2^{16}	2^{17}	N_{X_1}&N_{Y_1}	612	63
	2^{16}	2^{20}		621	76
	2^{20}	2^{21}		624	81
	2^{20}	2^{24}		633	84
	2^{24}	2^{25}		636	89
	2^{24}	2^{28}		645	92
$\Pi_{CM\_PSI}(X_1, Y_1)$	2^{12}	–		597	64
	2^{16}	–		609	72
	2^{20}	–		621	80
	2^{24}	–		633	88

5.2. Evaluation

In this section, we describe the various stages of multiple PSI, including the initial PSI, the next PSI, and the subsequent PSI. And we make an analysis and comparison in each stage.

Initial PSI. In the initial PSI stage, we test the runtime and communication overhead of protocols [KKRT16, CM20, RS21, RR22] respectively. Among them, the dataset sizes for comparison are $2^{16}, 2^{20}, 2^{24}$, and the dataset sizes owned by the sender and receiver are $N_{X_1} = N_{Y_1} = N$. In our protocol, we set the total estimated dataset size of both parties as $N_{X_{[1,n]}} = N_{Y_{[1,n]}} = 2N$. The experimental results are shown in Table 2. Note: the parameter κ in the Table 2 is approximately 128, and the λ_1 is 40.

Table 2: Runtime and communication overhead comparison in initial PSI.

Protocol	Times (ms)			Comm. (bits)			Comm. Asymptotic (bits)
	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}	$N_{X_1} = N_{Y_1} = N, N_{X_{[1,n]}} = N_{Y_{[1,n]}} = 2N$
[KKRT16]	154	2189	5632	984N	1008N	1032N	$6\kappa N_{X_1} + 3(\lambda_1 + \log(N_{X_1}N_{Y_1}))N_{Y_1}$
[CM20]	468	6547	154111	681N	701N	721N	$4.8\kappa N_{X_1} + (\lambda_1 + \log(N_{X_1}N_{Y_1}))N_{Y_1}$
[RS21]	511	4891	116795	960N	426N	398N	$2.4\kappa N_{X_1} + (\lambda_1 + \log(N_{X_1}N_{Y_1}))N_{Y_1} + 2^{17}\kappa {N_{X_1}}^{0.05}$
[RR22]	**75**	**1425**	**28401**	**206N**	**180N**	**196N**	$1.2\log(N_{X_1}N_{Y_1})N_{X_1} + (\lambda_1 + \log(N_{X_1}N_{Y_1}))N_{Y_1} + 2^{14.5}\kappa$
Ours	913	13861	297658	1370N	1410N	1450N	$4.8\kappa(N_{X_1} + N_{Y_1}) + (\lambda_1 + \log(N_{X_1}N_{Y_{[1,n]}}))N_{Y_1} + (\lambda_1 + \log(N_{X_{[1,n]}}N_{Y_1}))N_{X_1}$

Overall evaluation. According to Table 2, in terms of initial PSI, our runtime and communication overhead are bigger than other protocols, which is twice the runtime and communication overhead of the protocol CM20.

Next PSI. After the initial PSI, both parties will generate new PSI requirements as the dataset size grows. We set the dataset sizes of both sides as 2^{20} and 2^{24} in the initial PSI, and the new dataset sizes of both sides are $2^{16}, 2^{20}$ and 2^{24} in the next PSI. We test the runtime and communication overhead of compared protocols in two naive ways (*Protocol_All* and *Protocol_Split*), and finally, we get Table 3.

Table 3: Runtime and communication overhead comparison in next PSI.

Protocol		$(N_{X_1}+N_{X_2}, N_{Y_1}+N_{Y_2})$		$(N_{X_1}+N_{X_2}, N_{Y_1}+N_{Y_2})$			Comm. asymptotic (bits)
		$(2^{20}+2^{16}, 2^{20}+2^{16})$	$(2^{20}+2^{20}, 2^{20}+2^{20})$	$(2^{24}+2^{16}, 2^{24}+2^{16})$	$(2^{24}+2^{20}, 2^{24}+2^{20})$	$(2^{24}+2^{24}, 2^{24}+2^{24})$	**Next PSI data size: $(N_{X_1}+N_{X_2}, N_{Y_1}+N_{Y_2})$ $(N=N_{X_1}=N_{Y_1})$,$(N_1=N_{X_2}=N_{Y_2})$, $(N_{X_{[1,n]}}=N_{Y_{[1,n]}}=2N)$**
[KKRT16]-Split	Comm./bits	$9060N_1$	$2256N_1$	$124656N_1$	$9852N_1$	$2328N_1$	$(6\kappa N_{X_2}+3(\lambda_1+\log(N_{X_2}N_{Y_1}))N_{Y_1})+(6\kappa N_{Y_2}+3\left(\lambda_1+\log\left(N_{Y_2}(N_{X_1}+\right.\right.$
	Time/ms	3024	9342	39743	52569	146879	$\left.\left.N_{X_2})\right)\right)(N_{X_1}+N_{X_2}))$
[KKRT16]-All	Comm./bits	$17136N_1$	$2028N_1$	$265224N_1$	$17544N_1$	$2076N_1$	$6\kappa(N_{X_1}+N_{X_2})+3\left(\lambda_1+\log\left((N_{X_1}+N_{X_2})(N_{Y_1}+N_{Y_2})\right)\right)$
	Time/ms	4356	8099	65553	73298	134237	$(N_{Y_1}+N_{Y_2})$
[CM20]-Split	Comm./bits	$3751N_1$	$1491N_1$	$42306N_1$	$4038N_1$	$1530N_1$	$(4.8\kappa N_{X_2}+(\lambda_1+\log(N_{X_2}N_{Y_1}))N_{Y_1})+(4.8\kappa N_{Y_2}+\left(\lambda_1+\log\left(N_{Y_2}(N_{X_1}+\right.\right.$
	Time/ms	8576	13205	1451236	160329	368874	$\left.\left.N_{X_2})\right)\right)(N_{X_1}+N_{X_2}))$
[CM20] -All	Comm./bits	$11798N_1$	$1394N_1$	$181956N_1$	$11934N_1$	$1408N_1$	$4.8\kappa(N_{X_1}+N_{X_2})+\left(\lambda_1+\log\left((N_{X_1}+N_{X_2})(N_{Y_1}+N_{Y_2})\right)\right)$
	Time/ms	6399	13345	151011	161348	335358	$(N_{Y_1}+N_{Y_2})$
[RS21] -Split	Comm./bits	$4060N_1$	$932N_1$	$42724N_1$	$3464N_1$	$884N_1$	$2.4\kappa N_{X_2}+(\lambda_1+\log(N_{X_2}N_{Y_1}))N_{Y_1}+2^{17}\kappa {N_{X_2}}^{0.05}+2.4\kappa N_{Y_2}+\left(\lambda_1+\right.$
	Time/ms	2845	10642	52061	60369	231867	$\left.\log\left(N_{Y_2}(N_{X_1}+N_{X_2})\right)\right)(N_{X_1}+N_{X_2})+2^{17}\kappa {N_{Y_2}}^{0.05}$
[RS21] -All	Comm./bits	$7242N_1$	$852N_1$	$102286N_1$	$6766N_1$	$796N_1$	$2.4\kappa(N_{X_1}+N_{X_2})+\left(\lambda_1+\log\left((N_{X_1}+N_{X_2})(N_{Y_1}+N_{Y_2})\right)\right)(N_{Y_1}+N_{Y_2})+$
	Time/ms	4398	10325	110586	114758	247112	$2^{17}\kappa(N_{X_1}+N_{X_2})^{0.05}$
[RR22] -Split	Comm./bits	$2784N_1$	$442N_1$	$41326N_1$	$2984N_1$	$479N_1$	$1.2\log(N_{X_2}Y_1)N_{X_2}+(\lambda_1+\log(N_{X_2}N_{Y_1}))N_{Y_1}+1.2\log\left(N_{Y_2}(N_{X_1}+N_{X_2})\right)N_{Y_2}$
	Time/ms	**657**	**2995**	12372	**15589**	**58429**	$+2^{15.5}\kappa+\left(\lambda_1+\log\left(N_{Y_2}(N_{X_1}+N_{X_2})\right)\right)(N_{X_1}+N_{X_2})$
[RR22] -All	Comm./bits	$3060N_1$	$\mathbf{360N_1}$	$50372N_1$	$3332N_1$	$\mathbf{392N_1}$	$1.2\log\left((N_{X_1}+N_{X_2})(N_{Y_1}+N_{Y_2})\right)(N_{X_1}+N_{X_2})$
	Time/ms	1323	3081	27856	31157	59329	$+\left(\lambda_1+\log\left((N_{X_1}+N_{X_2})(N_{Y_1}+N_{Y_2})\right)\right)(N_{Y_1}+N_{Y_2})+2^{14.5}\kappa$
Ours	Comm./bits	$\mathbf{843N_1}$	$863N_1$	$\mathbf{859N_1}$	$\mathbf{879N_1}$	$899N_1$	$4.8\kappa N_{X_2}+(\lambda_1+\log(N_{X_2}N_{Y_2}))N_{Y_2}+(\lambda_1+\log(N_{X_{[1,n]}}N_{Y_1}))N_{X_2}+(\lambda_1+$
	Time/ms	988	14597	**4685**	18123	300954	$\log(N_{Y_{[1,n]}}N_{X_1}))N_{Y_2}$

Overall evaluation. According to Table 3, in terms of communication and runtime, due to the streaming PSI we designed, the communication overhead and runtime of $\mathcal{F}_{PSI}(X_2, Y_1)$ and $\mathcal{F}_{PSI}(X_1, Y_2)$ are low-cost (which doesn't need to generate a new OPRF, but uses the OPRF of previous PSI directly), the overall overhead of the second PSI is mainly related to $\mathcal{F}_{PSI}(X_2, Y_2)$. For instance, when the initial dataset size is 2^{24} and the second dataset size is 2^{16}, the communication overhead of our protocol exceeds that of the current optimal protocol RR22 by about 50×, and the runtime exceeds that of the protocol RR22 by about 2.6×. It is clear from the table that the larger the dataset size gap between initial dataset and next dataset, the bigger advantages of our protocol in communication overhead and runtime.

In addition, we find in the protocol [KKRT16, CM20, RS21, RR22], when the gap is obvious between the new dataset size and the initial dataset size, the overall overhead from the method of *Protocol_Split* is significantly lower than the overall overhead from the method of *Protocol_All*. And when the gap is not obvious, their overall overhead is close.

Multiple PSI. We simulate the multiple PSI processes. The initial PSI dataset size for both parties is set to 2^{24}, with a total estimated dataset size of 2^{25}. Subsequent newly added dataset sizes are 2^{12}, 2^{16} and 2^{20}, and the threshold is set to $th = 2^{16}$, 2^{20} and 2^{22}. When both parties execute the $16th$ or $4th$ new PSI, our protocol run the function Π_{LC_PSI}. Due to the better performance of the method of *Protocol_Split*, the protocols [KKRT16, CM20, RS21, RR22] are compared with our protocol by the method of *Protocol_Split* in the subsequent PSI, and the following Figure 9(a),(b),(c),(d),(e),(f) is obtained.

On the other hand, we count the sum of the runtime and communication overhead from all PSI (including initial PSI and subsequent PSI), and get the runtime and communication overhead in the entire PSI process, as shown in Figure 9(g),(h),(i),(j),(k),(l).

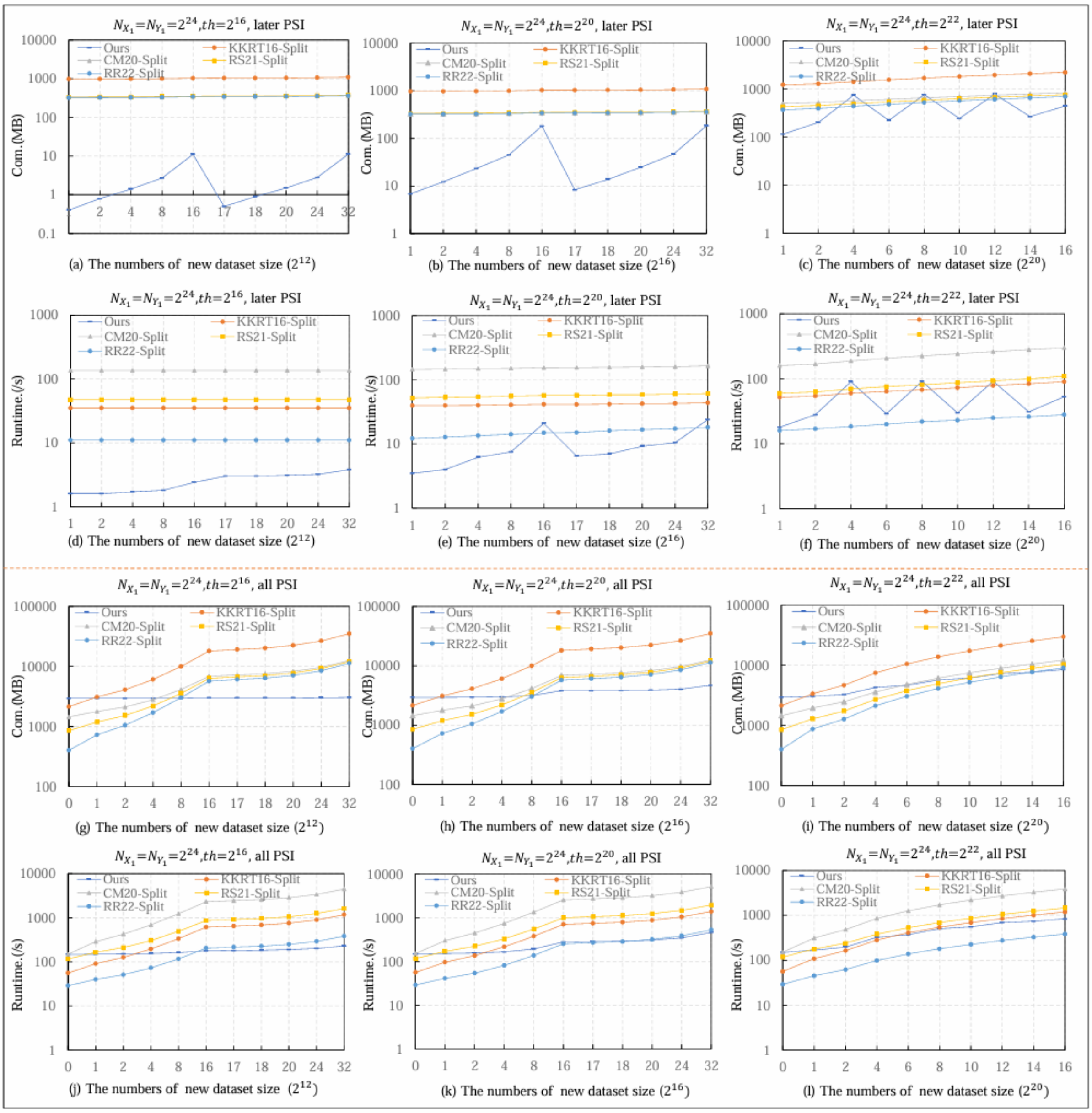

Figure 9: Runtime and communication overhead comparison in multiple PSI.

Overall evaluation. As shown in Figure 9(a),(b),(c),(d),(e),(f), on the whole, the communication overhead and runtime of our protocol are significantly lower than other protocols at the beginning. With the accumulation of new dataset, the communication overhead and runtime increase until the new dataset size reaches th. After executing Π_{LC_PSI}, the newly added dataset communication overhead and runtime decrease rapidly, the overall overhead is mainly related to newly added dataset size again. Our protocol has significant advantages over other protocols, and when the dataset scale gap is larger, the advantage of our protocol is bigger.

For instance, when the subsequent new dataset size is 2^{12} and th is 2^{16}, the communication overhead of our protocol is between (0.44MB,11MB), the runtime is about 4s. While the minimum communication overhead of comparison protocol is about 320MB, and the minimum runtime is about 11s. The communication overhead is improved by about (29×,727×), and the runtime is improved by about 2.7×. When the subsequent new dataset size is 2^{16} and th is 2^{20}, the communication overhead of our protocol is between (6.8MB,183MB), the runtime is between (4.1s,24s). While the minimum communication overhead of comparison protocol is about 323MB, and the entire runtime of our protocol is better than other protocols. When the subsequent new dataset size is 2^{20} and th is 2^{22}, on the whole, the communication overhead of our protocol is the lowest, while the runtime is second only to protocol RR22.

As shown in Figure 9(g),(h),(i),(j),(k),(l), in the initial PSI, the communication overhead and runtime of our protocol are the highest (we only make a small optimization to the initial PSI runtime, we let two streaming PSI run in parallel, and the runtime can be reduced by one time). It can be seen from Figure 9(g),(h),(i) that our protocol has the least communication overhead at the 8th, 8th, and 14th PSI, and the subsequent PSI continues to expand our advantage of communication overhead. From Figure 9(j),(k) that our protocol has the least runtime at the 13th and 19th PSI, and the subsequent PSI continues to expand our advantage of communication overhead. In Figure 9(l), the overall runtime is second only to protocol RR22 in the end.

All in all, the experiments show that our protocol outperforms other protocols obviously, which has significant advantage in multiple PSI scenario. Note that when each new dataset size is 2^{16}, we can execute a total of 257 PSI, and when each new dataset size is 2^{12}, we can execute a total of 4097 PSI. If we apply our idea to protocol RR22, the performance will be improved significantly. Therefore, the application value of our protocol in multiple PSI scenario is obvious.

6. Conclusion

Overall, we introduce an innovative multiple PSI protocol built upon the reusable OPRF. In short, on the basis of OPRF (Chase et al., Crypto 2020), we design the streaming PSI, which makes the OPRF of the previous PSI be reused in the subsequent PSI. Then we use it as a block to construct our protocol which is well-suited for multiple PSI scenario. Additionally, we provide a privacy-enhanced version, along with a general framework applicable to other OPRF-based protocols.

In the end, our experiments show that when the dataset added by both parties tends to be high in frequency but limited in size, the performance of our protocol is optimal.

Appendix

A. Security proof for streaming PSI

To prove the security from protocol $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$, it is necessary to prove the security from streaming PSI $\mathcal{F}_{PSI}(X_1, Y_{[1,n]})$ and $\mathcal{F}_{PSI}(X_{[1,n]}, Y_1)$, which is equivalent to protocol $\Pi_{CM\_PSI}(X_1, Y_{[1,n]})$ and $\Pi_{CM\_PSI}(X_{[1,n]}, Y_1)$. The proof of this part can be found in protocol [21].

B. Security proof for multiple PSI

We assume that P_1 and P_2 have dataset $\{X_1, X_{[2,t+1]}, X_{[t+2,2t+1]}, \cdots\}$ and $\{Y_1, Y_{[2,t+1]}, Y_{[t+2,2t+1]}, \cdots\}$. They invoke a new Π_{LC_PSI} whenever t new datasets are added. Then in the $(at+b)$-*th* PSI (we set $a \geq 0, b \in [2, t+1]$, and $at+b$ is no bigger than n), they have dataset $X_{[1,at+b]}$ and $Y_{[1,at+b]}$.

Security against corrupt P_1. We construct a simulator $SimHy_b$ in the ideal world to simulate the view of adversary $\mathcal{A}$ that corrupts party P_1 in the real world, where the simulator's inputs are $(1^\lambda, X_{[1,at+b]}, N_{Y_{[1,n]}}, f(X_{[1,at+b]}, Y_{[1,at+b]}))$, with $f(X_{[1,at+b]}, Y_{[1,at+b]})$ representing the intersection $X_{[1,at+b]} \cap Y_{[1,at+b]}$ in the ideal world, and the simulator outputs the adversary's view.

a) Hyb_0: In the real world, the adversary $\mathcal{A}$ interacts with the honest P_2.
b) Hyb_1: In the initial PSI, the simulator $SimHy_b$ sends $f(X_1, Y_1)$ and $N_{Y_1} - |f(X_1, Y_1)|$random values to the adversary.
c) Hyb_2: In the $(at + b)$-th PSI, for each $j \in [0, a - 1]$, the simulator $SimHy_b$ sends $f(X_1, Y_{at+b})$, $f(X_{[jt+2,(j+1)t+1]}, Y_{at+b})$ and $N_{Y_{at+b}} - |f(X_1, Y_{at+b})|$ - $|f(X_{[jt+2,(j+1)t+1]}$, $Y_{at+b})|$ random values to the adversary.
d) Hyb_3: In the $(at + b)$-th PSI, the simulator $SimHy_b$ sends $f(X_{[at+2,at+b]}, Y_{[at+2,at+b]})$ and $N_{Y_{at+b}} - |f(X_{[at+2,at+b]}, Y_{[at+2,at+b]})|$ random values to the adversary.
e) Hyb_4: In the $(at + b)$-th PSI, for each $j \in [0, a - 1]$, the simulator $SimHy_b$ invokes $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ and Π_{LC_PSI} $(X_{[jt+2,(j+1)t+1]}$, $N_{X_{[jt+2,n]}}, Y_{[jt+2,(j+1)t+1]}, N_{Y_{[jt+2,n]}})$, sends $f(X_{at+b}, Y_1)$ and $f\left(X_{at+b}, Y_{[jt+2,(j+1)t+1]}\right)$ to $\mathcal{A}$, $\mathcal{A}$ reconstruct $X_{[1,at+b]} \cap Y_{[1,at+b]}$ by these ideal intersections.

Lemma A.1. The world of Hyb_0 can be simulated and is indistinguishable from the world in Hyb_1.

Proof. In Hyb_0, honest P_2 sends $\left(Y_1, N_{Y_{[1,n]}}\right)$, and $\mathcal{A}$ sends $\left(X_1, N_{X_{[1,n]}}\right)$ to Π_{LC_PSI}, return $X_1 \cap Y_1$ to $\mathcal{A}$. In Hyb_1, simulator $SimHy_b$ sends the value $I = f(X_1, Y_1)$ to adversary, which is the same with $X_1 \cap Y_1$ by the behaviour of Π_{LC_PSI}. Therefore, the world of Hyb_0 can be simulated and is indistinguishable from the world in Hyb_1.□

Lemma A.2. The world of Hyb_1 can be simulated and is indistinguishable from the world in Hyb_2.

Proof. In Hyb_1, for each $j \in [0, a - 1]$, honest P_2 sends input Y_{at+b}, and $\mathcal{A}$ sends X_{at+b} to $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ and protocol Π_{LC_PSI} $(X_{[jt+2,(j+1)t+1]}$, $N_{X_{[jt+2,n]}}, Y_{[jt+2,(j+1)t+1]}, N_{Y_{[jt+2,n]}})$, return $X_1 \cap Y_{at+b}$ and $X_{[jt+2,(j+1)t+1]} \cap Y_{at+b}$ to $\mathcal{A}$.

In Hyb_2, simulator $SimHy_b$ sends $f(X_1, Y_{at+b})$ and $f\left(X_{[jt+2,(j+1)t+1]}, Y_{at+b}\right)$ to $\mathcal{A}$, which is the same with $X_1 \cap Y_{at+b}$ and $X_{[jt+2,(j+1)t+1]} \cap Y_{at+b}$ by the behaviour of Π_{LC_PSI}. Therefore, the world of Hyb_1 can be simulated and is indistinguishable from the world in Hyb_2. □

Lemma A.3. The world of Hyb_2 can be simulated and is indistinguishable from the world in Hyb_3.

Proof. In Hyb_2, honest P_2 sends input $Y_{[at+2,at+b]}$, and $\mathcal{A}$ sends $X_{[at+2,at+b]}$ to to Π_{LC_PSI} or Π_{CM_PSI}, return $X_{[at+2,at+b]} \cap Y_{[at+2,at+b]}$ to $\mathcal{A}$.

In Hyb_3, simulator $SimHy_b$ sends $f\left(X_{[at+2,at+b]}, Y_{[at+2,at+b]}\right)$ to $\mathcal{A}$, which is the same with $X_{[at+2,at+b]} \cap Y_{[at+2,at+b]}$ by the behaviour of Π_{LC_PSI} or or Π_{CM_PSI}. Therefore, the world of Hyb_2 can be simulated and is indistinguishable from the world in Hyb_3.

Lemma A.4. The world of Hyb_3 can be simulated and is indistinguishable from the world in Hyb_4.

Proof. In hybrid 3, for each $j \in [0, a - 1]$, honest P_2 sends input Y_{at+b}, and $\mathcal{A}$ sends X_{at+b} to $\Pi_{LC\_PSI}(X_1, N_{X_{[1,n]}}, Y_1, N_{Y_{[1,n]}})$ and protocol Π_{LC_PSI} $(X_{[jt+2,(j+1)t+1]}$, $N_{X_{[jt+2,n]}}, Y_{[jt+2,(j+1)t+1]}, N_{Y_{[jt+2,n]}})$, return $X_{at+b} \cap Y_{[1,at+1]}$ to $\mathcal{A}$.

In Hyb_4, simulator $SimHy_b$ sends $f\left(X_{at+b}, Y_{[1,at+1]}\right)$ to $\mathcal{A}$, which is the same with $X_{at+b} \cap Y_{[1,at+1]}$ by the behaviour of Π_{LC_PSI}. Finally, $\mathcal{A}$ reconstruct $f\left(X_{[1,at+b]}, Y_{[1,at+b]}\right) = f\left(X_{[1,at+b-1]}, Y_{[1,at+b-1]}\right) \cup f\left(X_{[1,at+1]}, Y_{at+b}\right) \cup f(X_{at+b}, Y_{[1,at+1]}) \cup f\left(X_{[at+2,at+b]}, Y_{[at+2,at+b]}\right)$. which is the same with $X_{[1,at+b]} \cap Y_{[1,at+b]}$. Therefore, the world of Hyb_3 can be simulated and is indistinguishable from the world in Hyb_4. □

Security against corrupt P_2. Since P_2's collusive behaviour is the same as P_1's, we do not describe it here.

C. Supplementary experiments

In this section, we will compare with the similar protocol [29, 30] in Section 1.2, which are [BMX22] and [BMSTZ24]. Since we did not find the source code, we use the experimental data in protocol [29] directly.

Overall evaluation. Based on the analysis in Table 4, our protocol outperforms protocol [30] in terms of communication overhead by approximately 141× to 246×, while incurring a slightly higher overhead compared to protocol [29] (by around 2× to 2.5×). In terms of runtime, our protocol outperforms protocol [30] by approximately 31× to 339×, and outperforms protocol [29] by approximately 6.8× to 25.3×. Overall, our protocol shows significant advantages.

Table 4: Runtime and communication overhead comparison.

Protocol		$(N_{X_1}+N_{X_2}, N_{Y_1}+N_{Y_2})$		$(N_{X_1}+N_{X_2}, N_{Y_1}+N_{Y_2})$	
		$(2^{18}+2^{8}, 2^{18}+2^{8})$	$(2^{18}+2^{10}, 2^{18}+2^{10})$	$(2^{20}+2^{8}, 2^{20}+2^{8})$	$(2^{20}+2^{10}, 2^{20}+2^{10})$
[BMX22]	Comm./MB	0.02	0.06	0.02	0.06
	Time/s	1.65	6.07	1.65	6.08
[BMSTZ24]	Comm./MB	7.08	27.7	7.57	29.6
	Time/s	20.2	79.4	21.6	84.9
Ours	Comm./MB	0.05	0.12	0.05	0.12
	Time/s	0.21	0.24	0.24	0.25

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported in Beijing Municipal Science \& Technology Commission New Generation of Information and Communication Technology Innovation Research and Demonstration Application of Key Technologies for Privacy Protection of Massive Data for Large Model Training and Application(Z231100005923047).

References

[1] C. Meadows, “A more efficient cryptographic matchmak ing protocol for use in the absence of a continuously available third party,” in Proc. IEEE Symp. Secur. Pri vacy. (SP). IEEE, 1986, pp. 134–134.

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

[3] E. De Cristofaro and G. Tsudik, “Practical private set intersection protocols with linear complexity,” in Int. Conf. Financial Cryptogr. Data Secur. Springer, 2010, pp. 143–159.

[4] 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 Proc. IEEE European Symp. Secur. Privacy. (EuroS&P). IEEE, 2020, pp. 370–389.

[5] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, “Extend ing oblivious transfers efficiently,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2003, pp. 145–161.

[6] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, “Spot light: lightweight private set intersection from sparse ot extension,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2019, pp. 401–431.

[7] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, “Phasing: Private set intersection using permutation based hashing,” in Proc. 24th USENIX Secur. Symp. (USENIX Secur)., 2015, pp. 515–530.

[8] P. Benny, S. Thomas, and Z. Michael, “Faster private set intersection based on ot extension,” in Proc. 23th USENIX Secur. Symp. (USENIX Secur)., 2014, pp. 797 812.

[9] P. Rindal and M. Rosulek, “Malicious-secure private set intersection via dual execution,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1229–1242.

[10] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, “Oblivious key-value stores and amplification for private set intersection,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2021, pp. 395–425.

[11] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, “Psi from paxos: fast, malicious private set intersection,” in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Tech. (EUROCRYPT). Springer, 2020, pp. 739–767.

[12] P. Rindal and P. Schoppmann, “Vole-psi: fast oprf and circuit-psi from vector-ole,” in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Tech. (EUROCRYPT). Springer, 2021, pp. 901–930.

[13] S. Raghuraman and P. Rindal, “Blazing fast psi from improved okvs and subfield vole,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2022, pp. 2505–2517.

[14] H. Chen, K. Laine, and P. Rindal, “Fast private set intersection from homomorphic encryption,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1243–1255.

[15] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, “Pir psi: scaling private contact discovery,” Proc. Privacy Enhancing Technol., 2018.

[16] A. C. D. Resende and D. F. Aranha, “Unbalanced ap proximate private set intersection.” IACR Cryptol. ePrint Arch., vol. 2017, p. 677, 2017.

[17] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, “Privacy preserving error resilient dna searching through oblivious automata,” in Proc. ACM SIGSAC Conf. Com put. Commun. Secur., 2007, pp. 519–528.

[18] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, “Epione: Lightweight contact tracing with strong privacy,” ArXiv preprint arXiv., 2020. [Online]. Available: https://arxiv.org/abs/2004.13293.

[19] J. Chan, D. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, P. Sharma et al., “Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing,” ArXiv preprint arXiv., 2020. [Online]. Available: https://arxiv.org/abs/2004.03544.

[20] J. L. Jenkins, M. Grimes, J. G. Proudfoot, and P. B. Lowry, “Improving password cybersecurity through in expensive and minimally invasive means: Detecting and deterring password reuse through keystroke-dynamics monitoring and just-in-time fear appeals,” Inf. Technol. Develop., vol. 20, no. 2, pp. 196–213, 2014.

[21] M. Chase and P. Miao, “Private set intersection in the internet setting from lightweight oblivious prf,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2020, pp. 34–63.

[22] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, “Efficient batched oblivious prf with applications to private set intersection,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2016, pp. 818–829.

[23] B. A. Huberman, M. Franklin, and T. Hogg, “Enhancing privacy and trust in electronic communities,” in Proc. 1th ACM conf. Electron. commerce., 1999, pp. 78–86.

[24] Y. Huang, D. Evans, and J. Katz, “Private set intersection: Are garbled circuits better than custom protocols?” in Proc. Symp. Netw. Distrib. Syst. Secur. (NDSS)., 2012.

[25] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, “Efficient circuit-based psi with linear communication,” in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Tech. (EUROCRYPT). Springer, 2019, pp. 122–153.

[26] B. Pinkas, T. Schneider, C. Weinert, and U. Wieder, “Efficient circuit-based psi via cuckoo hashing,” in Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Tech. (EURO CRYPT). Springer, 2018, pp. 125–157.

[27] 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 Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2019, pp. 291–308.

[28] G. Couteau, P. Rindal, and S. Raghuraman, “Silver: silent vole and oblivious transfer from hardness of decoding structured ldpc codes,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2021, pp. 502–534.

[29] S. Badrinarayanan, P. Miao, and T. Xie, “Updatable private set intersection,” Proc. Privacy Enhancing Technol., vol. 2022, no. 2, 2022.

[30] S. Badrinarayanan, P. Miao, X. Shi, M. Tromanhauser, and R. Zeng, “Updatable private set intersection revis ited: Extended functionalities, deletion, and worst-case complexity,” in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT). Springer, 2025, pp. 200–233.

[31] M. O. Rabin, “How to exchange secrets with oblivious transfer,” Cryptol. ePrint Archive., 2020. [Online]. Avail able: https://eprint.iacr.org/2005/187.

[32] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, “Keyword search and oblivious pseudorandom func tions,” in Proc. Theory. Cryptogr. Conf. (TCC). Springer, 2005, pp. 303–324.

[33] V. Kolesnikov and R. Kumaresan, “Improved ot extension for transferring short secrets,” in Proc. Annu. Int. Cryptol. Conf. (CRYPT). Springer, 2013, pp. 54–70.