Multi-Party Private Set Intersection with Optimal Online Round Complexity and Updatability

Jiadi Zhang*, Hao Wang*✉ , Ye Su*, Xiaochao Wei*, Lei Wu*, Zhi Li†
* Shandong Normal University, †Linyi University
*zhangjiadi2008@126.com, *{wanghao, suye, wxc, wulei}@sdnu.edu.cn, †lizhi1@lyu.edu.cn

***Abstract*—Multi-party private set intersection (MPSI) enables multiple parties to securely compute the intersection of their private datasets without revealing any information beyond the intersection itself. However, as the number of participants scales, the performance of multi-party PSI protocols is significantly influenced, with the number of interaction rounds emerging as a critical bottleneck. In this paper, we propose a novel MPSI protocol and its updatable extension based on function secret sharing and oblivious key-value stores, achieving optimal one-pass online interaction. Our construction guarantees security in the semi-honest model, resisting collusion among any $n-1$ parties when the Leader is honest, and among any $n-2$ parties when the Leader is corrupted—an assumption that aligns well with many practical deployment scenarios.**

Furthermore, our multi-party updatable PSI (MUPSI) protocol allows parties to efficiently compute the intersection over dynamically updated sets. Our MUPSI protocol achieves collusion resistance against any $n-1$ participants, assuming an honest Leader. It ensures that both computational and communication complexities scale exclusively with the size of the updates rather than the entire datasets, exhibiting superior performance particularly when handling unbalanced sets and large participant cohorts. All proposed protocols exhibit strong scalability with respect to participant count.

We demonstrate the superiority of our protocols through implementation and comparison with state-of-the-art MPSI protocols. Experiments show that when the number of participants ranges from 20 to 140 and the set size ranges from 2^{12} to 2^{20}, our MPSI protocol is competitive. Notably, in the WAN setting with 140 participants and a set size of 2^{20}, the running time is reduced by $49.1\times$ compared with GLW+24. Our MUPSI protocol avoids PSI operations on entire sets, achieving a reduction in running time by an order of magnitude.

I. Introduction

In secure multi-party computation (MPC), private set intersection (PSI) is arguably one of the functions with the most pressing practical application needs. In the PSI scenario, multiple participants each hold a set and aim to compute the intersection of all sets without disclosing any additional information.

✉Corresponding author: Hao Wang

A version of this paper is accepted to appear at NDSS 2027.

Based on the number of participants, PSI can be categorized into two-party PSI and multi-party PSI (MPSI). In recent years, two-party PSI [1]–[5] has advanced significantly. Protocols based on oblivious key-value store (OKVS) and vector oblivious linear evaluation (VOLE) [4], [5] achieve performance comparable to that of naive, insecure hashing-based PSI. These two-party PSI protocols are widely applicable in privacy-preserving scenarios, including security incident information sharing [3], private contact discovery [5], and password checkup. A notable real-world deployment is Google's 2019 "password-checkup" plugin, which employs PSI to securely detect password breaches while preserving user privacy. Furthermore, researchers have proposed various extensions of PSI, such as updatable PSI (UPSI). In the updatable scenario, PSI is performed on a regular (e.g., daily) basis with updated sets, where the updates can be small compared to the entire sets. This makes it suitable for many real-world applications, such as aggregated ads measurement and privacy-preserving contact tracing.

As data sharing expands to involve more participants, multi-party PSI (MPSI) [6]–[16] has gained increasing prevalence in recent years. Many applications of privacy computing find MPSI as a significant cryptographic tool, such as multi-party access control [17], anonymous voting systems, consensus mechanisms [6], targeted consumer discovery for advertising [18], and private pandemic contact tracing [19]–[22]. Another notable use case arises in edge computing, where a variant of MPSI [23] was employed for cache sharing in 2022. This technique enables multiple network operators to collaboratively store frequently accessed common data items in their capacity-limited shared caches while preserving the privacy of their respective datasets.

From the perspective of technology, the design of MPSI needs to consider different levels of collusion resistance, avoid leakage of intersection for partial participants, and consider the scalability of the number of participants. So it presents greater challenges compared to the design of two-party PSI. Most existing MPSI protocols suffer from the requirement for a significant number of participants to perform public-key operations, the necessity of multiple trusted special central participants, and involve complex communication rounds among central or even all participants. These limitations hinder their ability to support a large number of participants effectively. Furthermore, in the face of dynamic datasets, these protocols lack support for MPSI in updatable scenarios, further limiting

their practicality.

Recently, in [24], the authors proposed an efficient MPSI protocol that can reduce MPSI to two-party PSI between two central participants. However, their protocol is vulnerable to collusion between the two central participants. Furthermore, The number of interaction rounds of the multi-party PSI protocol has a significant impact on the effectiveness of applications, as does the communication complexity. It is not trivial to directly apply the efficiently realized two-party primitives to multi-party scenarios. The new multi-party PSI protocol needs to combine the benefits of both round and communication efficiency.

A. Our Contributions

In this work, we introduce an efficient multi-party PSI protocol and the updatable extension, which exhibit advantages in both round count and scalability. Our main contributions can be summarized as follows:

Multi-Party PSI with One-Pass Online Intersection (Section III). We propose a multi-party PSI protocol with only one-pass online intersection in which no parties engage in global broadcasting. On the basis of oblivious key-value store (OKVS) and distributed multi-point functions (DMPF) primitives, parties in our MPSI protocol require only one-pass online communication, which is a one-time, non-interactive message transmission, while the offline phase solely consists of a single zero-sharing operation. Benefiting from the star structure, our MPSI protocol supports parallel computation and exhibits strong scalability. Our construction guarantees security in the semi-honest model, resisting collusion among any $n-1$ parties when the Leader is honest, and among any $n-2$ parties when the Leader is corrupted. This assumption is strictly weaker than those required by many existing MPSI protocols (e.g., [24] relies on the non-collusion of two specific parties). Even in the worst case where the Leader is corrupted, the adversary still needs to corrupt an additional $n-2$ Clients to break the security. This condition is satisfied in most real-world deployments, as corrupting the Leader together with a large majority of Clients represents an exceptionally high attack threshold.

Multi-Party Updatable PSI (Section V). We propose a multi-party updatable PSI protocol that supports regular computation of the intersection over participants' private sets that are continuously updated. We present a protocol for this setting only allowing one party to learn the output. Our MUPSI protocol requires the Leader to be honest. Under this assumption, the protocol achieves collusion resistance against any $n-1$ participants and has the guarantee that both the computational and communication complexity only grow with the set updates instead of the entire sets. When the entire set size is sufficiently large and the update sets are sufficiently small, or the number of participants reaches a certain level, our MUPSI protocol demonstrates superior performance.

Evaluation (Section VI). We implement and evaluate the efficiency and scalability of our MPSI and MUPSI protocols. We perform experiments with up to 140 parties and 2^{20} elements to demonstrate the practicality of our protocol. Then we compare our protocols with the SOTA works. The results show that our multi-party protocols are competitive. Specifically, our MPSI protocol is 4.9–$49.1\times$ faster in the WAN setting. The performance improvement becomes even more significant as the number of participants and the size of the dataset increase. For our MUPSI protocol, compared with performing PSI using the entire set, our protocol achieves optimal running time and communication cost under both LAN and WAN settings in the vast majority of cases. The running time of the Leader is reduced $2.5-46.5\times$, and the running time of the Client is reduced by an order of magnitude. Moreover, the computation and communication costs depend only on the small updated set and do not increase with the number of participants. To the best of our knowledge, our updatable scheme is the first protocol that reports on more than 100 participants.

B. Related Work

In Table I, we present a comprehensive comparison of the existing MPSI protocols, including those based on homomorphic encryption (HE) [6], [8], [9], garbled circuit (GC) [7], [14], oblivious transfer (OT) [10]–[13], [15], [24] and secret sharing (SS) [25]. From a technical perspective, there are primarily two types of MPSI constructions: directly implementing MPSI and reducing MPSI to two-party PSI. Furthermore, we refer the reader to Ref. [26], in which Vos et al. provided a comprehensive overview of the current MPSI protocols.

Directly implementing MPSI. In 2024, Wu et al. [27] proposed an MPSI protocol, following the zero-sharing reconstruction paradigm, where the key idea is the OKVS chain. When one party is honest, they constructed an efficient MPSI protocol with the OKVS chain. To identify the honest party, they utilized oblivious pseudorandom function (OPRF) protocol to improve security.

In 2025, Dong et al. [28] proposed a framework for multi-party private set operations (MPSO) and its variants, including MPSI. The framework is based on a novel primitive called predicative zero-sharing, where parties receive shares of zero when a predicate evaluates to true, and shares of a random value otherwise. It is the first framework to achieve both high flexibility and generality for MPSO without relying on generic MPC techniques, offering linear complexity and security against arbitrary collusion in the standard semi-honest model.

In 2026, Guan et al. [29] proposed an MPSI protocol based on reducible zero-sharing. The protocol resists collusion by any t parties in the semi-honest model, follows an offline/online paradigm, and relies solely on symmetric-key operations in the online phase, achieving $O((n+t)m)$ computation and communication complexity. Experiments show that it significantly outperforms existing schemes in both online running time and communication cost.

TABLE I: Comparison of related work with our protocol. n is the number of participants. t is the maximum number of corrupted participants. Each party holds a set of size m. M is the domain of the element. k is the number of hash functions in GBF. λ and κ are statistical and computational security parameters respectively. The sign * implies that specific participants are not allowed to be corrupted simultaneously.

Building Block	Protocol	Topology	Communication		Computation		Rounds	Corrupt Threshold	Security
			Leader	Client	Leader	Client			
HE	CJS12 [8]	Mesh	$O(n^2m\lambda)$		$O(nm\lambda)$		$O(n)$	$t < \frac{n}{2}$	semi-honest
									malicious
	HV17 [9]	Star	$O(nm\lambda)$	$O(m\lambda)$	$O(nmlog(m\kappa))$	$O(m\kappa)$	8	$t < n$	semi-honest
			$O((n^2 + nmlogm)\kappa)$	$O((n + mlogm)\kappa)$	$O(m^2\kappa)$				malicious
	BEH+21 [6]	Star	$O(nm\lambda)$	$O(m\lambda)$	$O(nmk)$	$O(mk\lambda)$	4	$t < n$	semi-honest
GC	CDG+21 [7]	Star	$O(nm(\kappa + \lambda + logm))$	$O(m(\kappa + \lambda + logm))$	$O(nm\kappa)$	$O(m\kappa)$	16	$t < \frac{n}{2}$	semi-honest
	RKG+23 [14]	Mesh	$O(nm\kappa\ log(nm))$		$O(nm\kappa\ log(nm))$		$O(n)$	$t < n$	semi-honest
OT	KMP+17 [12]	Mesh	$O(nm\lambda)$	$O(m\lambda)$	$O(nm\kappa)$	$O(m\kappa)$	8	$t < n$	augmented
									semi-honest
				$O(tm\lambda)$		$O(tm\kappa)$			semi-honest
	IOP18 [10]	Star	$O(nmk\lambda)$	$O(mk\lambda)$	$O(nmk\lambda)$		$O(n)$	$t < n$	augmented
		Hypercube	$O(mk\lambda\ logn)$						semi-honest
		Star	$O(nmk\lambda)$						semi-honest
	ZLL+19 [15]	Star	$O(nm\kappa^2 + nm\kappa log(m\kappa))$		$O(nmk\kappa)$		5	$t < n^*$	malicious
	KMS21 [11]	Wheel	$O(nm\lambda)$	$O(mk\lambda)$	$O(nm\lambda)$	$O(mk\lambda)$	8	$t < n^*$	semi-honest
	NTY21 [13]	Mesh	$O((n + m)\kappa)$	$O(m\kappa)$	$O(nm\kappa)$	$O(m\kappa)$	8	$t = 1$	malicious
			$O(max\{t, n - t\}\ m\kappa)$		$O((n - t)m\kappa)$			$t < n$	
	WYC24 [27]	Wheel	$O(nm(\lambda + \kappa + logm))$	$O(tm(\lambda + \kappa + logm))$	$O(nm\kappa)$	$O(tm\kappa)$	$O(n)$	$t < n$	semi-honest
		Star					6		
	GLW+24 [24]	Star	$O(n\kappa + m\lambda)$	$O(m\lambda)$	$O(nm\lambda)$	$O(m\lambda)$	4	$t < n^*$	semi-honest
			$O((n + m)\kappa)$	$O(m\kappa)$	$O(nm\kappa)$	$O(m\kappa)$		$t < n - 1^*$	semi-honest
	DCZ+25 [28]	Star	$O(mn)$	$O(m)$	$O(mn)$	$O(m)$	4	$t < n$	semi-honest
SS	ZLM+25 [25]	Wheel	$O(mnlog(m + \lambda))$	$O(m\lambda)$	$O(mnlog^2m)$	$O(mlog^2m)$	3(online)	$t < n$	semi-honest
		Star	$O(mn\lambda)$		$O(m(log^2m + \lambda))$		3	$t < n - 2$	
	GGA+26 [29]	Star	$O((n + t)m\lambda + t\lambda logm)$	$O(t\kappa + m\lambda)$	$O((n + t)m\lambda)$	$O(tm\lambda)$	$O(t)$	$t < n$	semi-honest
	Ours	Star	$O(mn\lambda)$	$O(m\lambda)$	$O(m(n + \lambda))$	$O(m\lambda)$	1(online)	$t < n^*$	semi-honest

**Note: In this table, one round refers to one pass (a single message sent by one party).*

Reducing MPSI to two-party PSI. Another approach is to reduce MPSI to two-party PSI. All protocols using this approach require specific participants to not collude.

In 2019, Zhang et al. [15] realized n-party MPSI by invoking the two-party PSI protocol [2] for $O(n)$ times. They extended the cut-and-choose technique from [2] to the multi-party setting, where all participants make commitments to their inputs and allow other participants to verify a part of them, resulting in a maliciously secure MPSI protocol.

In 2021, Kavousi et al. [11] proposed a semi-honest MPSI protocol. They combined the star topology and the chain topology to create a wheel-like topology, where the Leader broadcasts messages and Clients process and forward them sequentially. This approach ensures that the computation and communication complexities of Clients are independent of the number of participants, but it assumes the Leader does not collude with any Client.

In 2024, Gao et al. [24] proposed a primitive called bicentric zero-sharing, which can reduce MPSI to two-party PSI between two central participants named Pivot and Leader. By combining this structure with two-party PSI, they proposed a new efficient and scalable MPSI protocol along with its variants, achieving semi-honest security under the assumption that the two central participants are not simultaneously corrupted.

In 2025, Zhao et al. [25] proposed two novel MPSI protocols. 1M$\mathbb{P}$SI achieves one-round online interaction via Ring-OLE, but its circular structure incurs implicit multi-round local overhead. 2M$\mathbb{P}$SI adopts a dual-core star structure with OKVS, achieving superior performance on large sets or many participants, yet requires the two cores not be simultaneously corrupted.

II. Preliminaries

A. Notations

We denote the computational and statistical security parameters by κ and λ, respectively. For the multi-party PSI protocols in this paper, we denote n as the number of parties and m as the set size. For $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, ..., n\}$. The party P_i holds a set $X_i = \{x_i^1, ..., x_i^m\}$, where $i \in [n]$. We set P_1 as the Leader who obtains the intersection and the rest of parties $P_2, P_3, ..., P_n$ as the Clients in our MPSI protocol.

For the multi-party UPSI protocol in this paper, we denote M as the size of old set and m as the size of update set, where

$m \ll M$. Let X_i^d be the set of P_i after d days and X_i^Δ be the update set input every day, where $i \in [n]$.

B. Ideal Functionality of Multi-Party PSI

We delineate particular secure computing tasks by specifying the computation of a desired functionality (executed by a trusted third party). The ideal functionality of multi-party PSI ($\mathcal{F}_{MPSI}$) is shown in Fig. 1.

An upper bound m may be postulated for the cardinality of each party's input set. This postulation implies that the parties $P_1, ..., P_n$ may possess input sets of varying sizes, each not exceeding m. Adopting this approach ensures that the precise sizes of the parties' sets remain undisclosed throughout the protocol's execution. Notably, we designate only P_1 obtains the intersection.

$\mathcal{F}_{MPSI}^{m,n}$

Parameters: n parties $P_1, ..., P_n$, the size of the sets m.
Functionality:

1) Each party P_i inputs set $X_i = \{x_i^1, ..., x_i^m\} \subseteq \{0,1\}^*$.
2) Party P_1 receives the intersection $I = \bigcap_{i=1}^n X_i$ and other parties receive nothing.

Fig. 1: The functionality of multi-party PSI ($\mathcal{F}_{MPSI}^{m,n}$).

C. Ideal Functionality of Multi-Party Updatable PSI

We define and describe the secure computing tasks of multi-party updatable PSI. It should be noted that we mainly focus on the addition-only setting, that is, parties only add new elements to their set every day. The ideal functionality of multi-party updatable PSI ($\mathcal{F}_{MUPSI}$) is shown in Fig. 2.

On day d, let all the parties hold the old private set X_i^{d-1}, as well as the update set X_i^Δ. In MUPSI, the party P_1 have already learned $I_{d-1} = \bigcap_{i=1}^n X_i^{d-1}$ of the old sets and would like to learn the updated intersection $I_d = \bigcap_{i=1}^n (X_i^{d-1} \cup X_i^\Delta)$, with no leaking about X_i^{d-1} and X_i^Δ.

D. Security Model

The formal security definition of an MPC protocol is based on comparing two output distributions from two worlds, *i.e.*, an ideal world and a real world. $A \stackrel{c}{\equiv} B$ denotes two distribution probability ensembles $A = \{X(a,n)\}_{a\in\{0,1\}^*; n\in\mathbb{N}}$ and $B = \{Y(a,n)\}_{a\in\{0,1\}^*; n\in\mathbb{N}}$ are *computationally indistinguishable*. That is, if for every non-uniform polynomial-time algorithm D, the symbol "$\stackrel{c}{\equiv}$" means that, for every $a \in \{0,1\}^*$ and $n \in \mathbb{N}$, there exists a negligible function $\mu(\cdot)$ satisfying,

$$|Pr[D(X(a,n)) = 1] - Pr[D(Y(a,n)) = 1]| \leq \mu(n).$$

We consider the semi-honest security in this paper due to its sufficiency in practical applications and this secure model can represent a wide range of real-world scenarios. All parties in the semi-honest model must execute the protocol correctly. On the other hand, if the adversary gains access to the records of all incoming messages, he may attempt to extract sensitive information that needs to be protected.

$\mathcal{F}_{MUPSI}^{M,m,n}$

Initialization: $X_i^0 = \emptyset$.
Day d:
Parameters: n parties $P_1, ..., P_n$, the old sets X_i^{d-1}, the size of the update sets m.
Functionality:

1) Input: P_i input an update set $X_i^\Delta \subseteq \{0,1\}^*$, where $X_i^\Delta \cap X_i^{d-1} = \emptyset$
2) Update: On receiving the inputs from parties, the ideal functionality updates $X_i^d = X_i^{d-1} \cup X_i^\Delta$.
3) Output: Party P_1 receives the intersection $I_d = \bigcap_{i=1}^n X_i^d$ and other parties receive nothing.

Fig. 2: The functionality of multi-party UPSI ($\mathcal{F}_{MUPSI}^{M,m,n}$).

In this paper, we consider an adversary that can corrupt up to $n-1$ participants. However, the actual collusion resistance depends on whether the Leader is corrupted:

- If the Leader remains honest, the protocol resists collusion among any $n-1$ participants.
- If the Leader is corrupted, the adversary is additionally assumed not to corrupt more than $n-3$ Clients (i.e., at most $n-2$ participants in total, including the Leader).

For the updatable scheme of our protocol, we require the Leader to be honest. Under this assumption, the protocol achieves collusion resistance against any $n-1$ participants.

The functionality of $\mathcal{F}_{MPSI}$ can be given by simply writing $f_1 : (X_1, ..., X_n) \mapsto (I, \perp, ..., \perp)$, and the functionality of $\mathcal{F}_{MUPSI}$ can be given by writing $f_2 : (X_1^\Delta, ..., X_n^\Delta) \mapsto (I, \perp, ..., \perp)$, where I is the intersection and $\perp$ denotes the empty string.

The formal security definition is given as follows.

Definition 1. *The security parameter is λ, and input sets X_i are from P_i. $view_i^\Pi(X_i, \lambda) = (x, r_i, M_i)$ denotes the view of P_i in the protocol Π, where x is the input set, r_i is the randomness used by P_i, and M_i is the message received by P_i. The output of P_i in the protocol is denoted as $output_i^\Pi(X_i, \lambda)$, and the joint output of the parties in the protocol is $output^\Pi(X, \lambda) = (output_1^\Pi(\cdot), ..., output_n^\Pi(\cdot))$. We say that the protocol securely computes ideal functionality in semi-honest model if*

- *The correctness holds* $output^\Pi(X, \lambda) \stackrel{c}{\equiv} \{f(X_1, ..., X_n)\}_{X_i \subseteq \{0,1\}^*, \lambda}$
- *Given the input from the parties set C controlled by adversary $\mathcal{A}$ in the real world, there exist probabilistic polynomial-time (PPT) simulator Sim under the security parameter in the ideal world, such that*

$$\{Sim_C(1^\lambda, input(X_{i,P_i\in C}), output(f_{i,P_i\in C}))\}_{X_i\subseteq\{0,1\}^*,\lambda}$$
$$\stackrel{c}{\equiv} \{view_i^\Pi(X_{i,P_i\in C}, \lambda)\}_{X_i\subseteq\{0,1\}^*,\lambda}$$

E. Distributed Multi-Point Functions

An $(2-party)$ FSS scheme can generate a pair of keys (k_0, k_1) for a secret function f. These two keys will not reveal any information about f. Furthermore, party P_b holding the key k_b and the public input x can compute the additive share of the function value $f(x)$ in a non-interactive way.

Multi-Point Functions. Given a field size N and an Abelian group $\mathbb{G}$, a t-point function $f_{A,B} : [N] \to \mathbb{G}$, where $A = \{\alpha_1, ..., \alpha_t\} \subset N$ and arranged in ascending order, $B = \{\beta_1, ..., \beta_t\} \in \mathbb{G}^t$. For $1 \leqslant i \leqslant t$, the t-point function outputs β_i on α_i and the output 0 on other inputs. The formal definition of the multi-point functions is as follows:

$$f_{A,B} = \begin{cases} \beta_1, & x = \alpha_1 \\ \beta_2, & x = \alpha_2 \\ \cdots & \\ \beta_t, & x = \alpha_t \\ 0, & other \end{cases}$$

We use FSS for the family $\mathcal{F}$ of *multi-point functions*, where a multi-point function $f_{A,B}$ evaluates to $\beta \in B$ on the special input $\alpha \in A$ and to 0 on all other inputs. An FSS scheme for multi-point functions is referred to as a distributed multi-point function (DMPF) [30].

Definition 2. *A (2-party) distributed multi-point function (DMPF) is a triple of algorithms with the following syntax:*

1) $Gen(1^\lambda, f_{A,B}) \to (k_0, k_1)$*: On input security parameter* $\lambda \in \mathbb{N}$ *and point function* $f_{A,B}$*, the key generation algorithm Gen returns a pair of keys* $k_0, k_1 \in \{0,1\}^*$.
2) $Eval_b(1^\lambda, k_b, x) \to y_b$*: On input key* $k_b \in \{0,1\}^*$ *and input* $x \in [N]$*, the evaluation algorithm of server b,* $Eval_b$*, returns* $y_b \in \mathbb{G}$.
3) $FullEval_b(1^\lambda, k_b)$*: Returns* $\{Eval_b(1^\lambda, k_b, x)\}_{x\in[N]}$.

The algorithms need to satisfy the following requirements:
Correctness: *For every* $\lambda, f_{A,B}$ *such that* $B \in \mathbb{G}^t, x \in [N]$

$$Pr\left[\begin{matrix}(k_0,k_1) \leftarrow Gen(1^\lambda, f), \\ Eval_0(k_0, x) + Eval_1(k_1, x) = f_{A,B}(x)\end{matrix}\right] = 1.$$

Security: *For any corrupted party b=0,1, there exists a probabilistic polynomial-time simulator* S *such that for any* $f_{A,B}$*, the following* $Real$ *and* $Ideal$ *are computationally indistinguishable:*

- $Real$*:* $(k_0, k_1) \leftarrow Gen(1^\lambda, f)$*, output* k_b.
- $Ideal$*: output* $S(1^\lambda, N, \mathbb{G}, t)$.

F. Oblivious Key-Value Store

An Oblivious Key-Value Store (OKVS) is a type of data structure that associates a collection of keys with their corresponding values. It was proposed as a primitive for PSI protocols [31], and improved by a series of works [4], [32].

Definition 3. *An Oblivious Key-Value Store is defined by a key set* K *and a value set* V*. It consists of two algorithms:*

1) $Encode(\{(k_1, v_1), ..., (k_m, v_m)\}) \to S$*: On input a set of key-value pairs* $\{k_i, v_i\}_{i\in[m]} \subseteq (K \times V)^m$*, the encoding algorithm outputs an object* $S \in V^n$ *(or, with a statistically negligible probability, an error indicator* $\perp$*).*
2) $Decode(S, k) \to v$*: On input an encoding* $S \in V^n$ *and a key* $k \in K$*, it outputs the corresponding value* $v \in V$ *or returns a random value.*

The scheme needs to satisfy the following requirements:
Correctness: *For all* $P \in (K \times V)^m)$ *such that* $Encode(P) \to S \neq \perp$, $Decode(S, k)$ *outputs* v *whenever* $(k, v) \in S$.
Obliviousness: *Given any distinct key sets* $K^0 = \{k_1^0, ..., k_m^0\}$ *and* $K^1 = \{k_1^1, ..., k_m^1\}$ *that are different, if they are associated with random values then the encoded output are computationally indistinguishable, i.e.,*

$$\{Encode_H(\{(k_1^0, v_1), ..., (k_m^0, v_m)\})|v_i \leftarrow V, i \in [n]\} \stackrel{c}{\equiv}$$
$$\{Encode_H(\{(k_1^1, v_1), ..., (k_m^1, v_m)\})|v_i \leftarrow V, i \in [n]\}$$

Furthermore, an OKVS is linear if, for all $k \in K$ and $S = S_1 \oplus S_2 \oplus ... \oplus S_n$, then $Decode(S, k) = Decode(S_1, k) \oplus Decode(S_2, k) \oplus ... \oplus Decode(S_n, k)$. It means that the Decode function can take any object with the same format as S as input. The majority of existing OKVS constructions [4], [5], [31]–[33] are based on matrices, where the Decode function only involves a vector-matrix multiplication, satisfying linearity.

III. One-Pass Multi-Party PSI

In this section, in the semi-honest model, we construct an efficient multi-party PSI protocol that securely computes the functionality in Fig. 1.

A. An Overview

From the perspective of communication patterns, our MPSI protocol uses the star structure, as shown in Fig. 3, where P_1 acts as the central core Leader, while the remaining parties act as Clients to form the outer circle. This structure allows our MPSI protocol to be highly scalable because the complexity for each party except for the central Leader remains independent of the total number of parties, rendering $O(m\lambda)$ computation and communication complexity.

One major advantage of our protocol is that it requires only one round online communication, or more precisely, each client only needs to send information to the leader once, known as one-pass, while in the offline phase, it requires solely one zero-sharing operation among clients.

B. MPSI protocol

Our multi-party PSI protocol is shown in Fig. 4. We designate P_1 to be the leader and the recipient of the intersection.

Initially, the preprocessing phase generates correlated randomness among clients, which is used in the online phase to

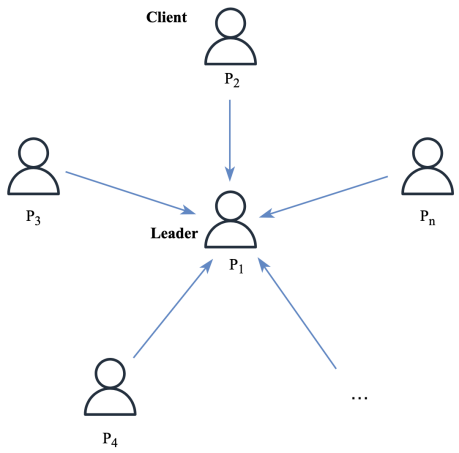

Fig. 3: Communication pattern in MPSI

protect individual client's information from being extracted by an honest-but-curious leader.

In the online phase, the Client P_i encodes its private set elements as a multi-party function and generates a pair of FSS keys. Then the P_i utilizes one of the FSS keys to calculate the FSS share $\omega_{i,k}^0$ for each element and constructs the OKVS encoding S_i using x_i^k as the key and $\omega_{i,k}^0$ as the value. Subsequently, P_i blinds the OKVS encoding with the zero-sharing share r_i. Finally, P_i sends S_i and the other FSS key key_i^1 to Leader P_1.

The Leader P_1 aggregates all S_i and decodes the aggregated S by its set elements x_1^k, utilizing the linearity of OKVS, where the blinding shares r_i cancel out as their XOR sum is zero, thus not affecting the decoding result. Next, P_1 uses each FSS key key_i^1 to calculate FSS shares and then aggregates the result evaluated by the same x_1^k. Finally, P_1 performs equality judgment between the decoding results of OKVS and the evaluation results of FSS and then computes the intersection of the sets.

C. Complexity Analysis

In this section, we present a theoretical complexity analysis of our MPSI protocol $\Pi_{MPSI}^{m,n}$.

The communication overhead of our $\Pi_{MPSI}^{m,n}$ consists of sending DMPF keys and OKVS encodings. Each Client needs to send a DMPF key of size $O(m)$ and an OKVS encoding of size $O(m\lambda)$ to the Leader. The communication overhead on each Client is $O(m\lambda)$, and the total communication overhead of the Leader is $O(mn\lambda)$.

The computational complexity of $\Pi_{MPSI}^{m,n}$ consists of two parts: the construction and querying of OKVS, and the key generation and evaluation of DMPF. The complexity of DMPF execution on Client is $O(m)$, and the cost of constructing OKVS is $O(m\lambda)$. The total computational overhead of the Clients is $O(m\lambda)$. For the Leader, the complexity of executing DMPF evaluation is $O(mn)$, and the Leader also requires $O(m\lambda)$ computational overhead to perform the OKVS query. The total computational overhead of the Leader is $O(m(n+\lambda))$.

IV. Security Proof

In this section, we begin by proving the correctness of our MPSI protocol $\Pi_{MPSI}^{m,n}$ and subsequently provide a rigorous security verification.

A. Correctness

This section rigorously proves the functional correctness of the proposed multi-party PSI protocol, that is, when all parties execute the protocol honestly, the Leader can accurately calculate the intersection of all party sets.

Lemma 1. *OKVS based on polynomials or matrices clearly exhibits linearity, which means if $S = \bigoplus_{i=1}^{n} S_i$, for all $k \in K$, $Decode(S,k) = \bigoplus_{i=1}^{n} Decode(S_i,k)$.*

Theorem 1. *Assume all parties execute the protocol honestly. Then the set output by the Leader is exactly the intersection of the input sets of all n parties.*

Proof: We consider two cases.

1) $x \in \bigcap_{i=1}^{n} X_i$ (x is in the intersection).
For each Client P_i ($i \in [2,n]$), since $x \in X_i$, by the definition of DMPF, x is an encoded special point. Let $\omega_i^0 = Eval(1^\lambda, key_i^0, x)$ and $\omega_i^1 = Eval(1^\lambda, key_i^1, x)$. By the correctness of DMPF, there is: $\omega_i^0 \oplus \omega_i^1 = \beta_i$.
Then, Client P_i encodes the key-value pairs $(x, \beta_i \oplus \omega_i^0)$ into its OKVS structure S_i, and blinds S_i through its zero-sharing share r_i. Due to the fact that $\bigoplus_{i=2}^{n} r_i = 0$, after the XOR aggregation of S_i, the value of r_i will cancel out. When the Leader P_1 decodes the aggregated OKVS encoding $S = \bigoplus_{i=2}^{n} S_i$, according to Lemma 1, we have:

$$\begin{aligned} result_1 &= Decode(S,x) \\ &= \bigoplus_{i=2}^{n} Decode(S_i,x) \\ &= \bigoplus_{i=2}^{n} (\beta_i \oplus \omega_i^0). \end{aligned}$$

Next, Leader P_1 uses all Clients' DMPF key key_i^1 to calculate: $result_2 = \bigoplus_{i=2}^{n} \omega_i^1$. By the correctness of DMPF, we can substitute ω_i^0 for ω_i^1 to obtain $result_2 = \bigoplus_{i=2}^{n} (\beta_i \oplus \omega_i^0)$. Hence, $result_1 = result_2$, satisfying the match condition, and x is correctly identified as an intersection element.

2) $x \notin \bigcap_{i=1}^{n} X_i$ (x is not in the intersection).
Let $R \subseteq \{2, ..., n\}$ be the set of Clients whose input contains x. Since x is not in the intersection, R is a proper subset. For any Client P_j where $j \notin R$:
• In OKVS encoding: $Decode(S_j, x)$ returns a pseudo-random value $r_j \neq \beta_i \oplus \omega_i^0$.
• In DMPF evaluation: $\omega_i^0 \oplus \omega_i^1 = 0$.

$\Pi_{MPSI}^{m,n}$

Parameters: Each party P_i has private set X_i of size at most m.

Initial Computation:

1) All Clients P_i $(i \in [2,n])$ jointly generate a set of random shares $\{r_2, ..., r_n\}$ satisfying $\bigoplus_{i=2}^{n} r_i = 0$. Each client P_i receives its share r_i.

For Each Client P_i $(i \in [2,n])$ **:**

2) Party P_i encodes its set elements as a multi-point function $f_{X_i,B}$ for $X_i = \{x_i^1, ..., x_i^m\}$ and $B = \{\beta_i^1, ..., \beta_i^m\}$ and generates a pair of keys $(key_i^0, key_i^1) \leftarrow DMPF.Gen(1^\lambda, f_{X_i,B})$.
3) Party P_i uses key_i^0 to locally calculates $\omega_{i,k}^0 \leftarrow DMPF.Eval(1^\lambda, key_i^0, x_i^k)$ for each elements x_i^k in its set X_i.
4) P_i construct the OKVS encoding S_i by computing $S_i = Encode(x_i^k, \beta_i^k \oplus \omega_{i,k}^0) \oplus r_i$ for all $x_i^k \in X_i$.
5) P_i sends S_i and key_i^1 to Leader P_1.

For Leader P_1 **:**

6) Leader P_i received $\{S_2, ..., S_n\}$ and $\{key_1^1, ..., key_n^1\}$. Then P_1 aggregates all S_i by computing $S = \bigoplus_{i=2}^{n} S_i$.
7) For each elements x_1^k in X_1, Leader P_1 retrieves $result_k$ from S by computing $result_k^1 = Decode(S, x_1^k)$.
8) For each elements x_1^k in X_1, Leader P_1 use key_i^1 to calculates $\omega_{i,k}^1 \leftarrow DMPF.Eval(1^\lambda, key_i^1, x_1^k)$.
9) Then for the result $\omega_{i,k}^1$ evaluated from the same x_1^k, P_1 aggregates it by computing $result_k^2 = \bigoplus_{i=2}^{n} \omega_{i,k}^1$.
10) Leader P_1 verifies whether $result_k^1 = result_k^2$. In the event of equality, P_1 proceeds to include element x_1^k in the intersection set.

Fig. 4: Our multi-party private set intersection $\Pi_{MPSI}^{m,n}$.

Thus:

$$\begin{aligned} & result_1 \oplus result_2 \\ &= \bigoplus_{j \notin R} (r_j \oplus \omega_i^1) \oplus \bigoplus_{j \in R} [(\beta_i \oplus \omega_i^0) \oplus \omega_i^1] \\ &= \bigoplus_{j \notin R} (r_j \oplus \omega_i^0) \oplus \bigoplus_{j \in R} [(\beta_i \oplus \omega_i^0) \oplus (\beta_i \oplus \omega_i^0)] \\ &= \bigoplus_{j \notin R} (r_j \oplus \omega_i^0). \end{aligned}$$

$(r_j \oplus \omega_i^0)$ is the XOR of pseudorandom values, which is itself pseudorandom. Therefore:

$$Pr[result_1 = result_2] \leq negl(\lambda).$$

The equality holds only with negligible probability due to pseudorandomness. In conclusion, the protocol $\Pi_{MPSI}^{m,n}$ is correct with a non-negligible probability.

B. Security

To streamline our security proof, we model the DMPF defined in Definition 2 as an ideal functionality, denoted by $\mathcal{F}_{DMPF}$.

The functionality $\mathcal{F}_{DMPF}$ behaves as follows:

- Key generation phase: $\mathcal{F}_{DMPF}$ receives a point function description $f_{A,B}$ from the calling party, and outputs a pair of key (k_0, k_1) to be sent to the two parties P_0 and P_1, respectively.
- Evaluation phase: Any party P_b can query $\mathcal{F}_{DMPF}$ with an input $x \in [N]$, and the functionality returns the corresponding evaluation result y_b, satisfying $y_0 + y_1 = f_{A,B}(x)$.

Its correctness and security properties are exactly as stated in Definition 2. In our protocol, each invocation of the DMPF key generation or evaluation algorithms by a party is modeled as a query to $\mathcal{F}_{DMPF}$.

Similarly, we model the OKVS scheme defined in Definition 3 as an ideal functionality denoted by $\mathcal{F}_{OKVS}$. In the ideal world, the functionality $\mathcal{F}_{OKVS}$ takes from a party an input key-value set $P = \{(k_i, v_i)\}$, and outputs an encoding S. It then allows querying with (S, k) to retrieve the value associated with k. Specifically, for any $(k, v) \in P$, the decoding query returns v, and for any $k \notin P$, it returns a random value.

Theorem 2. *Given an ideal functionality, the MPSI protocol in Fig 4 securely computes the functionality $\mathcal{F}_{MPSI}^{m,n}$ (Fig. 1) and is secure against semi-honest adversaries, provided that the Leader colludes with at most $n-3$ Clients. Under this assumption, the protocol resists arbitrary collusion of up to $n-1$ parties, as described in Definition 1.*

Proof: Two types of entities are considered based on whether they obtain the intersection: the Leader P_1 and other parties P_i where $i \in [2,n]$. In addition, according to the MPSI security model, two kinds of collusion attacks are taken into consideration: the collusion set C_1 that $P_1 \notin C_1$ and the collusion set C_2 that $P_1 \in C_2$. We denote P_H as the honest party set. Therefore, after analyzing the four specified

types of attacks, we establish that each simulated view cannot be distinguished from its corresponding view of the realistic protocol, respectively.

1) P_j is corrupted where $j \neq 1$.
Given an OKVS, simulating the perspective script of party P_j is straightforward because P_j does not output from either the protocol $\Pi^{m,n}_{MPSI}$ or the functionality $\mathcal{F}^{m,n}_{MPSI}$. Specifically, $view^{\Pi}_j(X_j, \lambda)$ consists of the input private set X_j. Therefore, the simulated view $\{Sim_j(1^\lambda, input(X_j))\}_{X_j \subseteq \{0,1\}^*, \lambda}$ is computationally indistinguishable with the realistic view $\{view^{\Pi}_j(X_j, \lambda)\}_{X_j \subseteq \{0,1\}^*, \lambda}$.

2) P_1 is corrupted.
According to the protocol $\Pi^{m,n}_{MPSI}$, $view^{\Pi}_1(X_1, \lambda)$ consists of the input set X_1, randomness and all the received message S_i and key^1_i from $P_2, ..., P_n$. Therefore, given the intersection I, the simulator Sim_1 first samples $m - |I|$ random elements and form a set $X'_2 = \{I, R_1, R_2, ..., R_{m-|I|}\}$. Then Sim_1 encodes set X'_2 as a multi-point function $f_{X'_2,B'}$ for B' consists of random number $\{\beta'_1, ..., \beta'_m\}$. Sim_1 queries $\mathcal{F}_{DMPF}$ with $f_{X'_2,B'}$ and receives keys $(key^{0'}_2, key^{1'}_2)$ in return. By the security of $\mathcal{F}_{DMPF}$, the pair of keys $(key^{0'}_2, key^{1'}_2)$ is computationally indistinguishable with the realistic view. Next, Sim_1 samples $n-2$ random values $\{r_2, ..., r_{n-1}\}$ and let $r_n = r_2 \oplus ... \oplus r_{n-1}$. Sim_1 queries $\mathcal{F}_{DMPF}$ with input $key^{0'}_2$ and each elements x'_k in set X'_2, and receives evaluation result ω'_k in return. Then, Sim_1 queries $\mathcal{F}_{OKVS}$ for encoding with input $(x'_k, \beta'_k \oplus \omega'_k)$ and perform XOR operation on the encoding results and r_2. It gets the encoding S'_2 in return. By the security of $\mathcal{F}_{OKVS}$ and the randomness of r_i, the code S'_2 is computationally indistinguishable with the real view. Therefore, the simulated view from P_1 and P_2 is computationally indistinguishable with the real view.
In the way described above, Sim_1 simulates the rest $(S'_i, key^{1'}_i)$ for $P_3, ..., P_n$ and forms the ideal view. It is trivial to prove the view in a hybrid model by substituting the ideal functionality with the real protocol one by one. Now we have $\{Sim_1(1^\lambda, input(X_1))\}_{X_1 \subseteq \{0,1\}^*, \lambda} \overset{c}{\equiv} \{view^{\Pi}_1(X_1, \lambda)\}_{X_1 \subseteq \{0,1\}^*, \lambda}$.

3) An adversary corrupts the set C_1 and $P1 \notin C_1$.
By merging the corrupted parties, the situation is similar to the first corruption strategy where only one party is corrupted except P_1. Therefore, we omit the detailed proof of this case. It should be noted that the protocol will be boiled to the two-party PSI when $|C_1| = n - 1$, where the honest P_1 gets the intersection.
We denote the intersection that the ideal functionality computes by $I' = (\cap X_{i,P_i \in P_H}) \bigcap (\cap X_{j,P_j \in P_{C_1}})$. The corruption set C_1 will infer some information via the output $I' = I = (\cap X_{i,P_i})$. That is to say, when $P_H = P_1$ and one element $x_C \in \cap X_{j,P_j \in P_{C_1}}$, the adversary will learn whether P_1 input x_C or not by observing the output I. The extra information inferred from the intersection is relatively close to the number of parties in the ideal functionality, which is not caused by the protocol security. For any elements $x_h \in X_{i,P_i \in P_H}$ and $x_h \notin \cap X_{j,P_j \in P_{C_1}}$, the simulator generates indistinguishable view using random numbers.

4) An adversary corrupts the set C_2 that $P_1 \in C_2$.
In this corruption strategy, no honest party gets the intersection and P_1 is curious and can perform local computations. In the case where the Leader is corrupted, we assume that the adversary does not corrupt more than $n-3$ Clients simultaneously.
When $|C_2| < n - 1$, after the protocol, P_1 learns the intersection $I = \cap X_{i,P_i}$. With assistance of C_2, P_1 knows the intersection $I_{C_2} = \cap X_{j,P_j \in P_{C_2}}$. Therefore, P_1 could locally compute the set $I' = I_{C_2}/I$, which essentially computes the set $I' = X_1 \cap \bigcap X_{j,P_j \in P_H}$. We define the leakage to provide formal proof:

$$I_\ell = I'/I = (X_1 / \bigcap X_{i,P_i \in C_2}) \cap X_{j,P_j \in P_H}.$$

By referring to the first two corruption strategies, it is easy to simulate the view of the party P_n. $view^{\Pi}_n(X_i, \lambda)$ consists of input set X_n. Given the output I and the leakage I_ℓ, the simulator Sim_1 first samples $m - |I| - |I_\ell|$ random elements and forms a set $X'_1 = \{I, I_\ell, R_1, R_2, ..., R_{n-|I|-|I_\ell|}\}$. Similar to the second corruption strategy, Sim_1 queries $\mathcal{F}_{DMPF}$ and $\mathcal{F}_{OKVS}$ to generates blinded OKVS encodings S'_i and DMPF keys $key^{1'}_i$. Therefore, $\{Sim_1(1^\lambda, input(X_{i,P_i \in C_2}), I, I_\ell)\}_{X_i \subseteq \{0,1\}^*, I, I_\ell, \lambda} \overset{c}{\equiv} \{view^{\Pi}_1(X_{i,P_i \in C_2}), \lambda\}_{X_i \subseteq \{0,1\}^*, I, \lambda}$.

V. Multi-Party Updatable PSI

In this section, in the semi-honest model, we construct an efficient multi-party updatable PSI protocol that securely computes the functionality in Fig. 2.

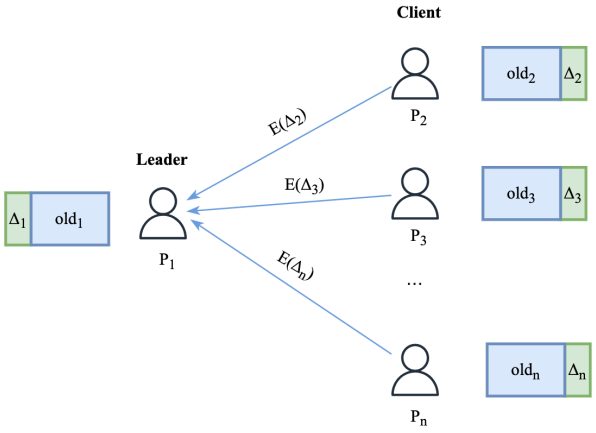

Fig. 5: Communication pattern in MUPSI

A. An Overview

From the perspective of communication patterns, our MUPSI protocol also uses the star structure, as shown in Fig. 5, where P_1 acts as the honest central core Leader, while the remaining parties act as Clients. This structure allows our MUPSI protocol to be highly scalable because the complexity for each party except for the central leader remains independent of the total number of parties, rendering $O(m\lambda)$ computation and communication complexity. We can also notice that both the computational and communication complexity of clients only grow with the set updates instead of the entire sets. Our protocols compare favorably when the total set size is sufficiently large, the new updates are sufficiently small, or in networks with low bandwidth.

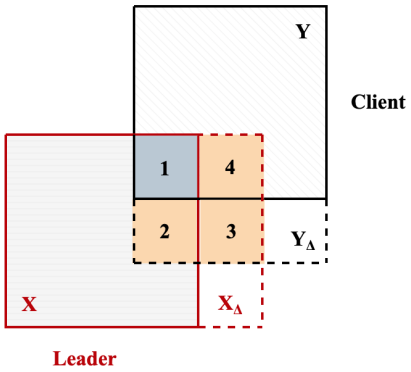

Fig. 6: Schematic diagram of two-party UPSI set update.

We first take the two-party UPSI between Leader P_1 and Client P_2 as an example to introduce the process of UPSI. For simplicity, let both parties hold the original private sets X and Y, as well as the update sets X_Δ and Y_Δ, respectively. In updatable PSI, the parties have already learned $I = X \cap Y$ of the old sets, and they would like to learn the updated intersection $I_d = (X \cup X_\Delta) \cap (Y \cup Y_\Delta)$, with no leakage about X, X_Δ, Y, Y_Δ. Further, we can see that:

$$\begin{aligned}&(X \cup X_\Delta) \cap (Y \cup Y_\Delta)\\ &= (X \cap Y) \cup (X \cap Y_\Delta) \cup (X_\Delta \cap Y) \cup (X_\Delta \cap Y_\Delta).\end{aligned}$$

Notice that $(X \cap Y)$ is known, so the problem becomes learning:

$$I_{update} = (X \cap Y_\Delta) \cup (X_\Delta \cap Y) \cup (X_\Delta \cap Y_\Delta).$$

We have noticed that $(X_\Delta \cap Y_\Delta)$ and $(X_\Delta \cap Y)$ cannot be calculated directly. Specifically, in the two-party updatable PSI illustrated in Fig. 6, part 2 (i.e., $X \cap Y_\Delta$) can be computed by the Leader independently, since the Leader can derive it from the final intersection result I_{update} by computing $I_{update} \cap X$, making this partial information inherently leakable. In contrast, parts 3 and 4 together constitute $X_\Delta \cap (Y \cup Y_\Delta)$ and must be computed jointly by the Leader, rather than separately. This is because, from the Leader's perspective, both the Client's original set X and the updated subset X_Δ are confidential, and thus the individual relations between the Leader's updated set and each of X and X_Δ must not be revealed.

We analyze the set I_{update} from the perspective of the Leader and divide the set into two disjoint parts. The first part is a subset of the original set, and the second part is a subset of the new set. Specifically, we define:

$$I_{old} = (X - X \cap Y) \cap Y_\Delta,$$
$$I_{new} = X_\Delta \cap (Y \cup Y_\Delta).$$

In Fig. 6, we present a visual example of a two-party set update to illustrate the relationships between I_{update}, I_{new}, and I_{old}. The relationship can be expressed as:

$$I_{update} = 2 \cup 3 \cup 4,$$
$$I_{old} = 2,\ I_{new} = 3 \cup 4.$$

Based on the above analysis of the two-party updatable PSI, for multi-party updatable PSI, let each party P_i holds the old set X_i and the update set X_i^Δ. We can regard the MUPSI as the process of calculating the intersection between the leader set and the aggregation of multiple client sets, i.e.,

$$\begin{aligned}I_{update} &\stackrel{def}{=} (X_1 \cap \bigcap_{i=2}^{n} X_i^\Delta) \cup (X_1^\Delta \cap \bigcap_{i=2}^{n} X_i)\\ &\cup (X_1^\Delta \cap \bigcap_{i=2}^{n} X_i^\Delta),\\ I_{old} &\stackrel{def}{=} (X_1 - \bigcap_{i=1}^{n} X_i) \cap \bigcap_{i=2}^{n} X_i^\Delta,\\ I_{new} &\stackrel{def}{=} X_1^\Delta \cap \bigcap_{i=2}^{n} (X_i \cup X_i^\Delta).\end{aligned}$$

Our MUPSI protocol will calculate the intersections I_{old} and I_{new} separately.

B. MUPSI protocol

In this section, we provide a detailed description of our MUPSI protocol. Compared with performing a full PSI on the updated sets, our protocol supports reusing results from previous rounds, thereby reducing communication and computation overhead. Specifically, our MUPSI scheme does not require recomputing the OKVS encoding for the entire set after each update. Each Client only needs to compute and send the OKVS encoding and FSS keys for the updated subset. Leader then integrates the new encodings with those from previous rounds to compute the updated intersection.

When the Leader computations the intersection for day d, it already possesses the OKVS encodings for each Client's set from day $d-1$, which were received during the previous round's computation.

During the computation of I_{old} on day d, the parties invoke our MPSI protocol, and the Leader receives from each Client the OKVS encoding for its updated subset. The Leader retains these encodings for use in computing I_{new}.

For the computation of I_{new}, the Leader takes the OKVS encodings of the Clients' updated subsets from day d (received in the current round) and XORs them with the OKVS encodings of the Clients' full sets from day $d-1$ (retained from the previous round). The result of this XOR operation yields the integrated OKVS encoding representing each Client's full set for day d. The Leader then uses this integrated encoding to compute the intersection for day d, and also retains it as a prerequisite for the intersection computation of day $d+1$, thereby achieving result reuse.

Our multi-party UPSI protocol is shown in Fig. 7. We designate P_1 to be the leader and the recipient of the intersection.

Initially, each party holds their private set X_i^0 and we set the initial intersection $I_0 = \emptyset$.

On day d, Leader P_1 has already learned the intersection I_{d-1} and the OKVS encoding S_i^{d-1} of each Client P_i from the previous $d-1$ days. All parties hold their private update sets and calculate $X_i^d = X_i^{d-1} \cup X_i^{\Delta}$.

Computing I_{old}: In this step, P_1 calculates the intersection of its old set X_1^{d-1} with the update sets of all Clients $\bigcap_{i=2}^{n} X_i^{\Delta}$. All parties invoke our protocol $\Pi_{MPSI}^{m,n}$ shown in Fig. 4 on the sets X_1^{d-1} and $X_i^{\Delta} (i \in [2,n])$, without performing zero-sharing operations in the preprocessing phase. Specifically, Clients execute the online phase of our MPSI protocol with all random shares r_i set to 0. Then Leader P_1 obtains the result I_{old}. During this process, each Client sends the OKVS encoding of their update sets to the Leader.

Computing I_{new}: It can be noticed that Leader P_1 has already received the OKVS encoding S_i^{d-1} of each Client P_i from the previous $d-1$ days and S_i^{Δ} of each Client on day d. Then P_1 needs to aggregate the OKVS encoding by $S_i^d = S_i^{d-1} \oplus S_i^{\Delta}$, and its correctness stems from the linearity of OKVS. Next, P_1 decodes each code S_i^{Δ} by its elements x_k in the update set X_1^{Δ} and performs an AND operation on the results decoded from the same element x_k. Finally, P_1 checks whether the result is 0 and then computes the intersection.

Output generation: The output of our MUPSI protocol on Day d is $I_d = I_{d-1} \cup I_{old} \cup I_{new}$. Leader P_1 also needs to compute the aggregation of OKVS encoding from each Client by $S_i^d = S_i^{d-1} \oplus S_i^{\Delta}$ in order to calculate the intersection for the next day.

C. Threat Model

In this section, we clarify the trust assumption of our protocol. Unlike our base MPSI protocol, our MUPSI scheme $\Pi_{MUPSI}^{M,m,n}$ assumes an honest Leader, who is trusted by all participating parties as a semi-honest, non-colluding coordinator. Under this assumption, the protocol achieves efficient set updates and collusion resistance against any coalition of up to $n-1$ participants.

We first clarify what information would be additionally leaked if the MUPSI leader is honest-but-curious. Specifically, the Leader would gain access to certain intermediate results during the protocol execution.

- The per-Client intersection between the Leader's updated set and each Client's full set. The functionality only allows the Leader to learn the overall intersection of its updated set with the union of all Clients' full sets.
- The Leader would separately obtain the intersection between its updated set and each Client's old set as well as its updated set. The functionality only allows the Leader to learn the overall intersection with all Clients' full sets (i.e., denoted as $3 \cup 4$ in Fig. 6), rather than the disjoint components 3 and 4 individually.

To provide a broader perspective on this trust assumption, we compare it with the trust models adopted in existing schemes. To the best of our knowledge, no existing multi-party PSI scheme currently supports the updatable feature in the same manner. While SOTA non-updatable schemes (e.g., [24], [29]) can operate without relying on an honest Leader, the communication and computation overhead incurred by re-executing the full protocol from scratch grows with the entire dataset size. In this context, our MUPSI scheme trades a trusted honest Leader for a persistent efficiency gain that accumulates over multiple update rounds.

The above trade-off makes our MUPSI well-suited for large-scale data collaboration scenarios that demand high performance and have access to a trusted coordinator. Specifically, we believe this assumption can be satisfied via either of the following two approaches, depending on the actual deployment environment.

Limited leakage is acceptable in practice as the Leader is bound by multiple constraints. In multi-center clinical studies or joint enterprise risk control, participating parties are typically of comparable standing and credibility. A reputable institution, such as the lead hospital or a leading enterprise, can serve as the Leader. Even if the Leader deviates from the semi-honest assumption, the private information it can obtain remains strictly limited by the protocol design, and the institution is bound by industry ethics, data protection regulations, or consortium bylaws. Therefore, this limited leakage is acceptable or manageable in practice.

An honest Leader can be elected from the participants. For stringent scenarios that demand zero leakage, a group of semi-honest parties can collaboratively elect a Leader through a decentralized and publicly verifiable Byzantine Fault-Tolerant (BFT) leader election protocol before the protocol execution [34]. Such protocols ensure that, under the assumption of at most one-third malicious parties, an honest Leader is elected with high probability and all parties reach consensus on the election result. Therefore, in our semi-honest model, this mechanism can elect a Leader that satisfies our trust assumption.

D. Complexity Analysis

In this section, we present a theoretical complexity analysis of our MUPSI protocol $\Pi_{MUPSI}^{M,m,n}$.

The communication overhead of our MUPSI protocol is also divided into two parts, which remain identical to those of our $\Pi_{MPSI}^{m,n}$. For the Client, the communication complexity is $O(m\lambda)$, and for the Leader, the communication complexity is $O(mn\lambda)$. We note that the communication complexity of

$\Pi_{MUPSI}^{M,m,n}$

Initialization: Each party P_i holds initial private set X_i^0. Initalize $I_0 = \emptyset$.

Day d: At the end of Day $d-1$, each P_i holds X_i^{d-1}. At the same time, Leader P_1 has already learned the private set intersection $I_{d-1} = \bigcap_{i=i}^{n} X_i^{d-1}$ and the OKVS encoding S_i^{d-1} of each Client from the previous $d-1$ days.

1) **Update sets:**
- Each party P_i holds the private update sets X_i^{Δ}. Update $X_i^d = X_i^{d-1} \cup X_i^{\Delta}$.

2) **Calculate the intersection** $I_{old} = X_1^{d-1} \cap \bigcap_{i=2}^{n} X_i^{\Delta}$ **:**
- Leader P_1 and all the Client invoke the protocol $\Pi_{MPSI}^{m,n}$ (without preprocessing) on the sets X_1^{d-1} and $X_i^{\Delta} (i \in [2, n])$. Specifically, Clients execute the online phase of our MPSI protocol with all random shares r_i set to 0. Then P_1 obtains the result I_{old}. During this process, Client P_i generates OKVS encoding S_i^{Δ} and DMPF key Δkey_i^1 on update set X_i^{Δ} and sends them to Leader.

3) **Calculate the intersection** $I_{new} = X_1^{\Delta} \cap \bigcap_{i=2}^{n} X_i^{d}$ **:**
- For each Client P_i, Leader P_1 aggregates their OKVS encoding: $S_i^d = S_i^{d-1} \oplus S_i^{\Delta}$.
- For each elements x_k in X_1^{Δ}, P_1 retrieves $result_i^k$ from S_i^d by computing $result_i^k = Decode(S_i^d, x_k)$.
- P_1 calculates $\omega_k = AND_{i=2}^{n}\ result_i^k$. If $\omega_k \neq 0$, P_1 proceeds to include element x_k in the intersection set I_{new}.

4) **Output generation:**
- P_1 output the intersection set of Day d: $I_d = I_{d-1} \cup I_{old} \cup I_{new}$.
- Leader P_1 aggregates the OKVS encoding of each Client: $S_i^d = S_i^{d-1} \oplus S_i^{\Delta}$.

Fig. 7: Our multi-party updatable private set intersection $\Pi_{MUPSI}^{M,m,n}$.

the protocol is only related to the update set size m, and not to the entire set size M.

The computational overhead in our MUPSI scheme comprises two key aspects. When computing I_{old}, each Client only requires to calculate the DMPF keys and OKVS encoding for the update set, with a complexity of $O(m\lambda)$, which solely correlates to the size m of the update set. For the Leader, it necessitates OKVS queries and DMPF evaluations, with complexities of $O(M\lambda)$ and $O(Mn)$ respectively. When computing I_{new}, the Leader needs to perform OKVS queries, with a complexity of $O(mn\lambda)$. Consequently, the total computational overhead for the Leader is $O(M(n+\lambda) + mn\lambda)$.

E. Security Proof

In this section, we begin by proving the correctness of our MUPSI protocol $\Pi_{MUPSI}^{M,m,n}$ and subsequently provide a security verification.

1) Correctness: This section proves the functional correctness of the proposed multi-party UPSI protocol, that is, when all parties execute the protocol honestly, the Leader can accurately calculate the intersection of all party sets.

Theorem 3. *Assume all parties execute the protocol honestly across multiple time epochs. Then the set output by the Leader is exactly the intersection I_t of the input sets of all n parties after the update computation of epoch t.*

Proof: As analyzed previously, the update set intersection I_d on day d can be divided into two parts: I_{d-1} and I_{update}. I_{d-1} is known, and I_{update} can also be divided into two parts: I_{old} and I_{new}. The correctness of I_{old} is guaranteed by the correct operation of the MPSI protocol. Now we proceed with the proof of the correctness of the calculation of I_{new}.

From Lemma 1, it can be inferred that after updating the set, the corresponding OKVS encodings can be merged. That is, if $S_{new} = S_{old} \oplus S_{\Delta}$, then $Decode(S_{new}, k) = Decode(S_{old}, k) \oplus Decode(S_{\Delta}, k)$. Given a deterministic-output OKVS, where decoding the same non-member query x always returns the same pseudorandom value, we consider the following three cases.

a) x is in the old set.
x is in the old set of P_i. When P_1 calculates the intersection with Client P_i, as shown in Fig. 6, x is in Part 4, which is the intersection of X_1^{Δ} and X_i. Then there is:

$$Decode(S_{old}, x) = value_x,$$
$$Decode(S_{\Delta}, x) = random(x),$$
$$Decode(S_{new}, x) = value_x \oplus random(x).$$

b) x is in the update set.
x is in the update set of P_i, as shown in Fig. 6, x is in

Part 3, which is the intersection of X_1^{Δ} and X_i^{Δ}. Then there is:

$$Decode(S_{old}, x) = random(x),$$
$$Decode(S_{\Delta}, x) = value_x,$$
$$Decode(S_{new}, x) = random(x) \oplus value_x.$$

c) x is not in the intersection. x is not in the set of P_i, then we have:

$$Decode(S_{old}, x) = random(x),$$
$$Decode(S_{\Delta}, x) = random(x),$$
$$Decode(S_{new}, x) = random(x) \oplus random(x) = 0.$$

Leader P_1 performs OKVS decoding calculations on each element in its update set X_1^{Δ}, and then performs an AND operation on the results. If one of the results is 0, indicating that x is not in the set of the Client P_i, P_1 will ultimately obtain 0, and the element x is not an intersection element. Conversely, x is considered to be an intersection element.

2) Security: Since we have designated Leader as the trusted party, our analysis of corruption strategies is similar to the first and third cases in Chapter IV-B.

Theorem 4. *Given an ideal functionality, the MUPSI protocol in Fig 7 securely computes the functionality $\mathcal{F}_{MUPSI}^{M,m,n}$ (Fig. 2) in the presence of semi-honest adversaries, assuming the Leader is honest, as described in Definition 1.*

When an adversary corrupts the set C and Leader $P_1 \notin C$, simulating the perspective script of corrupted party P_j is straightforward because P_j does not output from either the protocol $\Pi_{MUPSI}^{M,m,n}$ or the functionality $\mathcal{F}_{MUPSI}^{M,m,n}$. It should be noted that the protocol will be reduced to the two-party UPSI when $|C| = n-1$, where the honest party P_1 obtains the intersection. Specifically, $view_j^{\Pi}(X_{j,P_j \in C}, \lambda)$ consists of the input private sets X_j. Therefore the simulated view $\left\{Sim_j(1^{\lambda}, input(X_{j,P_j \in C}))\right\}_{X_j \subseteq \{0,1\}^*, \lambda}$ is computationally indistinguishable with the realistic view $\left\{view_j^{\Pi}(X_{j,P_j \in C}), \lambda\right\}_{X_j \subseteq \{0,1\}^*, \lambda}$.

VI. Performance Evaluation

Our evaluations focus on answering the following questions:

- Q1: Does our MPSI protocol $\Pi_{MPSI}^{m,n}$ support a large number of participants and have higher efficiency than the existing solution?
- Q2: Does our MUPSI protocol $\Pi_{MUPSI}^{M,m,n}$ have higher efficiency than the existing MPSI solution in updatable scenarios?

To answer the above questions, we implement our MPSI protocol $\Pi_{MPSI}^{m,n}$ in Fig. 4 and our MUPSI protocol $\Pi_{MUPSI}^{M,m,n}$ in Fig. 7 using Rust.

In this chapter, we evaluate the running time and communication cost of our implementation of MPSI and MUPSI with up to 140 participants and 2^{20} set size. We compare our constructions with the SOTA works [24][1], [27][2], [28][3] and [29][4]. We proceed with the implementation of our protocols utilizing the DMPF and OKVS source code in [30] to assess the practical computation and communication complexity.

To ensure a standardized evaluation environment, all experiments are conducted on a 6*16 core Intel Xeon 2.90 GHz CPU and 128GB of RAM. We evaluate our protocols in simulated networks. In the LAN setting, there is a 0.02 ms round-trip latency and 20 Gbps network bandwidth, while the WAN setting has a 96 ms round-trip latency and 200 Mbps network bandwidth. To enable parallel computation and communication among multiple participants on a single machine, we allocate a separate thread for each participant to perform computations. In our protocol, the Client only needs to send messages to the Leader and does not need to receive any messages. The threads that send messages from the Client can exist simultaneously. Since the machine has $6 * 16 * 2 = 192$ threads, it can handle the parallel tasks where $n = 140$.

All our evaluations are performed under statistical security parameter $\lambda = 40$, computational security parameter $\kappa = 128$, and elements x_i^k of length 128 bit. All sets have at least 20% intersection elements. The communication cost refers to the total data that a participant sent.

A. Evaluation of Our MPSI Construction

To understand the scalability of our MPSI protocol, we evaluate our implementation of $\Pi_{MPSI}^{m,n}$ on the range of $m \in \{2^{12}, 2^{16}, 2^{20}\}$ and $n \in \{20, 50, 80, 110, 140\}$. We report the detailed performance results in Table II, showing the running time in different network settings and the communication cost of Leader and Client.

It can be observed that as the number of participants increases from 20 to 140, the running time and communication cost of the Client remain unchanged. For the Leader, the running time increases by 5.2–13.3× in the LAN setting and by 6.0–6.5× in the WAN setting, while the communication cost increases by 6.9–7.3×. These increases are significantly lower than the 256× increase in the number of participants, demonstrating the excellent scalability of $\Pi_{MPSI}^{m,n}$ with the number of participants.

B. Comparison With Other MPSI Works

To answer question **Q1**, we compare our MPSI construction with the SOTA works [27] (using "O-ring" and "K-star"), [28] (using "DCZ+25"), [29] (using "GGA+26") and [24] (using "GLW+24"). We evaluate these schemes on the set size $m \in \{2^{12}, 2^{16}, 2^{20}\}$ on range of value $(n, t) \in \{(4, \{1, 3\}), (10, \{1, 4, 9\}), (15, \{1, 4, 7, 14\})\}$.

We compare our protocol with these works under WAN settings, where high latency and limited bandwidth make round complexity a critical performance factor. Since our protocol

[1]https://github.com/orzcy/BZS-MPSI
[2]https://github.com/private-panda/oring
[3]https://github.com/real-world-cryptography/MPSO
[4]https://github.com/Ame-ReIori/mPSI

TABLE II: Running time (in seconds) and communication cost (in MB) of $\Pi_{MPSI}^{m,n}$

Set Size m		2^{12}					2^{16}					2^{20}				
Number of Participants n		20	50	80	110	140	20	50	80	110	140	20	50	80	110	140
Running Time in LAN setting	Leader	0.604	1.262	2.460	3.036	4.211	1.503	6.206	11.46	14.63	20.11	10.64	18.36	31.95	47.67	55.54
	Client	0.065	0.065	0.065	0.065	0.065	0.291	0.291	0.291	0.291	0.291	1.476	1.476	1.476	1.476	1.476
Running Time in WAN setting	Leader	2.151	4.384	6.193	10.85	14.09	6.51	15.84	27.35	36.51	42.93	20.82	38.65	62.17	97.88	125.8
	Client	0.284	0.284	0.284	0.284	0.284	0.814	0.814	0.814	0.814	0.814	3.650	3.650	3.650	3.650	3.650
Communication Cost	Leader	36.49	94.11	151.7	209.3	266.7	304.4	761.1	1207	1654	2125	2297	5925	9554	13181	16809
	Client	1.920	1.920	1.920	1.920	1.920	15.22	15.22	15.22	15.22	15.22	120.8	120.8	120.8	120.8	120.8

TABLE III: Running time in WAN setting (in seconds) comparison between our protocol and other MPSI protocols. The best result is marked in Green and the second best result is marked in Blue.

Setting		Running Time in WAN setting											
Protocol		O-ring [27]		K-star [27]		DCZ+25 [28]	GGA+26 [29]		GLW+24 [24]		Ours	
m	(n,t)	Client	Total	Client	Total	Total	Client	Total	Client	Total	Client	Total
2^{12}	(4,1)	0.474	2.126	0.473	2.131	1.950	0.415	2.806	0.238	1.331	0.284	0.760
	(4,3)	3.422	3.649	3.418	3.643	1.950	0.267	2.819	0.238	1.331	0.284	0.760
	(10,1)	1.309	3.539	1.308	3.562	4.382	0.452	2.808	0.246	1.386	0.284	1.357
	(10,4)	3.754	5.693	3.754	5.603	4.382	0.468	2.943	0.246	1.386	0.284	1.357
	(10,9)	5.400	6.030	5.435	5.925	4.382	0.311	3.688	0.246	1.386	0.284	1.357
	(15,1)	5.561	8.273	5.561	8.314	4.860	0.453	2.808	0.251	1.412	0.284	1.696
	(15,4)	9.470	11.74	9.403	11.71	4.860	0.470	2.929	0.251	1.412	0.284	1.696
	(15,7)	9.552	11.81	9.647	11.70	4.860	0.489	2.259	0.251	1.412	0.284	1.696
	(15,14)	11.92	12.95	12.04	12.76	4.860	0.456	6.285	0.251	1.412	0.284	1.696
2^{16}	(4,1)	0.842	3.442	0.850	3.853	3.555	1.213	3.620	0.631	2.625	0.814	2.440
	(4,3)	4.339	5.238	4.342	5.201	3.555	1.064	4.047	0.631	2.625	0.814	2.440
	(10,1)	1.681	6.867	1.700	7.857	6.518	1.495	3.909	0.913	2.925	0.814	2.887
	(10,4)	4.736	9.810	4.786	8.708	6.518	1.922	5.193	0.913	2.925	0.814	2.887
	(10,9)	8.982	12.62	9.166	10.89	6.518	3.140	10.21	0.913	2.925	0.814	2.887
	(15,1)	5.941	14.10	5.952	14.69	8.750	1.749	4.172	1.086	3.129	0.814	4.926
	(15,4)	11.31	19.84	11.52	15.96	8.750	2.163	5.432	1.086	3.129	0.814	4.926
	(15,7)	12.60	21.04	12.75	17.76	8.750	2.981	8.171	1.086	3.129	0.814	4.926
	(15,14)	20.66	26.56	21.04	23.51	8.750	4.957	20.31	1.086	3.129	0.814	4.926
2^{20}	(4,1)	12.07	22.01	13.38	20.57	23.81	7.561	14.45	6.023	14.84	3.650	8.070
	(4,3)	24.76	34.49	25.32	31.90	23.81	9.582	25.24	6.023	14.84	3.650	8.070
	(10,1)	13.71	51.34	15.33	33.70	47.65	12.54	19.50	11.78	20.56	3.650	14.57
	(10,4)	44.50	82.15	50.18	64.34	47.65	17.95	40.96	11.78	20.56	3.650	14.57
	(10,9)	90.80	126.4	91.53	106.0	47.65	26.01	97.78	11.78	20.56	3.650	14.57
	(15,1)	18.39	78.51	20.46	47.83	70.00	17.12	24.16	14.47	24.00	3.650	18.15
	(15,4)	67.53	128.5	74.43	94.46	70.00	22.39	45.51	14.47	24.00	3.650	18.15
	(15,7)	98.48	164.8	111.7	132.3	70.00	28.65	85.19	14.47	24.00	3.650	18.15
	(15,14)	204.0	261.3	204.5	226.0	70.00	40.03	244.7	14.47	24.00	3.650	18.15

requires only one-pass online communication, this is the scenario where its advantage is most pronounced. The comparison results of running time in WAN setting are presented in Table III. It can be observed that under the WAN setting, our protocol achieves the optimal running time in most cases. Compared with [27], our protocol outperforms O-ring and K-star with $2.1-14.4\times$ and $2.1-12.5\times$ speedup, respectively. Compared with DCZ+25, our protocol is 1.5–$3.9\times$ faster. Compared with GGA+26, our protocol is 1.4–$13.5\times$ faster. Compared to GLW+24, our protocol achieves the suboptimal running time in a few cases. However, there is a critical weakness in the security model of the GLW+24 protocol: its security relies on the idealized assumption that two special parties never collude. Once these two special parties are simultaneously corrupted by an adversary, the protocol's security completely collapses. In contrast, our protocol offers much more robust security: even if the Leader is corrupted, the adversary still needs to corrupt an additional $n-2$ Clients to break the security. The attack threshold is therefore substantially higher than that of the GLW+24 protocol. This demonstrates that our protocol achieves competitive performance while providing stronger security.

When compared with the MPSI protocol in GLW+24, which represents the best-performing work among current SOTA protocols, our $\Pi_{MPSI}^{m,n}$ shows a significant advantage in running time under WAN settings, particularly for large

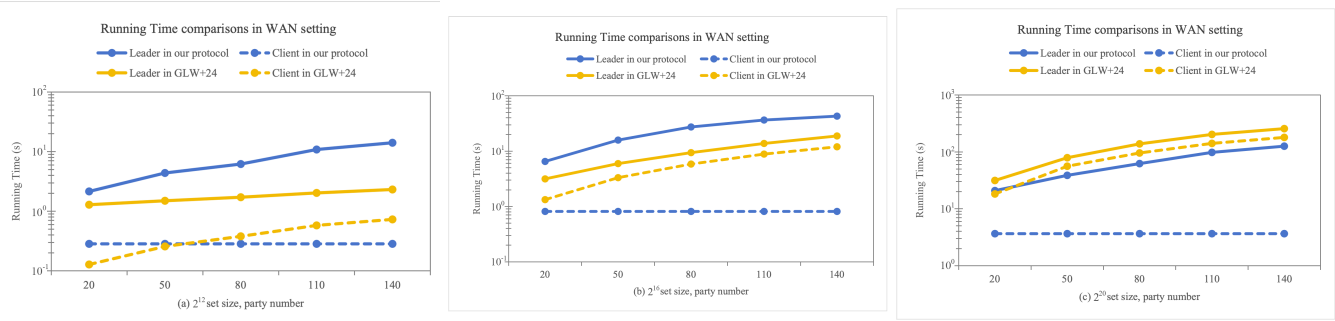

Fig. 8: Running time comparison with different party numbers and set sizes in WAN setting.

datasets and a higher number of parties. Fig. 8 presents the running time comparison between our protocol and GLW+24, showing the performance of both the Leader and Client as the number of participants varies, for datasets of different sizes.

First, as the number of participants increases, the running time of the Client in our protocol remains unchanged, which is evident in the evaluation results for all three types of dataset sizes. For 2^{12}-sized datasets, the running time of the Client in our protocol becomes increasingly advantageous when the number of participants reaches 50 or more (Fig. 8(a)). For 2^{16}-sized datasets, our protocol consistently achieves the best Client running time across all participant counts from 20 to 140 (Fig. 8(b)). For the 2^{20}-sized datasets, our protocol achieves optimal running time on both the Client and Leader across the entire range of participant counts (Fig. 8(c)). In particular, when the dataset size is 2^{20} and the number of participants is 140, our protocol achieves a $49.1\times$ speedup over the GLW+24 scheme in terms of Client running time. Overall, the performance improvement of our $\Pi^{m,n}_{MPSI}$ becomes more significant as the number of participants and dataset size increase. Therefore, in practical applications, our protocol is highly competitive for large-scale multi-party private data computation in WAN environments.

These results indicate that $\Pi^{m,n}_{MPSI}$ can be effectively utilized in most real-world scenarios. We attribute the advantages of $\Pi^{m,n}_{MPSI}$ to the following factors:

- The protocol compresses all communication into one-pass transmission from the Client to the Leader, eliminating the latency accumulation caused by multiple rounds of interaction. As a result, the increase in running time in the WAN setting is significantly smaller than that of other protocols.
- Utilizing the linear property of OKVS, the Leader can complete the merging of all OKVS encodings with just one XOR operation, avoiding the network dependency caused by multiple interactive decodings.
- All clients can compute in parallel.

C. Evaluation and Comparison of MUPSI Construction

To demonstrate the updatable property of our protocol, we consider the setting where all parties begin with an empty set to which m elements are added each day. Our evaluation represents the performance of the protocols on day $(\frac{M}{m})$ where the size of each party's set reaches M.

To answer the question **Q2**, we selected two of the best protocols GLW+24 and GGA+26 from the current SOTA works and compared our $\Pi^{M,m,n}_{MUPSI}$ with them, where on day $(\frac{M}{m})$, the parties run MPSI on their full sets of size M. In our evaluation, our protocol has more advantages when increasing the gap between M and m. As M increases (e.g., for million-size sets), we expect our protocol to be dominant for more network settings, larger m values and more participants numbers.

We evaluate our implementation of MUPSI protocol on the range of $M \in \{2^{18}, 2^{20}, 2^{22}\}$, $m \in \{2^6, 2^8, 2^{10}\}$ and $n \in \{20, 80, 140\}$. Notably, the protocol in GGA+26 involves a corruption threshold t that affects performance. Since our protocol and GLW+24 have no such tunable parameter, we set $t = n/2$ for a fair comparison. We report the detailed performance results in Table IV, showing the running time in different network settings and communication cost of Leader and Client.

It can be observed that the Client consistently achieves optimal running time and communication cost under both LAN and WAN settings, which depend only on the size m of the small update set and do not increase with the number of participants. Compared with other works, the running time of the Client in our protocol is reduced by an order of magnitude. In the WAN setting, we note that our protocol achieves the optimal total running time and communication cost, outperforming GLW+24 and GGA+26 with $2.5 - 46.5\times$ and $2.8 - 49.5\times$ speedup, respectively. In conclusion, given these communication and computation trade offs, our protocol $\Pi^{M,m,n}_{MUPSI}$ performs best with more realistic network configurations with lower network bandwidth.

VII. Conclusion

We introduced a novel multi-party PSI protocol with one-pass online intersection based on the OKVS and DMPF primitives. Through theoretical analysis and experimental comparison, our MPSI protocol exhibits significant advantages in round efficiency, especially in network settings with limited bandwidth and high latency. When the number of parties and the set size reach a certain scale, our MPSI protocol demonstrates more competitive performance in terms of both

TABLE IV: Running time (in seconds) and communication cost (in MB) of $\Pi_{MUPSI}^{M,m,n}$. Cells in Blue indicate the state-of-the-art performance, and those in Green indicate that our protocol performs better.

Old Set Size M	Updates Set Size m	Protocol	Number of Participants n	Running Time				Communication Cost	
				LAN		WAN			
				Leader	Client	Leader	Client	Leader (Total)	Client
2^{18}	-	GLW+24 [24]	20	0.824	0.199	12.27	4.914	10.86	11.53
			80	1.933	0.440	46.74	23.71	10.86	11.53
			140	2.506	0.671	84.03	49.26	10.86	11.53
	-	GGA+26 [29]	20	0.601	0.113	13.54	0.362	1300	5.440
			80	0.896	0.113	53.61	0.896	5767	5.440
			140	1.103	0.113	93.39	1.296	11874	5.440
	2^6	Ours	20	0.110	0.043 (ms)	0.713	0.792 (ms)	2.745	0.142
			80	0.456	0.043 (ms)	1.690	0.792 (ms)	11.41	0.142
			140	0.807	0.043 (ms)	3.252	0.792 (ms)	20.07	0.142
	2^8		20	0.432	0.092 (ms)	1.861	2.812 (ms)	10.83	0.577
			80	1.814	0.092 (ms)	5.814	2.812 (ms)	45.63	0.577
			140	3.133	0.092 (ms)	9.133	2.812 (ms)	80.32	0.577
	2^{10}		20	1.730	0.477 (ms)	4.809	17.14 (ms)	43.90	1.440
			80	5.142	0.477 (ms)	10.14	17.14 (ms)	130.5	1.440
			140	6.011	0.477 (ms)	16.01	17.14 (ms)	152.2	1.440
2^{20}	-	GLW+24 [24]	20	2.379	0.797	31.40	18.20	44.28	21.72
			80	3.499	1.764	138.1	95.59	44.28	21.72
			140	4.557	2.683	-	-	44.28	21.72
	-	GGA+26 [29]	20	1.658	0.442	39.46	8.520	2601	21.76
			80	2.722	0.442	156.2	21.09	11535	21.76
			140	3.351	0.442	272.1	30.51	23748	21.76
	2^6	Ours	20	0.430	0.040 (ms)	2.430	1.341 (ms)	2.745	0.142
			80	1.845	0.040 (ms)	11.84	1.341 (ms)	11.41	0.142
			140	3.210	0.040 (ms)	43.21	1.341 (ms)	20.07	0.142
	2^8		20	1.753	0.085 (ms)	3.753	4.609 (ms)	10.98	0.577
			80	5.162	0.085 (ms)	12.16	4.609 (ms)	45.63	0.577
			140	6.011	0.085 (ms)	36.01	4.609 (ms)	80.32	0.577
	2^{10}		20	5.348	0.480 (ms)	12.34	17.35 (ms)	43.90	1.440
			80	12.64	0.480 (ms)	17.64	17.35 (ms)	130.5	1.440
			140	18.04	0.480 (ms)	24.04	17.35 (ms)	152.2	1.440
2^{22}	-	GLW+24 [24]	20	13.53	2.188	313.5	70.63	157.6	85.90
			80	17.87	6.046	-	-	157.6	85.90
			140	21.30	9.737	-	-	157.6	85.90
	-	GGA+26 [29]	20	5.036	1.730	115.0	200.5	5576	87.04
			80	8.271	1.730	455.3	496.4	24724	87.04
			140	10.18	1.730	793.2	718.2	50882	87.04
	2^6	Ours	20	1.746	0.037 (ms)	6.746	2.096 (ms)	2.745	0.142
			80	5.190	0.037 (ms)	9.190	2.096 (ms)	11.41	0.142
			140	6.003	0.037 (ms)	20.03	2.096 (ms)	20.07	0.142
	2^8		20	5.348	0.088 (ms)	12.34	3.735 (ms)	10.98	0.577
			80	12.64	0.088 (ms)	16.64	3.735 (ms)	45.63	0.577
			140	18.04	0.088 (ms)	24.04	3.735 (ms)	80.32	0.577
	2^{10}		20	12.69	0.489 (ms)	23.69	16.19 (ms)	43.90	1.440
			80	17.36	0.489 (ms)	26.41	16.19 (ms)	130.5	1.440
			140	25.69	0.489 (ms)	55.69	16.19 (ms)	152.2	1.440

running time and communication overhead. We also proposed a novel multi-party updatable PSI protocol. When the total set size is sufficiently large and the update sets are sufficiently small, our MUPSI protocol can demonstrate its advantages. We provided formal security proof to demonstrate that our protocols are resilient to collusion under the semi-honest model. In the future, more practical security models, such as publicly verifiable cover (PVC) adversaries, and more protocol variants, including multi-party private set intersection cardinality (MPSI-CA) and multi-party threshold private set intersection (MTPSI), will be considered in multi-party PSI and UPSI protocols.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Natural Science Foundation of China under Grant 62472265, Grant 62302280, Grant 62472266 and Grant U25B2031; in part by Shandong Provincial Natural Science Foundation under Grant ZR2026MS1019, Grant ZR2026ZD45, Grant ZR2026QC0726, Grant ZR2025MS1027 and Grant

ZR2023QF133; in part by Young Talent of Lifting Engineering for Science and Technology in Shandong, China under Grant SDAST2025QTB026; and in part by the SDNU Young Academic Elites Development Program. We would also like to thank Yu Chen, Minglang Dong and Xiangfu Song for their constructive comments and valuable insights that helped improve this work.

REFERENCES

[1] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious prf," in *Annual International Cryptology Conference*. Springer, 2020, pp. 34–63.

[2] P. Rindal and M. Rosulek, "Improved private set intersection against malicious adversaries," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2017, pp. 235–259.

[3] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 2, pp. 1–35, 2018.

[4] S. Raghuraman and P. Rindal, "Blazing fast psi from improved okvs and subfield vole," in *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, 2022, pp. 2505–2517.

[5] P. Rindal and P. Schoppmann, "Vole-psi: fast oprf and circuit-psi from vector-ole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 901–930.

[6] A. Bay, Z. Erkin, J.-H. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1–15, 2021.

[7] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty psi and extensions to circuit/quorum psi," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1182–1204.

[8] J. H. Cheon, S. Jarecki, and J. H. Seo, "Multi-party privacy-preserving set intersection with quasi-linear complexity," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 8, pp. 1366–1378, 2012.

[9] C. Hazay and M. Venkitasubramaniam, "Scalable multi-party private set-intersection," in *IACR international workshop on public key cryptography*. Springer, 2017, pp. 175–203.

[10] R. Inbar, E. Omri, and B. Pinkas, "Efficient scalable multiparty private set-intersection via garbled bloom filters," in *International conference on security and cryptography for networks*. Springer, 2018, pp. 235–252.

[11] A. Kavousi, J. Mohajeri, and M. Salmasizadeh, "Efficient scalable multi-party private set intersection using oblivious prf," in *International Workshop on Security and Trust Management*. Springer, 2021, pp. 81–99.

[12] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1257–1272.

[13] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1151–1165.

[14] T. Reinhold, P. Kuehn, D. Günther, T. Schneider, and C. Reuter, "Extrust: Reducing exploit stockpiles with a privacy-preserving depletion system for inter-state relationships," *IEEE Transactions on Technology and Society*, vol. 4, no. 2, pp. 158–170, 2023.

[15] E. Zhang, F.-H. Liu, Q. Lai, G. Jin, and Y. Li, "Efficient multi-party private set intersection against malicious adversaries," in *Proceedings of the 2019 ACM SIGSAC conference on cloud computing security workshop*, 2019, pp. 93–104.

[16] S. Zhang, "Efficient vole based multi-party psi with lower communication cost," *Cryptology ePrint Archive*, 2023.

[17] M. Sheikhalishahi, G. Tillem, Z. Erkin, and N. Zannone, "Privacy-preserving multi-party access control," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, 2019, pp. 1–13.

[18] M. Ion, B. Kreuter, E. Nergiz, S. Patel, S. Saxena, K. Seth, D. Shanahan, and M. Yung, "Private intersection-sum protocol with applications to attributing aggregate ad conversions," *Cryptology ePrint Archive*, 2017.

[19] A. Berke, M. Bakker, P. Vepakomma, K. Larson, and A. Pentland, "Assessing disease exposure risk with location data: A proposal for cryptographic preservation of privacy," *arXiv preprint arXiv:2003.14412*, 2020.

[20] S. Dittmer, Y. Ishai, S. Lu, R. Ostrovsky, M. Elsabagh, N. Kiourtis, B. Schulte, and A. Stavrou, "Function secret sharing for psi-ca: With applications to private contact tracing," *arXiv preprint arXiv:2012.13053*, 2020.

[21] T. Duong, D. H. Phan, and N. Trieu, "Catalic: Delegated psi cardinality with applications to contact tracing," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2020, pp. 870–899.

[22] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epione: Lightweight contact tracing with strong privacy," *arXiv preprint arXiv:2004.13293*, 2020.

[23] D. T. Nguyen and N. Trieu, "Mpccache: privacy-preserving multi-party cooperative cache sharing at the edge," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 80–99.

[24] Y. Gao, Y. Luo, L. Wang, X. Liu, L. Qi, W. Wang, and M. Zhou, "Efficient scalable multi-party private set intersection (-variants) from bicentric zero-sharing," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4137–4151.

[25] S. Zhao, S. Lu, Y. Meng, C. Zhao, S. Jing, Z. Chen, and Q. Xu, "Multi-party private set intersection with one-round online interaction," *IEEE Transactions on Information Forensics and Security*, 2025.

[26] J. Vos, M. Conti, and Z. Erkin, "Sok: Collusion-resistant multi-party private set intersections in the semi-honest model," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 465–483.

[27] M. Wu, T. H. Yuen, and K. Y. Chan, "O-ring and k-star: Efficient multi-party private set intersection," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 6489–6506.

[28] M. Dong, Y. Chen, C. Zhang, Y. Bai, and Y. Cao, "Multi-party private set operations from predicative zero-sharing," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 4079–4093.

[29] Y. Guan, H. Guo, M. H. Au, J. Huo, J. Tan, and Z. Guan, "Practical multi-party private set intersection with reducible zero-sharing," Cryptology ePrint Archive, Paper 2025/2019, 2025.

[30] E. Boyle, N. Gilboa, M. Hamilis, Y. Ishai, and Y. Tu, "Improved constructions for distributed multi-point functions," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 2414–2432.

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

[32] A. Bienstock, S. Patel, J. Y. Seo, and K. Yeo, "Near-optimal oblivious key-value stores for efficient psi, psu and volume-hiding multi-maps," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 301–318.

[33] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Psi from paxos: Fast, malicious private set intersection," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 739–767.

[34] A. Bhangale, C.-D. Liu-Zhang, J. Loss, K. Nayak, and S. Yandamuri, "Leader election with poly-logarithmic communication per party," in *Annual International Cryptology Conference*. Springer, 2025, pp. 37–68.