Efficient Privacy-Preserving User Tracking from Threshold Multi-Party Private Set Intersection

Bo Zhao, Haining Yang, Jing Qin, Jianting Ning and Jixin Ma

***Abstract*—The ubiquitous sensing capabilities of the Internet of Things (IoT) enable large-scale user tracking by identifying users who appear in at least t distributed location datasets. However, the distribution of these datasets across multiple tracking entities significantly increases the risk of sensitive data exposure. To address this problem, threshold multi-party private set intersection (T-MPSI) provides a promising privacy-preserving solution. Although the known works about T-MPSI have made valuable contributions, especially in terms of security, the efficiency deficiency in current T-MPSI protocols becomes apparent in large-scale deployment for user tracking. The core challenge is to develop an efficient T-MPSI protocol under the relaxed security constraint that is acceptable for user tracking. We first design a lightweight batch replicated secret sharing private membership test protocol with high performance. Moreover, we develop a one-round secure aggregation algorithm that bridges the gap between the secure query and the secure comparison built upon replicated secret sharing. Building on these techniques, we present an efficient T-MPSI protocol tailored to the designated k-collusion model. Our protocol significantly enhances secure query efficiency and ensures that the communication complexity of secure comparison remains independent of the number of parties. We formally prove its security, and extensive experiments in a LAN setting demonstrate at least a $6\times$ speedup for secure query and a $3\times$ speedup for secure comparison over the state-of-the-art protocol. These results confirm the practicality and efficiency of the proposed protocol for privacy-preserving user tracking.**

***Index Terms*—Data security, secure computation, threshold multi-party private set intersection, user tracking, Internet of Things.**

I. Introduction

THE Internet of Things (IoT) [1], [2] has significantly enhanced user tracking capabilities, as ubiquitous sensing and communication infrastructures enable the continuous collection of user spatiotemporal information [3]–[5]. A representative application is cross-organizational contact tracing in public-health emergencies, where public authorities, telecom operators, and transportation infrastructures collaboratively identify potentially exposed individuals under regulatory supervision. In such settings, an authorized authority maintains a predefined tracking list of target identifiers (e.g., identifiers associated with diagnosed users), while multiple distributed tracking entities each hold spatiotemporal data collected from user devices. Such data typically consist of pseudonymous identifiers observed over time and location, including proximity logs from Bluetooth-based contact tracing (e.g., Rolling Proximity Identifiers exchanged between nearby devices) as well as device identifiers or network attachment records [6]. Due to the incompleteness and noise in cross-entity data, relying on a single entity may lead to unreliable conclusions. To enhance reliability, a threshold mechanism is employed. In more detail, a target is considered to be valid only if it appears in at least t independent entities, where t denotes the minimum number of entities required to establish reliable evidence of exposure.

A traditional user tracking solution, as illustrated in Fig. 1(a), requires each tracking entity to transmit its local dataset to the authorized authority, which then performs centralized processing [7], [8]. However, tracking entities are often reluctant to disclose such datasets, as the contained spatiotemporal information may reveal sensitive user information, such as activity patterns and daily routines [9], [10]. Therefore, protecting user privacy is a primary concern in IoT-enabled tracking systems.

In order to address the privacy concerns, a solution should work even without transmitting the raw datasets to the authorized authority. Following this idea, multi-party private set intersection (MPSI) [11]–[13] seems to be feasible, but it is not the case. MPSI allows a leader party to learn the intersection of multiple private datasets without revealing any additional information about the datasets. If MPSI is used to implement user tracking, the authorized authority can learn only the users appearing across all the datasets, whereas no information about other users on the tracking list is revealed. Actually, the solution derived from MPSI is applicable to user tracking when the threshold t equals the total number of tracking entities. Hence, this solution fails to consider the case that the threshold t is less than the total number of tracking entities, because it overlooks the users who appear in only a subset of the tracking entities.

Threshold MPSI (T-MPSI) [14], [15] extends MPSI by enabling the leader party to learn the elements in its dataset appearing in at least t other datasets. The standard security requirement for T-MPSI protocols is resistance against arbitrary k-collusion, where any subset of up to k parties may collude to infer private information. However, this assumption may be overly strong in practical IoT-based user tracking scenarios.

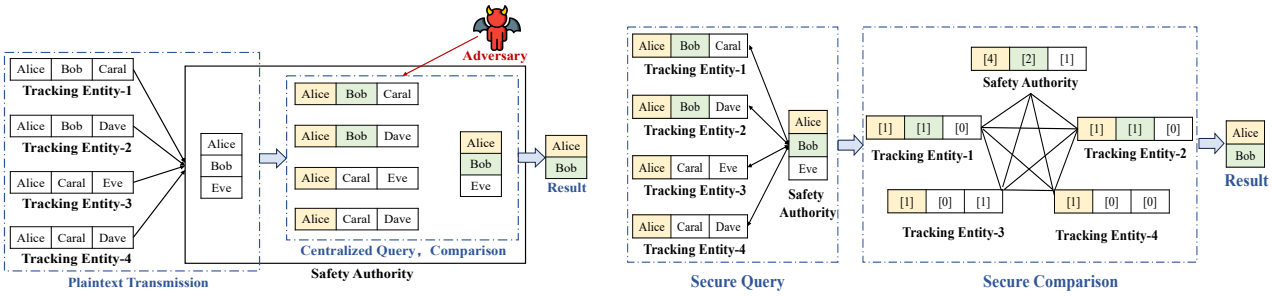
This work is supported by the National Natural Science Foundation of China (No.62402290, No.U24A20244) and Shandong Province Higher Education Institutions Youth Innovation and Technology Support Program (No.2024KJN051). (Corresponding author: Haining Yang.)

Bo Zhao and Haining Yang are with the School of Mathematics, Shandong University, Jinan, Shandong 250100, China, and also with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China (e-mail: zhaobomath@163.com; hainingyang@sdu.edu.cn).

Jing Qin is with the School of Mathematics, Shandong University, Jinan, Shandong 250100, China (e-mail: qinjing@sdu.edu.cn).

Jianting Ning is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Faculty of Data Science, City University of Macau, Macau, China (e-mail: jtning88@gmail.com).

Jixin Ma is with the School of Computing and Mathematical Sciences, University of Greenwich, SE10 9LS London, U.K. (e-mail: j.ma@greenwich.ac.uk).



(a) Traditional Solution (b) T-MPSI

Fig. 1. Comparison between traditional user tracking and threshold multi-party private set intersection (T-MPSI) for distributed user tracking with a threshold parameter $t \geq 2$. A safety authority maintains a tracking list of target identifiers, while multiple tracking entities independently collect user identifiers from distributed spatiotemporal records. An identifier is considered valid only if it appears in at least t tracking entities. (a) Traditional solution: each tracking entity directly transmits its plaintext dataset to the safety authority, which performs centralized query and comparison to identify valid targets, thereby exposing sensitive user information to the authority. (b) T-MPSI solution: the safety authority securely queries distributed datasets and performs privacy-preserving comparison using secure matching indicators (e.g., the boolean matching indicators shown in the figure), such that only identifiers satisfying the threshold condition are revealed, while the raw datasets of the tracking entities remain private.

In such settings, trust relationships among entities are often asymmetric due to differences in administrative domains and regulatory constraints. For example, government agencies, telecom operators, and infrastructure providers are typically subject to strict legal oversight, which limits their ability to arbitrarily collude, whereas less-regulated entities may pose higher collusion risks. This asymmetry naturally leads to non-uniform collusion risks, under which collusion is more likely to occur within specific subsets of entities that share common administrative control, rather than arbitrarily across all participants. This motivates considering resistance against designated k-collusion in IoT-based user tracking scenarios.

The security against arbitrary k-collusion has been the standard requirement for T-MPSI protocols. Clearly, these protocols can serve as a solution for the user tracking under designated k-collusion setting, where the efficiency of user tracking is dominated by the underlying T-MPSI protocol. From a functional perspective, T-MPSI consists of two main modules: secure query and secure comparison. The secure query module computes, for each element, the number of datasets in which it appears, while the secure comparison module identifies elements whose occurrence counts exceed a predefined threshold. However, existing T-MPSI protocols suffer from significant inefficiency. First, the secure query module typically relies on heavy cryptographic primitives, such as additive or fully homomorphic encryption [16], [17] and oblivious programmable pseudorandom functions [18]. Second, the secure comparison module incurs high communication complexity. For example, the protocol in [14] is based on additive homomorphic encryption and has quadratic communication complexity in the number of participants, while the protocols in [15], [19] are based on Shamir secret sharing and have linear communication complexity. The main reason for this inefficiency is that existing protocols are designed to achieve security against arbitrary k-collusion. Accordingly, we can design a more efficient T-MPSI protocol by relaxing the security requirement to security against the designated k-collusion.

This work intends to propose an efficient T-MPSI protocol secure against designated k-collusion. Towards this goal, we first put forward an aided batch replicated secret sharing private membership test (ab-rssPMT). The proposed ab-rssPMT protocol is derived from the light-weight pseudorandom function, such that its performance outperforms the known private membership test protocols. In order to improve the efficiency in secure comparison module, it seems possible to employ a more efficient secure comparison built upon replicated secret sharing [20], [21], whose efficiency is independent of the number of participants. However, it is non-trivial to employ replicated secret sharing based secure comparison in T-MPSI, because the output of the secure query is distributed among all parties, whereas replicated secret sharing based secure comparison requires the input to be privately held by only three parties. To bridge the gap, we further propose a secure aggregation algorithm. This algorithm is optimal in the sense that it requires only one round of communication, introducing a slight overhead to the proposed T-MPSI protocol.

A. Our Contribution

To improve the efficiency of privacy-preserving user tracking, we investigate the design of an efficient T-MPSI protocol under the designated k-collusion setting. Our approach improves both the secure query and secure comparison modules through lightweight cryptographic techniques and efficient secure aggregation. Based on these techniques, we construct an efficient T-MPSI protocol with significantly reduced communication and computation overhead. The main contributions of this work are summarized as follows.

- We propose a private membership test protocol called aided batch replicated secret sharing private membership test. This protocol relies solely on lightweight pseudorandom functions in the membership test phase, thereby achieving better performance than existing private membership test protocols.
- We propose an efficient secure aggregation algorithm. This algorithm is optimal in the sense that it requires only one round of communication, introducing a slight overhead to the proposed T-MPSI protocol.

- We present an efficient T-MPSI protocol that is secure under the designated k-collusion. The proposed protocol enhances secure query efficiency and ensures that the communication complexity of secure comparison remains independent of the number of parties.
- We implemented our T-MPSI protocol in C++ and evaluated it in both LAN and WAN settings. For instance, with nine parties and a dataset size of 5×10^6, the speedups in a LAN setting are $6\times$ for secure query and $38\times$ for secure comparison over the state-of-the-art, while in a WAN setting, the corresponding speedups are $1.5\times$ and $51\times$, respectively. These results confirm the effectiveness of our T-MPSI protocol across different network environments.

B. Related Work

MPSI allows the leader party to learn the intersection of multiple private datasets (including its own) without revealing any additional information about the datasets. Freedman et al. [24] first introduced the concept of MPSI and proposed a protocol based on secret sharing. Their key idea is that a secret can be reconstructed if and only if the corresponding element belongs to the intersection. Following this idea, subsequent works [13], [14] adopted zero-summation secret sharing and [11], [12] adopted zeroXOR secret sharing to achieve the same functionality with higher efficiency. The shared characteristics of these protocol is that they can only identify the elements appearing in every dataset. This property may limit the applications of MPSI, where the elements present in a subset of the datasets is needed in some cases.

T-MPSI is an advanced variant of MPSI, because it strengthens MPSI with the t-and-over mechanism. To be more specific, T-MPSI enables participants to learn the elements that appear in at least t datasets. Existing T-MPSI functionalities can be classified into three categories based on their output semantics: *global-output*, *local-output*, and *single-output*. The global-output T-MPSI [25], [26] reveals all elements that appear in at least t datasets to every participant. In contrast, local-output T-MPSI allows each participant to learn only the over-threshold elements contained in its own dataset. The single-output T-MPSI restricts the output to a single designated party (i.e., the leader), who learns only the over-threshold elements in its dataset. A straightforward approach to realizing local-output or single-output T-MPSI is to first execute a global-output T-MPSI protocol and then locally filter out elements that are not contained in the target dataset. However, this approach is insecure, as it may reveal elements that do not belong to the participant's dataset. In this paper, we focus on T-MPSI protocols with local-output and single-output semantics.

Mahdavi et al. [22] proposed the first local-output T-MPSI protocol by leveraging the t-out-of-n reconstruction property of Shamir secret sharing. However, the reconstruction phase incurs a high computational cost of $O(m(n\log(m/t))^{2t})$, where m denotes the dataset size and n denotes the number of participants. To mitigate this overhead, Arpaci et al. [23] introduced a hashing-based binning technique with pseudorandom ordering, reducing the complexity to $O(t^2 m\binom{n}{t})$. Notably, their protocol operates in a non-interactive model, where each participant sends a single message to an aggregator, providing advantages in terms of communication rounds and deployment simplicity. Local-output T-MPSI can be directly adapted to the single-output setting by simply designating the output party. Nevertheless, such protocols typically incur high computational overhead. For example, in the protocol of Arpaci et al. [23], when the threshold equals the total number of participants, the computational complexity degrades to $O(n^2 m)$.

Bay et al. [14] proposed a single-output T-MPSI protocol based on Bloom filters and additive homomorphic encryption (AHE). For n participants, the designated party exchanges n^2 ciphertexts with the other parties to perform secure comparison, resulting in quadratic communication complexity. Chandran et al. [15] presented a T-MPSI protocol based on symmetric-key techniques, including oblivious programmable pseudorandom functions (OPPRF) and Shamir secret sharing, achieving linear communication complexity. Yang et al. [19] designed a protocol tailored for unbalanced scenarios where the designated party holds a relatively small dataset. Their construction leverages the batching capability of fully homomorphic encryption (FHE) to evaluate multiple elements within a single ciphertext, thereby reducing communication overhead.

Current single-output T-MPSI protocols are designed to be secure against arbitrary k-collusion, which has been the standard security requirement. However, such strong security guarantees inherently lead to high overhead in secure query and communication complexity in secure comparison. In particular, under the standard multi-party secure computation model, it is known that the communication complexity must scale at least linearly with the number of parties n, and often $\Omega(n \cdot g)$ for circuits of size g [27], [28]. These results suggest that the communication costs of existing protocols are largely unavoidable under standard security assumptions.

In this work, we take a different approach by relaxing the security requirement to a designated k-collusion model, where only a specific subset of parties may collude. Such a relaxation is well motivated in many practical scenarios where trust assumptions are asymmetric. By weakening the adversarial model, our protocol operates outside the standard setting in which these lower bounds apply. This allows us to significantly reduce the communication complexity compared to existing protocols that achieve arbitrary k-collusion security. To further highlight the differences between our protocol and existing approaches, we provide a detailed comparison in Table I.

C. Paper Outline

This paper is organized as follows: Section II reviews the cryptographic knowledge and notations. Section III presents our ab-rssPMT protocol, including its formal definition, construction, and security proof. Section IV introduces our T-MPSI protocol and analyzes its security. Section V evaluates the complexity and performance of our T-MPSI protocol. Finally, Section VI concludes the paper.

TABLE I
COMPARISON OF EXISTING WORKS WITH OUR SCHEME.

Protocol	Collusion	Secure Query				Secure Comparison			
		Prim.	Comm.	Comp.	Rnd.	Prim.	Comm.	Comp.	Rnd.
[22]	$(n-1)^*$	OPR-SS	$O(mnt)$	$O(mnt)$	2	SSS	$O(mn)$	$O\big(m(n\log(m/t))^{2t}\big)$	1
[23]	$(n-1)^*$	OPR-SS	$O(mnt)$	$O(mnt)$	3	SSS	$O(mnt)$	$O\big(m\binom{n}{t}t^2\big)$	1
	$(n-1)^\dagger$	HMAC	0	$O(mnt)$	0	SSS	$O(mnt)$	$O\big(m\binom{n}{t}t^2\big)$	1
[14]	$n-1$	AHE	$O(mn)$	$O(mn^2)$	1	AHE	$O(mn^2)$	$O(mn^2)$	$O(n)$
[15]	$n/2$	OPPRF	$O(mn)$	$O(mn)$	$O(1)$	SSS	$O(mn)$	$O(mn)$	$O(1)$
[19]	$n/2$	FHE	$O(mn)$	$O(mn)$	$O(1)$	SSS	$O(mn)$	$O(mn)$	$O(1)$
Ours	$(n-3)^*$	PRF	$O(mn)$	$O(mn)$	$O(1)$	RSS	$O(m)$	$O(m)$	$O(1)$

Notation. Prim.: Primitive; Comm.: Communication; Comp.: Computation; Rnd.: Round. m denotes the dataset size; n the number of parties; and t the threshold. OPR-SS: Oblivious PRF with Secret Sharing; HMAC: Hash-based Message Authentication Code; PRF: Pseudorandom Function; OPPRF: Oblivious Programmable PRF; SSS: Shamir Secret Sharing; RSS: Replicated Secret Sharing; AHE: Additive Homomorphic Encryption; FHE: Fully Homomorphic Encryption. Collusion refers to the maximum number of colluding parties. $*$ denotes a designated subset of non-colluding parties. † indicates that the protocol involves an external non-colluding server.

II. PRELIMINARIES

This section introduces the main cryptographic knowledge used to construct our T-MPSI. Table II, we illustrate the main notations used in this paper.

TABLE II
DESCRIPTION OF NOTATIONS

Notation	Description
n	the number of parties
t	Minimum number of occurrences of an element
k	Maximum number of colluding parties allowed
λ	Computational security parameter
κ	Statistical security parameter
$[m]$	The set $\{1, 2, \ldots, m\}$
$[m_1, m_2]$	The set $\{m_1, m_1+1, \ldots, m_2\}$
$\lvert\mathcal{Q}\rvert$	The size of a set $\mathcal{Q}$
$\mathbb{Z}_{2^l}$	The ring of integers modulo 2^l
$\vec{x}$	A vector $(x_1, x_2, \ldots, x_m)$ for some $m \in \mathbb{N}$
$q_1 \| q_2$	Concatenation of two strings q_1 and q_2

A. Threshold Multi-Party Private Set Intersection

Problem Formulation. Let Q denote the leader, and $D_1, \ldots, D_{n-1}$ denote the data holders, where $n \geq 3$. The leader holds a query set $\mathcal{Q}$ of size m_1, and each data holder D_i holds a dataset $\mathcal{D}_i$ of size m_2. The query threshold is denoted by $t \in [1, n-1]$. The threshold multi-party private set intersection (T-MPSI) protocol enables the leader to learn which elements in $\mathcal{Q}$ appear in at least t of the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_{n-1}$, without revealing any additional information about the query set $\mathcal{Q}$ or the datasets $\{\mathcal{D}_i\}_{i=1}^{n-1}$. Formally, the leader learns the set

$$Y = \Big\{ q \in \mathcal{Q} : \big|\{i \in [n-1] : q \in \mathcal{D}_i\}\big| \geq t \Big\}.$$

Note that this functionality differs from threshold PSI defined over set intersection [29], as the threshold in our setting is defined over the number of parties in which an element appears, rather than over the size of the intersection.

Application. T-MPSI is motivated by privacy-preserving user tracking, where a leader identifies users appearing in at least t location datasets without revealing other users' location information. Beyond this, it also applies to collaborative network intrusion detection [23] and anonymous electronic voting [30].

We define the ideal functionality as $\mathcal{F}_{\text{T-MPSI}}^{n,m_1,m_2,t}$, see II.1 for details.

Functionality II.1. $\mathcal{F}_{\text{T-MPSI}}^{n,m_1,m_2,t}$
Parameters: There are n parties: a leader Q and $n-1$ data holders $D_1, \ldots, D_{n-1}$. The leader holds a query set $\mathcal{Q}$ of size m_1, each data holder holds a dataset $\mathcal{D}_i$ of size m_2, and a threshold $t \in [1, n-1]$ is given.
Behavior:

- Wait to receive the query set $\mathcal{Q}$ and the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_{n-1}$.
- For each element $q \in \mathcal{Q}$, compute $\eta_q = \big|\{i \in [n-1] : q \in \mathcal{D}_i\}\big|$.
- Output $Y = \{q \in \mathcal{Q} : \eta_q \geq t\}$ to the leader.

B. Adversarial Model and Security Definition

Semi-Honest Model. Let $\mathcal{P} = \{Q, D_1, \ldots, D_{n-1}\}$ denote the set of participating parties. We consider a semi-honest adversary that corrupts a subset of parties $\mathcal{C} \subseteq \mathcal{P}$. The corrupted parties follow the protocol specification honestly but may attempt to infer additional information from their internal states and the messages they receive during the protocol execution.

Designated k-Collusion Model. We adopt a designated corruption structure tailored to practical deployment scenarios. Specifically, we designate a subset of parties $\{Q, D_1, D_2\}$ that are assumed to be non-colluding with each other and with the remaining parties. The adversary is allowed to corrupt any subset of parties $\mathcal{C} \subseteq \mathcal{P}$ such that

$$|\mathcal{C} \cap \{Q, D_1, D_2\}| \leq 1,$$

that is, at most one party in $\{Q, D_1, D_2\}$ can be corrupted. This model captures the setting where certain entities (e.g., regulator or infrastructure providers) are assumed to be non-colluding due to organizational or legal constraints, while the remaining parties may arbitrarily collude.

Security Definition. We adopt the standard simulation-based security framework. Let π be a protocol that realizes the functionality $\mathcal{F}_{\text{T-MPSI}}^{n,m_1,m_2,t}$. For any probabilistic polynomial-time adversary $\mathcal{A}$ corrupting a set $\mathcal{C} \subseteq \mathcal{P}$, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that

$$\text{View}_{\mathcal{C}}^{\pi}(\mathcal{Q}, \mathcal{D}_1, \ldots, \mathcal{D}_{n-1}) \stackrel{c}{\equiv} \mathcal{S}(\{\mathcal{Q}\}_{Q\in\mathcal{C}}, \{\mathcal{D}_i\}_{D_i\in\mathcal{C}}, Y),$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability between two distributions. This implies that the protocol leaks no information beyond the inputs of the corrupted parties and the output of the functionality.

C. Cuckoo Hashing and Simple Hashing

Cuckoo hashing [31] maps the m elements of a dataset $\mathcal{Q}$ into β buckets, where each bucket can hold at most one element. The cuckoo hashing procedure works as follows: it initializes empty table T_1 with β buckets and chooses h independent hash functions $H_1, \ldots, H_h : \{0,1\}^* \to [\beta]$, where $\beta = \varepsilon m$ and $\varepsilon > 1$ is an expansion factor. For each element $q \in \mathcal{Q}$, if there exists an index $i \in [1, h]$ such that the bucket $H_i(q)$ is empty, the element is inserted into the bucket with the smallest such index. Otherwise, an index $i \in [1, h]$ is chosen uniformly at random, the element currently in bucket $H_i(q)$ is evicted, and q is inserted into $H_i(q)$. This process is repeated until either all elements are inserted without further evictions, or the number of evictions exceeds the eviction threshold, in which case the procedure is treated as a failure. To guarantee that the procedure succeed with overwhelming probability, we follow the parameter configuration proposed in [32]; see Section IV-C for details.

Simple hashing maps the m elements of a dataset $\mathcal{Q}$ into β buckets, where each bucket can hold multiple elements. The simple hashing procedure works as follows: it initializes empty table T_1 with β buckets and chooses h independent hash functions $H_1, \ldots, H_h : \{0,1\}^* \to [\beta]$. For each element $q \in \mathcal{Q}$, the element is inserted into the buckets $H_1(q), \ldots, H_h(q)$.

D. Three-Party Secure Computation Functionality

Our construction employs several functionalities based on replicated secret sharing, which are secure in the three-party non-colluding setting. These functionalities can be instantiated using the protocols proposed in [20], [21]. The employed functionalities are briefly described below.

Replicated secret sharing shares a secret value $x \in \mathbb{Z}_{2^l}$ among three parties. Specifically, the secret is split into three random values $x_1, x_2, x_3 \in \mathbb{Z}_{2^l}$ such that $x = x_1 + x_2 + x_3$. The shares are distributed to the three parties as overlapping pairs: (x_1, x_2), (x_2, x_3), and (x_1, x_3). We use the notation $\langle x \rangle$ to denote the replicated secret sharing of x, i.e., $\langle x \rangle = \{(x_1, x_2), (x_2, x_3), (x_1, x_3)\}$. Similarly, for a vector $\vec{x} = \{x_1, \ldots, x_m\} \in \mathbb{Z}_{2^l}^m$, we write $\langle \vec{x} \rangle$ to denote the element-wise replicated secret sharing of $\vec{x}$. When $l = 1$, the secret is defined over the binary field $\mathbb{Z}_2$, where the operations $\oplus$ and $\wedge$ correspond to addition and multiplication, respectively. To distinguish this case from the general setting where $l > 1$, We use $\langle x \rangle_B$ and $\langle \vec{x} \rangle_B$ to denote a replicated secret sharing and vectorized replicated secret sharing in the binary field, respectively. We also refer to them as boolean replicated secret sharing and vectorized boolean replicated secret sharing.

The functionalities $\mathcal{F}_{\text{Mul}}^m, \mathcal{F}_{\text{Add}}^m, \mathcal{F}_{\text{EQZ}}^m, \mathcal{F}_{\text{GTZ}}^m, \mathcal{F}_{\text{B2A}}^m, \mathcal{F}_{\text{Reveal}}^m$ are described as follows:

- $\mathsf{Add}^m(\langle \vec{x} \rangle, \langle \vec{y} \rangle)$: Input $\langle \vec{x} \rangle, \langle \vec{y} \rangle$ and output $\langle \vec{x} + \vec{y} \rangle$, where $\vec{x}, \vec{y} \in \mathbb{Z}_{2^l}^m$ and $+$ denotes element-wise addition.
- $\mathsf{Add}^m(\langle \vec{x} \rangle, y)$: Input $\langle \vec{x} \rangle, y$ and output $\langle \vec{x} + \vec{y} \rangle$, where $\vec{y}$ is a vector with each entry equal to y.
- $\mathsf{EQZ}^m(\langle \vec{x} \rangle)$: Input $\langle \vec{x} \rangle$ and output $\langle \vec{y} \rangle_B$, where $\vec{x} \in \mathbb{Z}_{2^l}^m, \vec{y} \in \mathbb{Z}_2^m$ and $y_i = 1$ if $x_i = 0$, otherwise $y_i = 0$.
- $\mathsf{GTZ}^m(\langle \vec{x} \rangle)$: Input $\langle \vec{x} \rangle$ and output $\langle \vec{y} \rangle_B$, where $\vec{x} \in \mathbb{Z}_{2^l}^m, \vec{y} \in \mathbb{Z}_2^m$ and $y_i = 1$ if $x_i \geq 0$, otherwise $y_i = 0$.
- $\mathsf{B2A}^m(\langle \vec{x} \rangle_B)$: Input $\langle \vec{x} \rangle_B$ and output $\langle \vec{x} \rangle$.
- $\mathsf{Reveal}^m(\langle \vec{x} \rangle)$: Input $\langle \vec{x} \rangle$ and output $\vec{x}$.

TABLE III
COMMUNICATION COMPLEXITY

Functionality	Computation	Communication	Round
Add^m	$O(m)$	0	0
EQZ^m	$O(ml)$	$O(ml)$	$O(\log l)$
GTZ^m	$O(ml)$	$O(ml)$	$O(\log l)$
$\mathsf{B2A}^m$	$O(m)$	$O(ml)$	$O(1)$
Reveal^m	$O(m)$	$O(ml)$	1

E. Oblivious Key-Value Store

A key-value store (KVS) is a data structure that encodes a set of key-value pairs. KVS consists of two algorithms.

- Encode: Input a key-value pairs set $\{(k_i, v_i)\}_{i=1}^m$ from $\mathcal{K} \times \mathcal{V}$, and output a structure T (or, with negligible probability, an error indicator $\perp$).
- Decode: Input a structure T, a key k and output a value v.

Correctness. A KVS is said to be correct if, for any $M \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys, it holds that for every $(k, v) \in M$:

$$\Pr[\mathsf{Decode}(\mathsf{Encode}(M), k) = v] \ \textit{is overwhelming}.$$

Obliviousness. An oblivious KVS (OKVS) [33] can hide the set of keys used in key-value pairs. Consider Experiment II.2.

Experiment II.2. $\text{Exp}_{\text{OKVS}}^{\mathcal{A}}(k_1, k_2, \ldots, k_m)$
- choose uniform $v_i \leftarrow \mathcal{V}$, for $i \in [m]$.
- return $\mathcal{A}(\mathsf{Encode}(\{(k_1, v_1), \ldots, (k_m, v_m)\}))$.

The KVS is oblivious if for all distinct key $k_1^1, \ldots, k_m^1$ and $k_1^2, \ldots, k_m^2$ and all probabilistic polynomial time adversaries $\mathcal{A}$:

$$\big|\Pr[\text{Exp}_{\text{OKVS}}^{\mathcal{A}}(k_1^1, k_2^1, \ldots, k_m^1)] - \Pr[\text{Exp}_{\text{OKVS}}^{\mathcal{A}}(k_1^2, k_2^2, \ldots, k_m^2)]\big| \leq \text{negl}(\kappa),$$

where negl is a negligible function.

In this paper, we adopt the construction in [34], which achieves linear-time encoding and decoding complexity with a near-optimal encoding rate. The encoding rate, defined as the ratio of the size of the encoded structure to that of the

input, measures the space efficiency of the construction. In our setting, the size of the encoded structure is approximately $1.28\times$ that of the input.

III. Aided Batch Replicated Secret Sharing Private Membership Test

A. Definitions

Aided batch replicated secret sharing private membership test (ab-rssPMT) is a three-party functionality involving a receiver, a sender, and an aider. The receiver holds a query vector $\vec{Q} = (q_1, \ldots, q_\beta)$. The sender holds a vector of sets $\vec{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_\beta)$, where each $\mathcal{S}_i$ is a set, and the total number of elements satisfies $\sum_{i=1}^{\beta} |\mathcal{S}_i| = N$. The ab-rssPMT functionality enables the parties to obtain a replicated secret sharing of the result vector $\langle \vec{x} \rangle_B$, where $\vec{x} = (x_1 \ldots, x_\beta) \in \mathbb{Z}_2^\beta$ and

$$x_i = \begin{cases} 1 & \text{if } q_i \in \mathcal{S}_i, \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in [1, \beta].$$

This functionality ensures that the parties obtain no additional information apart from what is revealed by their respective inputs and outputs. In more detail, each party obtains only secret shares of the results, without learning the results in plaintext.

We consider a semi-honest adversary, which follows the protocol specification faithfully but attempts to infer additional information from the execution transcript. In addition, we assume that the adversary may corrupt at most one of the parties, meaning that all parties are non-colluding. The ideal functionality of ab-rssPMT is formally defined in Functionality III.1.

Functionality III.1. $\mathcal{F}_{\text{ab-rssPMT}}^{\beta,N}$
Parameters: There are three parties: receiver, sender, and aider. The receiver holds a query vector $\vec{Q} = (q_1, \ldots, q_\beta)$. The sender holds a vector of sets $\vec{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_\beta)$, where each $\mathcal{S}_i$ is a set and total number of elements satisfies $\sum_{i=1}^{\beta} |\mathcal{S}_i| = N$. Aider holds nothing.
Behavior:

- Wait to receive the query vector $\vec{Q}$ from receiver and the set vector $\vec{S}$ from the sender.
- For each $i \in [\beta]$, set $x_i = 1$ if $q_i \in \mathcal{S}_i$, and $x_i = 0$ otherwise.
- Output $\langle \vec{x} \rangle_B$, where $\vec{x} = (x_1, \ldots, x_\beta) \in \mathbb{Z}_2^\beta$.

B. Construction of ab-rssPMT Protocol

To achieve an efficient private membership test, we introduce a non-colluding third party, referred to as the aider. The introduction of such a party is justified under the designated k-collusion model of T-MPSI, since the parties Q, D_1, and D_2 are assumed not to collude with any of the other parties. With the assistance of the aider, our private membership test can be implemented using only pseudorandom function. Additionally, to ensure that the membership test results can be securely used in T-MPSI, we require the protocol that does not reveal the plaintext membership test results to the parties. In more detail, the membership test results must be obtained in encrypted form and subsequently converted into boolean replicated secret sharing. Accordingly, our ab-rssPMT protocol consists of two phases: Private Membership Test and Replicated Secret Sharing.

Private Membership Test. In this phase, our goal is to check whether $q_i \in \mathcal{S}_i$ without revealing the membership result in plaintext. To achieve this, we require that the sender and the receiver obtain the same secret value if and only if the query element q_i belongs to the set $\mathcal{S}_i$. Specifically, the sender samples a random value w_i for the set $\mathcal{S}_i$, and masks every element $s_{ij} \in \mathcal{S}_i$ using the same w_i. If the query element $q_i \in \mathcal{S}_i$, the receiver can successfully unmask and recover w_i; otherwise, it obtains a pseudorandom value. However, this design suffers from two issues. **Security**: when the query domain is small, the receiver can brute-force all possible queries to learn w_i. **Lookup**: When performing batch tests, the sender needs to mix all sets $\vec{\mathcal{S}}$ into a single set to prevent statistical attacks, since sending each $\mathcal{S}_i$ individually would allow the receiver to infer the element distribution from their sizes. But this mixing causes the receiver to lose the correspondence between queries and sets, making it unable to determine the correct query to unmask, even if $q_i \in \mathcal{S}_i$.

- To address the security issue, we employ a double-PRF masking and an aider-helped computation, which limits the receiver can perform only one unmasking attempt. Concretely, the sender samples two PRF keys $k_1, k_2 \in \{0,1\}^\lambda$, sending k_1 to the receiver and k_2 to the aider. Then, for each $s_{ij} \in \mathcal{S}_i$, the sender computes

  $$F(k_2, F(k_1, s_{ij} \| i)) \oplus w_i.$$

  The receiver locally computes $F(k_1, q_i \| i)$ and sends it to the aider, who helps computes $F(k_2, F(k_1, q_i \| i))$. The receiver can only use this output $F(k_2, F(k_1, q_i \| i))$ to attempt unmasking.
- To resolve the lookup issue, we employ OKVS for batch encoding and decoding of elements. This approach conceals the mapping between the query element and the queried set, yet enables efficient retrieval of the masked element when the queried item is contained in the set. Specifically, the sender encodes all key-value pairs $\{s_{ij} \| i, F(k_2, F(k_1, s_{ij} \| i)) \oplus w_i\}$ into an OKVS structure T. Upon receiving T, the receiver uses the key $q_i \| i$ to decode. If $q_i \in \mathcal{S}_i$, the receiver can recover w_i from the decoded value; otherwise, it obtains a pseudorandom value. That is, the receiver computes

  $$v_i = F(k_2, F(k_1, q_i \| i)) \oplus \mathsf{Decode}(T, q_i \| i).$$

This phase obtains membership test results in encrypted form, relying solely on double-PRF. Next, we convert the encrypted test results into the form of replicated secret sharing.

Replicated Secret Sharing. In this phase, our goal is to convert the values w_i and v_i into boolean replicated secret sharing form. First, the sender and receiver convert their respective values into replicated shares. Specifically, The receiver samples a random value $y_{1i} \in \mathbb{Z}_{2^\sigma}$ and masks v_i as

Protocol III.1. $\pi^{\beta,N}_{\text{ab-rssPMT}}$
Parameters:

- Computational security parameter λ.
- There are three parties: receiver, sender, and aider. Receiver has a query vector $\vec{Q} = (q_1, \ldots, q_\beta)$. Sender has a vector of sets $\vec{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_\beta)$, where each set $\mathcal{S}_i = \{s_{ij}\}_{j\in[|\mathcal{S}_i|]}$, with $\sum_{i=1}^{\beta} |\mathcal{S}_i| = N$.
- A PRF $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\sigma$.
- Functionality $\mathcal{F}^{\beta}_{\text{EQZ}}$ executed over ring $\mathbb{Z}_{2^\sigma}$.
- An OKVS scheme (Encode, Decode).

Protocol:

1) Private Membership Test:
   - Sender uniformly samples k_1 and k_2 from $\{0,1\}^\lambda$, then sends k_1 to receiver and k_2 to aider.
   - Receiver computes the vector $\mathcal{Q}' = \{q'_1, \ldots, q'_\beta\}$, where $q'_i = F(k_1, q_i\|i)$, and sends $\mathcal{Q}'$ to aider.
   - Aider computes the vector $\mathcal{Q}'' = \{q''_1, \ldots, q''_\beta\}$, where $q''_i = F(k_2, q'_i)$, and sends $\mathcal{Q}''$ to receiver.
   - Sender randomly samples independent values $w_1, w_2, \ldots, w_\beta \in \{0,1\}^\sigma$ for β sets. Then, sender constructs an OKVS $T = \mathsf{Encode}(\{s_{ij}\|i, F(k_2, F(k_1, s_{ij}\|i)) \oplus w_i\}_{i\in[\beta], s_{ij}\in\mathcal{S}_i})$, then sends T to receiver.
   - For each $i \in [\beta]$, receiver computes $v_i = q''_i \oplus \mathsf{Decode}(T, q_i\|i)$.
2) Replicated Secret Sharing:
   - Receiver randomly samples vector $\vec{y}_1 = (y_{11}, \ldots, y_{1\beta}) \in \mathbb{Z}^{\beta}_{2^\sigma}$ and sets $\vec{y}'_2 = ((v_1 \bmod 2^\sigma) + y_{11}, \ldots, (v_\beta \bmod 2^\sigma) + y_{1\beta})$, then sends $\vec{y}_1$ to aider and $\vec{y}'_2$ to sender.
   - Sender randomly samples vector $\vec{y}_3 = (y_{31}, \ldots, y_{3\beta}) \in \mathbb{Z}^{\beta}_{2^\sigma}$ and sets $\vec{y}_2 = ((w_1 \bmod 2^\sigma) - y_{3i} - y'_{2i}, \ldots, (w_\beta \bmod 2^\sigma) - y_{3\beta} - y'_{2\beta})$. The sender then sends $\vec{y}_3$ to aider and $\vec{y}_2$ to receiver. As a result, receiver, sender and aider obtain the replicated secret sharing $\langle \vec{y} \rangle = \{((y_{11}, y_{21}), (y_{21}, y_{31}), (y_{11}, y_{31})), \ldots, ((y_{1\beta}, y_{2\beta}), (y_{2\beta}, y_{3\beta}), (y_{1\beta}, y_{3\beta}))\}$.
   - Finally, receiver, sender and aider invoke the functionality $\mathcal{F}^{\beta}_{\text{EQZ}}$ to compute $\langle \vec{x} \rangle_B \leftarrow \text{EQZ}^{\beta}(\langle \vec{y} \rangle)$, and output $\langle \vec{x} \rangle_B$.

$v_i \bmod 2^\sigma + y_{1i}$. Similarly, the sender samples $y_{3i} \in \mathbb{Z}_{2^\sigma}$ and masks w_i as $w_i \bmod 2^\sigma - y_{3i}$. They then compute:

$$y_{2i} = w_i \bmod 2^\sigma - y_{3i} - (v_i \bmod 2^\sigma + y_{1i}),$$

If $w_i = v_i$, then $y_{1i} + y_{2i} + y_{3i} = 0$; otherwise, this sum is uniformly random in $\mathbb{Z}_{2^\sigma}$. The shares are assigned as: receiver holds (y_{1i}, y_{2i}), the sender holds (y_{2i}, y_{3i}), and the aider holds (y_{1i}, y_{3i}). Second, convert the replicated share into a boolean replicated secret share. The parties invoke the $\mathcal{F}^{\beta}_{\text{EQZ}}$ functionality to check whether $\{(y_{1i}, y_{2i}), (y_{2i}, y_{3i}), (y_{1i}, y_{3i})\}$ equals zero, thereby obtaining the membership test result in boolean replicated secret sharing form.

We provide the formal description in Protocol III.1.

C. Correctness and Security

Theorem 1. *Protocol $\pi^{\beta,N}_{\text{ab-rssPMT}}$ is correct.*

Proof. To show correctness, we consider the following cases.

$q_i \in \mathcal{S}_i$. Sender sample w_i and encodes the key-value pair $(q_i\|i, F(k_2, F(k_1, q_i\|i)) \oplus w_i)$ into the OKVS T and sends it to receiver. Receiver receives $q''_i = F(k_2, F(k_1, q_i\|i))$ from the aider and computes:

$$\begin{aligned} v_i &= q''_i \oplus \mathsf{Decode}(T, q_i\|i) \\ &= q''_i \oplus (F(k_2, F(k_1, q_i\|i)) \oplus w_i) \\ &= F(k_2, F(k_1, q_i\|i)) \oplus (F(k_2, F(k_1, q_i\|i)) \oplus w_i) \\ &= w_i. \end{aligned}$$

Receiver samples y_{1i} and computes $y'_{2i} = (v_i \bmod 2^\sigma) + y_{1i}$. Sender samples y_{3i} and computes $y_{2i} = ((w_i \bmod 2^\sigma) - y_{3i} - y'_{2i})$. Then, we have:

$$\begin{aligned} &y_{1i} + y_{2i} + y_{3i} \\ &= y_{1i} + ((w_i \bmod 2^\sigma) - y_{3i} - y'_{2i}) + y_{3i} \\ &= y_{1i} + ((w_i \bmod 2^\sigma) - y_{3i} - (v_i \bmod 2^\sigma) - y_{1i}) + y_{3i} \\ &= 0. \end{aligned}$$

Thus, receiver, sender and aider invoke the $\mathcal{F}^{\beta}_{\text{EQZ}}$ functionality, obtaining $\langle x_i \rangle_B = \langle 1 \rangle_B$.

$q_i \notin \mathcal{S}_i$. Sender does not encode q_i into the OKVS T. Thus, with overwhelming probability $1 - 2^{-\sigma}$, we have $w_i \neq v_i$, and therefore $y_{1i} + y_{2i} + y_{3i} \neq 0$. As a result, receiver, sender and aider invoke the $\mathcal{F}^{\beta}_{\text{EQZ}}$ functionality, obtaining $\langle x_i \rangle_B = \langle 0 \rangle_B$. □

Theorem 2. *Protocol $\pi^{\beta,N}_{\text{ab-rssPMT}}$ securely computes functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ in the $\mathcal{F}^{\beta}_{\text{EQZ}}$-hybrid model and in the presence of a semi-honest adversary that can corrupt at most one party.*

Proof. We simulate the view of corrupted parties. We consider the following cases.

Corrupted receiver. The simulator is given the query vector $\vec{Q}$ and receiver's output as input. In the private membership test phase, the receiver obtains a random key, a pseudorandom vector and an OKVS. To simulate this, the simulator samples two random keys $k_1, k_2 \in \{0,1\}^\lambda$ and a vector $\mathcal{Q}'' = \{q''_1, \ldots, q''_\beta\}$ uniformly at random. The simulator then sends k_1 and $\mathcal{Q}''$ to the receiver. The security of the PRF ensures that $\mathcal{Q}''$ is computationally indistinguishable from the vector generated

in the real execution. To simulate the OKVS, the simulator selects a random set of key-value pairs $\{(a_i, b_i)\}_{i\in[\beta]}$ and computes $T = \mathsf{Encode}(\{(a_i, b_i)\})$. The encoded OKVS T is then sent to the receiver. Due to the obliviousness property of the OKVS, the encoded T is computationally indistinguishable from an OKVS generated using $\mathcal{S}_i$ together with k_1 and k_2 in the real execution. In the replicated secret sharing phase, the receiver receives a vector. To simulate this, the simulator randomly selects $\vec{y}_2$ and sends it to the receiver. Since both $\vec{y}_3$ and $\{w_i\}_{i\in[\beta]}$ are random, $\vec{y}_2$ is indistinguishable from the vector received in the real execution. Finally, the simulator outputs the receiver's output.

Corrupted sender. The simulator is given vector $\vec{\mathcal{S}}$ and sender's output as input. In the private membership test phase, the sender receives nothing in the real execution. In the replicated secret sharing phase, the sender receives a vector. To simulate this, the simulator randomly selects $\vec{y}_2'$ and sends it to the server. Since both $\vec{y}_1$ and $\{v_i\}_{i\in[\beta]}$ are random, $\vec{y}_2'$ is indistinguishable from the vector received in the real execution. Finally, the simulator outputs the sender's output.

Corrupted aider. The simulator is given aider's output as input. In the private membership test phase, the aider receives a random key and a pseudorandom vector. To simulate this, the simulator selects a random key $k_2 \in \{0,1\}^\lambda$ and sends it to the aider. The simulator then randomly samples $\mathcal{Q}' = \{q_1', \ldots, q_\beta'\}$ and sends it to the aider. The security of the PRF ensures that $\mathcal{Q}'$ is computationally indistinguishable from the vector in the real execution. In the replicated secret sharing phase, the aider receives two random vectors. To simulate this, the simulator randomly selects $\vec{y}_1, \vec{y}_3$ and sends both vectors to the aider. Finally, the simulator outputs the aider's output. □

D. Parameter Choice

Choice of the parameter σ. In the correctness proof, for the case where $q_i \notin \mathcal{S}_i$, the probability that $w_i = v_i$ is at most $2^{-\sigma}$. To ensure that the failure probability of $\pi^{\beta,N}_{\text{ab-rssPMT}}$ is bounded by $2^{-\kappa}$, it is required that $\beta \cdot 2^{-\sigma} < 2^{-\kappa}$, which means that $\sigma > \kappa + \log_2 \beta$.

E. Complexity

Communication and Round Complexity. In the private membership test phase, the sender sends the key k_2 to the aider, and simultaneously sends the key k_1 together with the OKVS table T to the receiver. According to Section II-E, the OKVS encoding achieves a near-optimal rate of 1.28, resulting in a communication complexity of $O(\beta)$. This step requires one round. The receiver then exchanges $\mathcal{Q}$ and $\mathcal{Q}'$ with the aider, incurring a communication complexity of $O(\beta)$ in one round. Thus, this phase requires $O(\beta)$ communication and 2 rounds. In the replicated secret sharing phase, the parties exchange vectors $\vec{y}_1$, $\vec{y}_2$, and $\vec{y}_3$, incurring $O(\beta)$ communication in one round. Finally, the protocol invokes $\mathcal{F}^{\beta}_{\text{EQZ}}$. According to Section II-D, this functionality incurs a communication complexity of $O(\beta\sigma)$ and requires $O(\log\sigma)$ rounds. According to Section III-D, we choose the smallest integer σ such that $\sigma > \kappa + \log\beta$, i.e., $\sigma = \lceil \kappa + \log\beta \rceil$. Thus, the total communication complexity is $O(\beta\kappa)$, and the round complexity is $O(\log\kappa)$.

Computation Complexity. In the private membership test phase, the sender computes 2β PRF evaluations and performs β encodings, while the receiver and aider compute 2β PRF evaluations in total and perform β decodings. According to Section II-E, both the OKVS encoding and decoding are linear, yielding a complexity of $O(\beta)$. In the replicated secret sharing phase, computing $\vec{y}_1$, $\vec{y}_2$, and $\vec{y}_3$ is linear, resulting in a complexity of $O(\beta)$. Finally, the protocol invokes $\mathcal{F}^{\beta}_{\text{EQZ}}$, which incurs a computation complexity of $O(\beta\sigma)$. Substituting $\sigma = \lceil \kappa + \log\beta \rceil$, we obtain a total computation complexity of $O(\beta\kappa)$.

IV. Threshold Multi-Party Set Intersection

A. Protocol Overview

In prior T-MPSI protocols, the same set of parties jointly executes both the secure query and secure comparison phases, meaning that all parties are involved in both phases. As a result, the computationally expensive secure comparison must be performed by all participants, leading to high overhead. In contrast, our protocol decouples these two phases by assigning them to different sets of parties. Specifically, all parties participate in the secure query phase, while only a small subset of parties, namely Q, D_1, and D_2, performs the secure comparison. To enable this secure decoupling, we introduce a secure aggregation algorithm that securely transfers the intermediate results from all parties to the designated comparison parties. In the following, we describe our protocol in detail.

Our T-MPSI protocol under the designated k-collusion model is composed of two modules: **Secure Query** and **Secure Comparison**. In the Secure Query module, the leader computes the number of occurrences across all the datasets for each element. In the Secure Comparison module, the target elements are determined by checking whether the occurrence count of each element exceeds the threshold. To provide a clearer exposition of the workflow, we further decompose these two modules into six phases. Specifically, the **Secure Query** module is divided into four phases: Bucketing, Private Membership Test, Secure Conversion, and Secure Aggregation; and the **Secure Comparison** module is divided into two phases: Secure Comparison and Output. The detailed procedures of each phase are described as follows:

Bucketing. This phase is designed to reduce the complexity of the private membership test. To test whether each element in $\mathcal{Q}$ belongs to dataset $\mathcal{D}_i$, we need to perform a membership test that compares each element of $\mathcal{Q}$ against the entire dataset $\mathcal{D}_i$. To reduce the complexity, each dataset $\mathcal{D}_i$ is partitioned into multiple smaller buckets, so that the membership test for each element only needs to be performed within a subset of the dataset rather than the entire dataset. Specifically, the leader and data holders select h hash functions $H_1, \ldots, H_h$ and initialize table $T, T_1, \ldots, T_{n-1}$, each with β buckets. The leader employs cuckoo hashing, where each element of $\mathcal{Q}$ is inserted into table T according to one of the h hash functions. Each data holder D_i applies simple hashing, inserting every element of $\mathcal{D}_i$ into all h buckets corresponding to the hash

functions in table T_i. Therefore, we only need to test whether the element $T[j]$ belongs to the set $T_i[j]$, where j is the index of the query element in table T.

Private Membership Test. This phase aims to check whether each query element in table T existing in the corresponding bucket of the table T_i. We invoke the ab-rssPMT protocol as follows. The leader Q acts as the receiver, inputting the table T as the query vector. The data holder acts as the sender, inputting the table T_i as the set vector. We designate the non-colluding party D_1 as the fixed aider, except in cases where it acts as a sender, in which case the aider role is taken over by the other non-colluding party D_2. After executing the ab-rssPMT protocol, the leader obtains $n-1$ boolean replicated secret share vectors; D_1 obtains $n-1$ boolean replicated secret share vectors; D_2 obtains 2 boolean replicated secret share vectors; and each of $D_3, \ldots, D_{n-1}$ obtains 1 boolean replicated secret share vector.

Secure Aggregation. The goal of this phase is to securely transform the secret share vectors distributed among n parties into a form suitable for three-party secure comparison based on replicated secret sharing. We designate the three non-colluding parties Q, D_1, and D_2 as the computation nodes for secure comparison, which are required to hold all $n-1$ replicated secret share vectors. To achieve this, the parties $D_3, \ldots, D_n$ send their vectors to D_2. Through this one-round communication, the three non-colluding parties Q, D_1, and D_2 collectively hold all $n-1$ replicated secret share vectors.

Secure Conversion. The goal of this phase is to compute the number of occurrences of each query element in table T across the tables $T_1, \ldots, T_{n-1}$. We first invoke the functionality $\mathcal{F}^{\beta}_{\text{B2A}}$ to convert the $n-1$ secret share vectors over $\mathbb{Z}_2^{\beta}$ into vectors over $\mathbb{Z}_{2^l}^{\beta}$, where $2^l > n-1$. By summing these $n-1$ share vectors, we obtain $\langle \vec{x} \rangle$, where each entry indicates the number of occurrences of the corresponding query element in the datasets held by the data holders.

Secure Comparison. This phase determines whether each query element in table T appears in at least t of the tables $T_1, \ldots, T_{n-1}$. To this end, we compare the replicated secret-shared vector $\langle \vec{x} \rangle$ against the threshold t, and obtain a boolean vector $\vec{z}$, where each entry indicates whether the corresponding element satisfies $\langle x_j \rangle \geq t$. Specifically, we employ the circuit-based secure comparison protocol GTZ^{β} from [20] to check whether $\langle x_j \rangle \geq t$ for each $\langle x_j \rangle \in \langle \vec{x} \rangle$. Finally, we invoke $\mathcal{F}^{\beta}_{\text{Reveal}}$ to let the leader obtain the plaintext results $\vec{z}$.

Output: In this phase, the leader uses the result vector $\vec{z}$ to identify which query elements in T appear in at least t of the datasets. This only needs the leader to map the indices of the entries in $\vec{z}$ that are equal to 1 to the corresponding query elements in table T.

We provide the formal description in Protocol IV.1.

B. Correctness and Security

Theorem 3. *Protocol $\pi^{n,m_1,m_2,t}_{\text{T-MPSI}}$ is correct.*

Proof. For $q \in \mathcal{Q}$, define $\eta_q = \{i \in [1, n-1] : q \in \mathcal{D}_i\}$. We assume that the cuckoo hashing performed by Q succeeds with a probability of at least $1 - 2^{-\kappa}$. We now consider the following cases.

$q \in \mathcal{Q}$ and $|\eta_q| \geq t$. By the properties of cuckoo hashing and simple hashing, there exists an index $j \in [\beta]$ such that the element q is placed into bucket $T[j]$ as well as bucket $T_i[j]$ for each $i \in \eta_q$. According to the correctness of the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$, we obtain shares $\langle x_{ij} \rangle_B = \langle 1 \rangle_B$ for $i \in \eta_q$, and $\langle x_{ij} \rangle_B = \langle 0 \rangle_B$ for $i \notin \eta_q$. Then, by the correctness of $\mathcal{F}^{\beta}_{\mathsf{B2A}}$ and $\mathcal{F}_{\mathsf{Add}}$, we obtain $\langle x_j \rangle = \sum_{i=1}^{n-1} \langle x_{ij} \rangle = \langle |\eta_q| \rangle$. Since $|\eta_q| \geq t$, we have $x_j - t \geq 0$. By the correctness of $\mathcal{F}^{\beta}_{\mathsf{GTZ}}$ and $\mathcal{F}^{\beta}_{\mathsf{Reveal}}$, it follows that $z[j] = 1$, and hence $q \in Y$.

$q \in \mathcal{Q}$ and $|\eta_q| < t$. By the properties of cuckoo hashing and simple hashing, there exists an index $j \in [\beta]$ such that the element q is placed into bucket $T[j]$ as well as bucket $T_i[j]$ for each $i \in \eta_q$. According to the correctness of the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$, we obtain shares $\langle x_{ij} \rangle_B = \langle 1 \rangle_B$ for $i \in \eta_q$, and $\langle x_{ij} \rangle_B = \langle 0 \rangle_B$ for $i \notin \eta_q$. Then, by the correctness of $\mathcal{F}^{\beta}_{\mathsf{B2A}}$ and $\mathcal{F}_{\mathsf{Add}}$, we obtain $\langle x_j \rangle = \sum_{i=1}^{n-1} \langle x_{ij} \rangle = \langle |\eta_q| \rangle$. Since $|\eta_q| < t$, we have $x_j - t < 0$. By the correctness of $\mathcal{F}^{\beta}_{\mathsf{GTZ}}$ and $\mathcal{F}^{\beta}_{\mathsf{Reveal}}$, it follows that $z[j] = 0$, and hence $q \notin Y$.

$q \notin \mathcal{Q}$. Suppose q is the random element inserted into table T. We argue that for such the random element in table T, the corresponding value $z[j] = 0$. This is because the real elements are all distinct from q, and hence, with overwhelming probability, the element at position j in $T_i[j]$ differs from q. □

Theorem 4. *Protocol $\pi^{n,m_1,m_2,t}_{\text{T-MPSI}}$ securely computes functionality $\mathcal{F}^{n,m_1,m_2,t}_{\text{T-MPSI}}$ in the $(\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}, \mathcal{F}^{\beta}_{\text{B2A}}, \mathcal{F}^{\beta}_{\text{Add}}, \mathcal{F}^{\beta}_{\text{GTZ}}, \mathcal{F}^{\beta}_{\text{Reveal}})$-hybrid model and in the presence of a semi-honest adversary who can corrupt any subset $\{D_3, \ldots, D_{n-1}\}$ or Q or D_1 or D_2.*

Proof. We simulate the view of corrupted parties. We consider the following cases.

Corrupted Q. The simulator is given $\mathcal{Q}$ and the protocol's output Y as input. The bucketing phase is performed locally, the simulator can simulate this phase using $\mathcal{Q}$ to generate table T. In the private membership test phase, the parties Q, D_1, D_2 and Q, D_i, D_1 (for $i \in [2, n-1]$) invoke the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$, where Q obtains $n-1$ replicated secret share vectors in $\mathbb{Z}_2^{\beta}$. To simulate this, the simulator selects random boolean vectors $\vec{x}_i$ (for $i \in [1, n-1]$) and generates the corresponding shares in $\mathbb{Z}_2^{\beta}$ for Q. In the secure conversion phase, parties Q, D_1, D_2 invoke the functionality $\mathcal{F}^{\beta}_{\text{B2A}}$ to convert $\langle \vec{x}_i \rangle_B$ to $\langle \vec{x}_i \rangle$, where Q obtains the share of $\vec{x}_i$ in ring $\mathbb{Z}_{2^l}^{\beta}$. The simulator can generates the corresponding shares in $\mathbb{Z}_{2^l}^{\beta}$ based on the vector $\vec{x}_i$. Additionally, Q, D_1, D_2 invoke the functionality $\mathcal{F}^{\beta}_{\text{Add}}$ to compute the sum of the shares of $\vec{x}_i$. The simulator simulates this process by summing the shares of $\vec{x}_i$ to obtain a share of the vector $\vec{x}$. In the secure comparison phase, Q invokes the functionalities $\mathcal{F}^{\beta}_{\text{GTZ}}$ and $\mathcal{F}^{\beta}_{\text{Reveal}}$, where Q only receives a share of the vector. The simulator simulates this process as follows: it first computes the $\vec{y} = \vec{x} - t$. Then, for each $j \in [\beta]$, it sets $z_j = 1$ if $T[j] \in Y$, and $z_j = 0$ otherwise. Finally, the simulator generates the corresponding shares for $\vec{y}$ and $\vec{z}$ for Q. The security of $\mathcal{F}^{\beta}_{\text{GTZ}}$ ensures that the generated shares are computationally indistinguishable from those in the real execution. The simulator uses the vector $\vec{z}$ as

Protocol IV.1. $\pi^{n,m_1,m_2,t}_{\text{T-MPSI}}$

Parameters:

- There are n parties: a leader Q and $n-1$ data holders $D_1, \ldots, D_{n-1}$. The leader holds a query set $\mathcal{Q}$ of size m_1, each data holder holds a dataset $\mathcal{D}_i$ of size m_2.
- Three-party functionalities executed over both the binary field $\mathbb{Z}_2$ and the ring $\mathbb{Z}_{2^l}$.
- The threshold is $t \in [1, n-1]$.

Secure Query:

1) Bucketing:
   - Parties Q and $D_1, \ldots, D_{n-1}$ agree on h hash functions $H_1, \ldots, H_h : \{0,1\}^* \to [\beta]$, where $\beta = \varepsilon m_1$ and ε denotes the expansion factor.
   - Party Q invokes cuckoo hashing. In cuckoo hashing, the hash functions $H_1, \ldots, H_h$ map each element of the set $\mathcal{Q}$ into a table T with β buckets, where each bucket can store only one element. Then, party Q inserts random elements into empty buckets.
   - For $i \in \{1, \ldots, n-1\}$, party D_i invokes simple hashing. In simple hashing, the hash functions $H_1, \ldots, H_h$ map each element of set $\mathcal{D}_i$ into table T_i with β buckets, where each bucket can store multiple elements. For each element $d \in \mathcal{D}_i$, if $H_1(d), \ldots, H_h(d)$ are not distinct, party D_i inserts random elements into the collision buckets. Table T_i contains $N = hm_2$ elements.
2) Private Membership Test:
   - Parties Q, D_1, D_2 invoke the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ as follows:
     - Party Q (receiver) inputs β buckets $\{T[j]\}_{j=1}^{\beta}$ of table T as the query vector; party D_1 (sender) inputs β buckets $\{T_1[j]\}_{j=1}^{\beta}$ of table T_1 as the vector of sets; party D_2 (aider) inputs nothing.
     - Parties Q, D_1, D_2 receive $\langle \vec{x}_1 \rangle_B$.
   - For $i \in [2, n-1]$, parties Q, D_i, D_1 invoke the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ as follows:
     - Party Q (receiver) inputs β buckets $\{T[j]\}_{j=1}^{\beta}$ of table T as the query vector; party D_i (sender) inputs β buckets $\{T_i[j]\}_{j=1}^{\beta}$ of table T_i as the vector of sets; party D_1 (aider) inputs nothing.
     - Parties Q, D_i, D_1 receive $\langle \vec{x}_i \rangle_B$.
3) Secure Aggregation: For $i \in [3, n-1]$, party D_i sends its sharing of $\vec{x}_i$ to party D_2. Parties Q, D_1, D_2 obtain $\langle \vec{x}_1 \rangle_B, \ldots, \langle \vec{x}_{n-1} \rangle_B$.
4) Secure Conversion: Parties Q, D_1, D_2 convert the replicated shares from $\mathbb{Z}_2$ to $\mathbb{Z}_{2^l}$ as follow:
   - For $i \in [1, n-1]$, parties Q, D_1 and D_2 input $\langle \vec{x}_i \rangle_B$ into the functionality $\mathcal{F}^{\beta}_{\text{B2A}}$ and obtain $\langle \vec{x}_i \rangle$.
   - Parties Q, D_1 and D_2 invoke the functionality $\mathcal{F}^{\beta}_{\text{Add}}$ to compute $\langle \vec{x} \rangle = \sum_{i=1}^{n-1} \langle \vec{x}_i \rangle$.

Secure Comparison:

5) Secure Comparison: Parties Q, D_1 and D_2 jointly perform the following steps for secure comparison:
   - Input $\langle \vec{x} \rangle$ and threshold t invoke the functionality $\mathcal{F}^{\beta}_{\text{Add}}$ to compute $\langle \vec{y} \rangle = \langle \vec{x} \rangle - t$.
   - Input $\langle \vec{y} \rangle$ and invoke the functionality $\mathcal{F}^{\beta}_{\text{GTZ}}$ to compute $\langle \vec{z} \rangle = \mathsf{GTZ}^{\beta}(\langle \vec{y} \rangle)$.
   - Input $\langle \vec{z} \rangle$ and invoke the functionality $\mathcal{F}^{\beta}_{\text{Reveal}}$, then output $\vec{z}$ to party Q.
6) Output: For each $j \in [\beta]$, party Q outputs:

$$Y = \bigcup_{z[j]=1} T[j],$$

where $T[j]$ is the j-th element of table T.

the simulated output of functionality $\mathcal{F}^{\beta}_{\text{Reveal}}$. This completes this simulation.

Corrupted D_1. The simulator is given $\mathcal{D}_1$ as input. The bucketing phase is local, the simulator can simulate this phase using the $\mathcal{D}_1$ to generate table T_1. The simulation of private membership test, secure conversion and secure comparison phases are same to the case of corrupted Q, except that the simulator generates the vector $\vec{z}$ by selecting it at random. This completes the simulation as D_1 does not receive any further messages in the subsequent phases.

Corrupted D_2. The simulator is given $\mathcal{D}_2$ as input. The bucketing phase is local, the simulator can simulate this phase using the $\mathcal{D}_2$ to generate table T_2. In the private membership test phase, the parties Q, D_1, D_2 invoke the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$, where D_2 obtains 2 replicated secret share vectors in $\mathbb{Z}_2^{\beta}$. To simulate this, the simulator selects random boolean vectors $\vec{x}_1, \vec{x}_2$ and generates the corresponding shares in $\mathbb{Z}_2^{\beta}$ for D_2. In the secure aggregation phase, D_2 receive the shares from $D_3, \ldots, D_{n-1}$. The simulator can simulate this phase by selecting random boolean vectors $\vec{x}_i$ and generating the corresponding replicated shares for D_2 for each $i \in [3, n-1]$. The security by the functionalities $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ ensures that the

shares is indistinguishable from those in the real world. The secure conversion phase and the secure comparison phase, are identical to the case where Q is corrupted, except that the simulator generates the vector $\vec{z}$ by selecting it at random. This completes the simulation as D_2 does not receive any further messages in the subsequent phases.

Corrupted $D_3, \ldots, D_{n-1}$. The simulator is given $\mathcal{D}_3, \ldots, \mathcal{D}_{n-1}$ as input. The bucketing phase is local, the simulator can simulate this phase using the $\mathcal{D}_3, \ldots, \mathcal{D}_{n-1}$ to generate table $T_3, \ldots, T_{n-1}$. In the private membership test phase, for $i \in [3, n-1]$, the parties Q, D_i, D_1 invoke the functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$, where D_i obtains a share of $\vec{x}_i$ in $\mathbb{Z}_2^\beta$. To simulate this, the simulator selects random boolean vectors $\vec{x}_3, \ldots, \vec{x}_{n-1}$ and generates the corresponding shares in $\mathbb{Z}_2^\beta$ for D_i. This completes the simulation as D_i does not receive any further messages in the subsequent phases. □

C. Parameter Selection

Ring Parameter. In the protocol $\pi^{n,m_1,m_2,t}_{\text{T-MPSI}}$, the ring $\mathbb{Z}_{2^l}$ is used for secure computation. To ensure correctness, we require that $2^l > n-1$.

Cuckoo Hashing Parameters. In the bucketing phase of protocol $\pi^{n,m_1,m_2,t}_{\text{T-MPSI}}$, party Q invokes a cuckoo hash function to insert m elements into a table. To ensure successful execution of the bucketing phase, the failure probability of cuckoo hashing must be bounded by $2^{-\kappa}$. The work of [32] provides an experimental analysis of the relationship among the number of hash functions h, the number of inserted elements m, the expansion factor ε, and the statistical security parameter κ for cuckoo hashing. In particular, when $h = 3$, the failure probability satisfies $\kappa = 123.5\varepsilon + (-130 - \log_2 m)$. In this paper, we provide concrete parameters used in our implementation, assuming a security parameter of $\kappa = 40$.

TABLE IV
CUCKOO HASH PARAMETER

m	10^3	10^4	10^5	10^6
ε	1.45	1.48	1.51	1.54

V. Performance Evaluation

In this section, we compare our protocol with several representative protocols in [14], [15], [23] and a generic circuit-based approach, denoted by CPSI + Semi2k [35]. The protocol in [14] achieves optimal collusion resistance. The protocol of [15] is a state-of-the-art protocol with optimal efficiency. For [23], we consider its non-colluding deployment, which achieves optimal round complexity.

Since the protocols in [14], [15], [23] differ in design, we categorize their phases into two modules, secure query and secure comparison, for ease of comparison. The detailed phases of each protocol and their correspondence to these two modules are summarized in Table V.

TABLE V
MAPPING OF PROTOCOL PHASES TO THE TWO MODULES

	Secure Query	**Secure Comparison**
[14]	Initialization, Local EBFs generation, Set Intersection generation by the server (1-3)	Set Intersection generation by the server (4-12), Output
[15]	Hashing, $\mathcal{F}^{\beta,\sigma,N}_{\text{w-PSM}}$, $\mathcal{F}^{\sigma}_{\text{EQ}}$, $\mathcal{F}^{\mathbb{F}}_{\text{B2A}}$	Pre-processing, n-party functionalities, Output
[23]	Share Generation	Reconstruction, Output
Ours	Bucketing, Private Membership Test, Secure Conversion, Secure Aggregation	Secure Comparison, Output

A. Theoretical Analysis

In this subsection, we analyze the theoretical communication complexity of the two modules in our work: secure query and secure comparison. We instantiate the functionalities $\mathcal{F}^{\beta}_{\text{EQZ}}$, $\mathcal{F}^{\beta}_{\text{GTZ}}$, $\mathcal{F}^{\beta}_{\text{B2A}}$, and $\mathcal{F}^{\beta}_{\text{Mul}}$ as described in Section II-D, the OKVS as described in Section II-E, and $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ as described in Section III-B, and the cuckoo hash parameters as specified in Section IV-C. For convenience, we assume that all parties hold datasets of equal size m, i.e., $|\mathcal{Q}| = |\mathcal{D}_1| = \cdots = |\mathcal{D}_{n-1}| = m$.

Communication and Round Complexity of Secure Query. In the private membership test phase, the protocol invokes the $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ functionality $n-1$ times in parallel. The functionality $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ incurs a communication complexity of $O(\beta\kappa)$ and requires $O(\log\kappa)$ rounds. Since $\beta = \epsilon m$, this phase achieves a total communication complexity of $O(nm\kappa)$ and a round complexity of $O(\log\kappa)$. In the secure aggregation phase, parties $D_3, \ldots, D_{n-1}$ send shares to D_2, which requires one round of communication and incurs a communication cost of $O(mn)$. In the secure conversion phase, the protocol invokes the $\mathcal{F}^{\beta}_{\text{B2A}}$ functionality $n-1$ times in parallel, resulting in a communication complexity of $O(mnl)$ and $O(1)$ rounds, where $n-1 < 2^l$. We choose the smallest integer l such that $2^l > n-1$, i.e., $l = \lceil \log_2(n-1) \rceil \ll \kappa$. Consequently, the overall communication complexity of the secure query module is $O(mn\kappa)$, and the round complexity is $O(\kappa)$. In this work, we set the statistical security parameter to $\kappa = 40$. Therefore, the communication complexity is $O(mn)$ and the round complexity is $O(1)$.

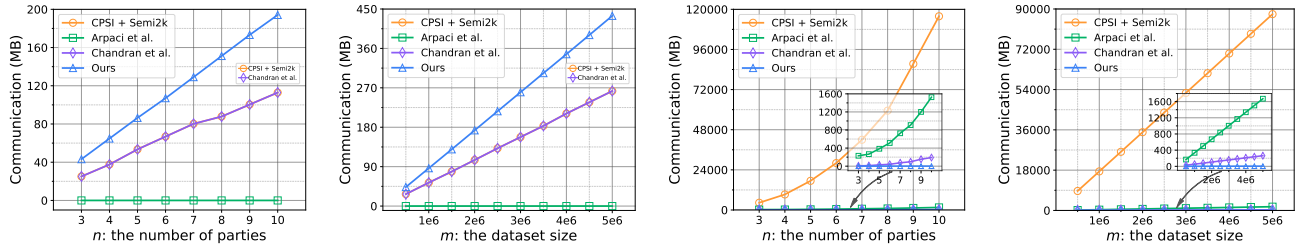
Computation Complexity of Secure Query. In the bucketing phase, the parties perform cuckoo hashing and simple hashing, which incurs a computation complexity of $O(mn)$. In the private membership test phase, the protocol invokes the $\mathcal{F}^{\beta,N}_{\text{ab-rssPMT}}$ functionality $n-1$ times, resulting in a computation complexity of $O(nm\kappa)$. In the secure conversion phase, the protocol invokes the $\mathcal{F}^{\beta}_{\text{B2A}}$ functionality $n-1$ times, resulting in a computation complexity of $O(mn)$. Therefore, the overall computation complexity of the secure query module is $O(nm\kappa)$, which simplifies to $O(mn)$ when κ is set to 40.

Communication and Round Complexity of Secure Comparison. Parties Q, D_1, and D_2 invoke the functionalities $\mathcal{F}^{\beta}_{\text{Add}}$, $\mathcal{F}^{\beta}_{\text{GTZ}}$, and $\mathcal{F}^{\beta}_{\text{Reveal}}$ over the ring $\mathbb{Z}_{2^l}$. According to Section II-D, these functionalities incur a communication complexity of $O(ml)$ and require $O(\log l)$ rounds. In this work, we set the parameter $l = 8$. Therefore, the communication

TABLE VI
RUNTIME UNDER LAN AND WAN FOR DIFFERENT NUMBERS OF PARTIES n AND DATASET SIZES m.

		m	1e3			1e6			2e6			5e6		
		n	3	6	9	3	6	9	3	6	9	3	6	9
LAN (ms)	Secure Query	CPSI + Semi2k	260	644	1028	7174	16502	26794	13429	33565	53572	34593	83631	135117
		[14]	45920	114550	182896	–	–	–	–	–	–	–	–	–
		[23]	1520	2520	3670	62800	155000	279000	126000	312000	560000	319000	794000	1317000
		[15]	260	644	1028	7174	16502	26794	13429	33565	53572	34593	83631	135117
		Ours	**9**	**23**	**39**	**1048**	**2484**	**3918**	**2191**	**5051**	**7977**	**5502**	**12836**	**20392**
	Secure Comparison	CPSI + Semi2k	407	1854	7257	34466	152565	579940	66369	314773	1159124	166164	782349	2919114
		[14]	109225	393873	863287	–	–	–	–	–	–	–	–	–
		[23]	715	918	1037	1104	2800	21115	1483	4700	42662	3905	12580	108560
		[15]	3.73	5.21	14.28	70	202	822	127	387	1506	305	969	3688
		Ours	**0.37**	**0.37**	**0.37**	**19**	**19**	**19**	**39**	**39**	**39**	**95**	**95**	**95**
WAN (s)	Secure Query	CPSI + Semi2k	4.80	11.94	19.09	13.98	34.13	53.77	21.62	53.65	84.68	45.34	111.69	182.23
		[14]	46.04†	114.70†	183.09†	–	–	–	–	–	–	–	–	–
		[23]	**1.52**†	**2.52**†	**3.67**†	**6.28**†	**15.55**†	**27.9**†	12.6†	31.2†	56.0†	31.9†	79.4†	131.7†
		[15]	4.80	11.94	19.09	13.98	34.13	53.77	21.62	53.65	84.68	45.34	111.69	182.23
		Ours	2.44	6.02	9.56	7.72	20.26	32.55	**11.08**	**28.96**	**46.64**	**21.18**	**57.95**	**86.51**
	Secure Comparison	CPSI + Semi2k	4.24	17.36	54.37	209.75	1276.84	–	415.92	–	–	1046.56	–	–
		[14]	109.54†	394.58†	864.44†	–	–	–	–	–	–	–	–	–
		[23]	**0.89**†	1.23†	1.57†	9.09†	25.16†	65.24†	17.37†	49.33†	130.82†	43.48†	124.01†	328.81†
		[15]	2.52	5.33	8.14	4.08	9.23	14.37	7.20	17.00	26.80	16.56	40.33	64.11
		Ours	1.02	**1.02**	**1.02**	**1.14**	**1.14**	**1.14**	**1.20**	**1.20**	**1.20**	**1.25**	**1.25**	**1.25**

– indicates that the experiment was not evaluated due to memory limitations.
† indicates that the WAN running time is estimated.



(a) (b) (c) (d)

Fig. 2. Comparison of communication cost (MB) for varying numbers of parties and dataset sizes: (a) secure query with different numbers of parties; (b) secure query with different dataset sizes; (c) secure comparison with different numbers of parties; (d) secure comparison with different dataset sizes.

complexity is $O(m)$, while the round complexity is $O(1)$.

Computation Complexity of Secure Comparison. The computation complexity is the same as the communication complexity. The secure comparison module incurs a computation complexity of $O(ml)$. Therefore, when $l = 8$, the computation complexity is $O(m)$.

B. Practical Analysis

Our experiments are conducted on a machine running Ubuntu 20.04.1 LTS with GCC 9.4.0, equipped with an AMD Ryzen 5 4600H processor (3.00 GHz) and 3 GB of RAM. We evaluate both runtime and communication performance under different network settings by emulating real-world bandwidth and latency using the tc command. Specifically, the LAN setting has a round-trip time (RTT) of 0.02 ms and a bandwidth of 10 Gbps, while the WAN setting has an RTT of 96 ms and a bandwidth of 200 Mbps. The computational security parameter is set to $\lambda = 128$, and the statistical security parameter is set to $\kappa = 40$. We set $\sigma = 64$ and $l = 8$. The query threshold is set to $t = \lfloor \frac{n-1}{2} \rfloor$. For simplicity, we assume that all parties hold datasets of equal size m, i.e., $|\mathcal{Q}| = |\mathcal{D}_1| = \cdots = |\mathcal{D}_{n-1}| = m$. Throughout this subsection, we use the notation 1e6 to denote 1×10^6, following standard scientific notation.

We implement our protocol using VolePSI[1] for the OKVS functionality and cryptoTools[2] for secure computation based on replicated secret sharing.

Since [15] does not provide an open-source implementation, we implement it using VolePSI and MP-SPDZ[3]. The CPSI + Semi2k baseline follows the same implementation setting as [15].

The open-source implementations of [14], [23] only support local (LAN) execution and do not include communication. Therefore, their WAN running time is estimated by $T_{\text{WAN}} = T_{\text{LAN}} + \frac{\text{Comm}}{\text{Bandwidth}} + \text{RTT} \times R$, where Comm denotes the communication cost and R is the number of communication rounds.

Runtime. Table VI reports the runtime under LAN and WAN for different dataset sizes and numbers of parties. We report runtimes in milliseconds for LAN and in seconds for WAN.

[1]https://github.com/Visa-Research/volepsi
[2]https://github.com/ladnir/cryptoTools
[3]https://github.com/data61/MP-SPDZ

In the LAN setting, our protocol consistently outperforms all baselines across all settings. In the secure query module, the performance improvement varies with the dataset size. For small datasets ($m = 1e3$), our protocol achieves approximately 26–29× speedup over CPSI + Semi2k and [15], and up to 94–169× over [23]. As the dataset size increases ($m \geq 1e6$), the speedup over CPSI + Semi2k and [15] stabilizes at around 6–7×, while maintaining a consistent 60–70× advantage over [23]. The protocol in [14] incurs significantly higher overhead due to its public-key operations and is only practical for small datasets. For $m = 1e3$, our protocol achieves over 4000× speedup compared to [14]. For larger datasets, the open-source implementation of [14] runs out of memory and thus cannot be evaluated. In the secure comparison module, our protocol achieves substantially larger performance gains. Compared to CPSI + Semi2k, our protocol achieves speedups ranging from 10^3 to over 3×10^4, with the gap increasing as the number of parties grows. Compared to [23], our protocol achieves improvements from 40 to over 2×10^3 times. Even against the optimized protocol of [15], our approach achieves 3–40× speedup.

In the WAN setting, our protocol continues to demonstrate clear advantages, especially for large datasets. Due to the impact of network latency and bandwidth constraints, the performance gap between different protocols becomes more sensitive to communication efficiency. In the secure query module, our protocol outperforms CPSI + Semi2k and [15]. For small datasets ($m = 1e3$), our protocol achieves around 2× speedup. As the dataset size increases, the improvement remains stable at approximately 1.6–2×. This gain primarily stems from the fact that our protocol requires fewer communication rounds. Compared with [23], the advantage is less pronounced under WAN, since [23] relies solely on local computation. As a result, our protocol is slightly slower for small datasets (by about 1.5–2.6×), but becomes competitive for larger datasets, achieving 1.1–1.5× speedup when $m \geq 2e6$. In the secure comparison module, our protocol demonstrates substantial advantages. Compared to CPSI + Semi2k, our protocol achieves up to two to three orders of magnitude improvement. For instance, when $m = 1e6$, CPSI + Semi2k requires over 1276 seconds, whereas our protocol completes in only 1.14 seconds. Compared with [15], our protocol achieves speedups ranging from 5× to over 50×, with the advantage increasing as the dataset size grows. Compared to [23], our protocol becomes faster when $m \geq 2e6$, achieving approximately 15–250× speedup.

Overall, in the LAN setting, our protocol achieves at least a 6× speedup in the secure query and at least a 3× improvement in the secure comparison. In the WAN setting, our secure query remains efficient for large datasets (achieving at least 1.1× speedup), while the secure comparison phase achieves at least a 5× improvement.

Communication. Fig. 2 shows the communication cost of our protocol under varying numbers of parties and dataset sizes. The communication cost of the secure query module grows linearly with both the number of parties and the dataset size. Meanwhile, the communication cost of the secure comparison module is independent of the number of parties and scales linearly only with the dataset size. Fig. 2 shows that our protocol incurs higher communication cost in the secure query phase. However, the OT-based protocols in [15] and CPSI + Semi2k require more communication rounds, and [23] suffers from high computational overhead. Consequently, our protocol still demonstrates clear advantages under WAN settings, as shown in the runtime. In comparison, our protocol significantly reduces the communication cost, which is particularly beneficial in high-latency network settings.

Overall, while our protocol incurs slightly higher communication cost in the secure query phase, it achieves better scalability and improved end-to-end efficiency by making the communication cost of the secure comparison module independent of the number of parties.

TABLE VII
BREAKDOWN OF COMPUTATION COST ACROSS DIFFERENT BUILDING BLOCKS.

Building Block	Percentage
Cuckoo Hashing & Simple Hashing	∼3.5%
PRF	∼0.3%
OKVS · (Encode + Decode)	∼9.6%
$\mathcal{F}_{\text{EQZ}} + \mathcal{F}_{\text{GTZ}}$	∼86.6%

Breakdown Analysis. Table VII shows the relative contribution of each building block. We observe that the secure comparison functionality dominates the overall runtime, accounting for over 80% of the total cost. In contrast, all other building blocks contribute only a small fraction. Therefore, further optimizations of the secure comparison functionality are expected to yield significant end-to-end performance improvements.

VI. CONCLUSION

To achieve more efficient privacy-preserving user tracking, we focus on improving the efficiency of secure query and secure comparison under the designated k-collusion model. We introduce a lightweight batch replicated secret sharing private membership test protocol, where the private membership test phase relies solely on pseudorandom functions, achieving better performance than the known private membership test protocols. We further develop a one-round secure aggregation algorithm that bridges the gap between secure query and secure comparison built upon replicated secret sharing. By using these techniques, we design an efficient T-MPSI protocol under the designated k-collusion model. The proposed protocol improves secure query efficiency and ensures that the communication complexity of secure comparison remains independent of the number of parties. We formally prove its security and evaluate its performance. The results show at least 6× and 3× improvements in the efficiency of secure query and secure comparison, respectively, compared with the state-of-the-art protocol in a LAN setting. These results demonstrate that the proposed protocol enables efficient privacy-preserving user tracking.

ETHICAL CONSIDERATIONS

We acknowledge that the proposed protocol may introduce risks if misused in large-scale tracking systems. In particular, such systems could be abused to track sensitive groups, including political dissidents, journalists, or minority populations, if deployed without appropriate safeguards. Moreover, there is a risk of harm if the tracking list is not properly controlled, as unauthorized or unregulated use of the query set may enable unjustified surveillance. In practical deployments, the tracking list should be managed exclusively by authorized entities and subject to auditing and accountability mechanisms. Furthermore, while our model considers designated k-collusion, risks may still arise if nominally non-colluding parties (i.e., Q, D_1, D_2) share similar administrative or institutional affiliations. Therefore, this assumption should be carefully evaluated in each deployment context to ensure that it accurately reflects real-world trust relationships.

We strongly encourage the proposed protocol to be used strictly within applicable legal and ethical frameworks. Deployments should be accompanied by appropriate compliance mechanisms to mitigate misuse and ensure accountability. Additionally, we explicitly state the underlying security assumptions of our model to reduce the risk of unsafe or inappropriate deployment.

REFERENCES

[1] G. Chen, P. Liu, Z. Liu, H. Tang, L. Hong, J. Dong, J. Conradt, and A. C. Knoll, "Neuroaed: Towards efficient abnormal event detection in visual surveillance with neuromorphic vision sensor," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 923–936, 2021.

[2] H. Kopetz and W. Steiner, *Real-Time Systems - Design Principles for Distributed Embedded Applications, Third Edition*. Springer, 2022.

[3] J. Ning, Z. Cao, X. Dong, J. Gong, and J. Chen, "Traceable CP-ABE with short ciphertexts: How to catch people selling decryption devices on ebay efficiently," in *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, I. G. Askoxylakis, S. Ioannidis, S. K. Katsikas, and C. Meadows, Eds., vol. 9879. Springer, 2016, pp. 551–569.

[4] J. Kim, J. J. Jang, and I. Y. Jung, "Near real-time tracking of iot device users," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2016, Chicago, IL, USA, May 23-27, 2016*. IEEE Computer Society, 2016, pp. 1085–1088.

[5] Y. Zhu, W. Ma, J. Cui, X. Xia, Y. Peng, and J. Ning, "Pvct: A publicly verifiable contact tracing algorithm in cloud computing," *Secur. Commun. Networks*, vol. 2021, pp. 5 514 137:1–5 514 137:18, 2021.

[6] T. Duong, D. H. Phan, and N. Trieu, "Catalic: Delegated PSI cardinality with applications to contact tracing," in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds. Springer, 2020, pp. 870–899.

[7] Y. Yu, R. Chen, H. Li, Y. Li, and A. Tian, "Toward data security in edge intelligent iiot," *IEEE Netw.*, vol. 33, no. 5, pp. 20–26, 2019.

[8] B. K. Mohanta, D. Jena, and S. Sobhanayak, "Multi-party computation review for secure data processing in iot-fog computing environment," *Int. J. Secur. Networks*, vol. 15, no. 3, pp. 164–174, 2020.

[9] Y. Miao, G. Wang, X. Li, H. Li, K. R. Choo, and R. H. Deng, "Efficient and secure geometric range search over encrypted spatial data in mobile cloud," *IEEE Trans. Mob. Comput.*, vol. 24, no. 3, pp. 1621–1635, 2025.

[10] Z. Zeng, C. Tang, Q. Zhou, Z. Liu, Z. Deng, and D. He, "EFSC: efficient and forward-secure conditional privacy-preserving scheme for internet of vehicles," *IEEE Internet Things J.*, vol. 12, no. 7, pp. 8406–8420, 2025.

[11] 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, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 1257–1272.

[12] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1151–1165.

[13] J. Vos, M. Conti, and Z. Erkin, "Fast multi-party private set operations in the star topology from secure ands and ors," *IACR Cryptol. ePrint Arch.*, p. 721, 2022.

[14] A. Bay, Z. Erkin, J. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1–15, 2022.

[15] 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 *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1182–1204.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238.

[17] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, S. Goldwasser, Ed. ACM, 2012, pp. 309–325.

[18] N. Chandran, D. Gupta, and A. Shah, "Circuit-psi with linear complexity via relaxed batch OPPRF," *Proc. Priv. Enhancing Technol.*, vol. 2022, no. 1, pp. 353–372, 2022.

[19] X. Yang, L. Cai, Y. Wang, K. Yin, L. Sun, and J. Hu, "Efficient unbalanced quorum PSI from homomorphic encryption," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024*, J. Zhou, T. Q. S. Quek, D. Gao, and A. A. Cárdenas, Eds. ACM, 2024.

[20] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 35–52.

[21] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 1, pp. 188–208, 2021.

[22] R. A. Mahdavi, T. Humphries, B. Kacsmar, S. Krastnikov, N. Lukas, J. A. Premkumar, M. Shafieinejad, S. Oya, F. Kerschbaum, and E. Blass, "Practical over-threshold multi-party private set intersection," in *ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 772–783.

[23] O. E. Arpaci, R. Boutaba, and F. Kerschbaum, "Over-threshold multi-party private set intersection for collaborative network intrusion detection," *CoRR*, vol. abs/2510.12045, 2025.

[24] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 1–19.

[25] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 241–257.

[26] A. Miyaji and S. Nishida, "A scalable multiparty private set intersection," in *Network and System Security - 9th International Conference, NSS 2015, New York, NY, USA, November 3-5, 2015, Proceedings*, ser. Lecture Notes in Computer Science, M. Qiu, S. Xu, M. Yung, and H. Zhang, Eds., vol. 9408. Springer, 2015, pp. 376–385.

[27] I. Damgård, K. G. Larsen, and J. B. Nielsen, "Communication lower bounds for statistically secure mpc, with or without preprocessing,"

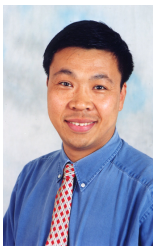
in *Advances in Cryptology – CRYPTO 2019*, ser. Lecture Notes in Computer Science, vol. 11693. Springer, 2019, pp. 61–84.
[28] I. Damgård, B. Li, and N. I. Schwartzbach, "More communication lower bounds for information-theoretic mpc," in *Proceedings of the 2nd Conference on Information-Theoretic Cryptography (ITC 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 199. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 1–18.
[29] S. Ghosh and M. Simkin, "The communication complexity of threshold private set intersection," in *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Boldyreva and D. Micciancio, Eds. Springer, 2019, pp. 3–29.
[30] X. Qian, L. Wei, J. Zhang, and L. Zhang, "Malicious-secure threshold multi-party private set intersection for anonymous electronic voting," *Cryptogr.*, vol. 9, no. 2, p. 23, 2025.
[31] R. Pagh and F. F. Rodler, "Cuckoo hashing," in *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, ser. Lecture Notes in Computer Science, F. M. auf der Heide, Ed., vol. 2161. Springer, 2001, pp. 121–133.
[32] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, "PIR-PSI: scaling private contact discovery," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 4, pp. 159–178, 2018.
[33] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Malkin and C. Peikert, Eds., vol. 12826. Springer, 2021, pp. 395–425.
[34] 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, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 2505–2517.
[35] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10992. Springer, 2018, pp. 769–798.

Bo Zhao received the BS degree from the School of Mathematics and Information Science, Hebei Normal University, China, in 2019, and the MS degree from the School of Mathematics, Shandong University, China, in 2022. From 2022 to 2024, he worked for Huakong Qingjiao Technology Co., Ltd., Beijing, China. He is currently working toward the PhD degree in the School of Mathematics, Shandong University. His research interests include secure multi-party computation and searchable encryption.

Haining Yang received the BS degree from the School of Mathematics and Statistics from Shandong Normal University, China, in 2016, and the PhD degree from the School of Mathematics, Shandong University, China, in 2021. He was a visiting Ph.D. student in School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. He worked as a postdoctoral in State Key Laboratory of Cryptology, Beijing, China, from August, 2021 to August, 2023. He is working in School of Mathematics, Shandong University from September, 2023. His research interest mainly includes public-key cryptography and cloud security.

Jing Qin received the BS degree from Information Engineering University, Zhenzhou, China, in 1982, and the PhD degree from the School of Mathematics, Shandong University, China, in 2004. She is a professor with the School of Mathematics, Shandong University China. Her research interests include information security, design and analysis of security about cryptologic protocols. She has coauthored two books and has published more than 30 professional research papers. She is a senior member of Chinese association for Cryptologic Research (CACR) as well as China Computer Federation (CCF).

Jianting Ning (Member, IEEE) received the Ph.D degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2016. He is currently with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China, and the Faculty of Data Science, City University of Macau, Macau, China. He has published papers in major conferences or journals, such as ACM CCS, NDSS, Asiacrypt, ESORICS, ACSAC, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His research interests include applied cryptography and information security.



Jixin Ma is a Full Professor of Computer Science (Artificial Intelligence) and the Director of PhD/Postgraduate Research Programme in the School of Computing and Mathematical Sciences at University of Greenwich, U.K. He has been the Director of the Centre for Computer and Computational Science and the Lead of Artificial Intelligence Research Group. Professor Ma is also a Visiting Professor of Beijing Normal University, Hainan University, Anhui University, Zhengzhou Light Industrial University and Macau City University. Professor Ma obtained his BSc and MSc of Mathematics in 1982 and 1988, respectively, and PhD of Computer Sciences in 1994. His main research areas include Artificial Intelligence, Data Science, and Information Systems, with special interests in Temporal Logic, Information Security, Machine Learning, Case-Based Reasoning and Pattern Recognition. Professor Ma has been a member British Computer Society, American Association of Artificial Intelligence, ACIS/IEEE, World Scientific and Engineering Society, and Special Group of Artificial Intelligence of BCS. He has also been the Editor of several international journals and international conference proceedings, Conference/Program Chair, and Invited Keynote Speakers of many international conferences. Professor Ma has published more than 200 research papers in peer-reviewed international journals and conferences.