Efficient and Practical Multi-party Private Set Intersection Cardinality Protocol

1st Shengzhe Meng
Beijing Institute of Mathematical Sciences and Applications
and
Tsinghua University
Beijing, China
msz22@mails.tsinghua.edu.cn

2nd Xiaodong Wang
Beijing Institute of Mathematical Sciences and Applications
and
Tsinghua University
Beijing, China
wangxd22@mails.tsinghua.edu.cn

3rd Zijie Lu
Beijing Institute of Mathematical Sciences and Applications
Beijing, China
lzjluzijie@gmail.com

4th Bei Liang
Beijing Institute of Mathematical Sciences and Applications
Beijing, China
lbei@bimsa.cn

***Abstract*—We present an efficient and simple multi-party private set intersection cardinality (PSI-CA) protocol that allows several parties to learn the intersection size of their private sets without revealing any other information. Our protocol is highly efficient because it only utilizes the Oblivious Key-Value Store and zero-sharing techniques, without incorporating components such as OPPRF (Oblivious Programmable Pseudorandom Function) which is the main building block of multi-party PSI-CA protocol by Gao et al. (PoPETs 2024). Our protocol exhibits better communication and computational overhead than the state-of-the-art. To compute the intersection between 16 parties with a set size of 2^{20} each, our PSI-CA protocol only takes 5.84 seconds and 326.6 MiB of total communication, which yields a reduction in communication by a factor of up to 2.4× compared to the state-of-the-art multi-party PSI-CA protocol of Gao et al. (PoPETs 2024). We prove that our protocol is secure in the presence of a semi-honest adversary who may passively corrupt any $(t-2)$-out-of-t parties once two specific participants are non-colluding.**

***Index Terms*—Secure multi-party computation, PSI-CA, OKVS.**

I. Introduction

Private Set Intersection Cardinality (PSI-CA), a variant of the Private Set Intersection (PSI) problem, is a special case of multi-party computation (MPC), which allows several parties to learn the intersection size of their private sets without learning any element in the intersection. PSI-CA has been found as a crucial cryptographic tool in many real-world applications such as private contact tracing platforms related to COVID-19 [1]–[4] and so on. Relying on PSI-CA, Trieu et al. [5] proposed a privacy-preserving contact tracing system, allowing multiple participants (users and healthcare providers) to privately match contact information and notify users who may have been infected.

Moreover, Trieu et al. [5] proposed a specific PSI-CA application scenario that has been widely studied. In this scenario, the health authorities maintain a database of tokens corresponding to users diagnosed with the disease, while every user maintains a small database of tokens corresponding to their close contacts. In this situation, the user does not want to disclose information about their contacts, and the health authorities need to protect the privacy of diagnosed patients. The user only needs to know how many of their contacts have been diagnosed to assess their risk. A user can run a multi-party PSI-CA protocol with health authorities and nearby users to assess the risk of infection. In this case, the user (receiver) is not likely to collude with the health authorities. We will study PSI-CA in this setting, where we assume two specific participants are non-colluding. This is a weaker security model than the general setting where an arbitrary collusion is allowed. PSI-CA can also be applied to privately solve aggregate conversion rates for advertising campaigns [6]. In addition, PSI-CA can be extended to other protocols that allow participants to compute functions on the payloads in the intersection [7]. Very recently, Gao et al. [8] illustrated that PSI-CA can be used to implement and improve the performance of two privacy-preserving applications, including COVID-19 heatmap computation and associated rule learning (ARL).

PSI-CA offers enhanced privacy protection compared to PSI, and as a result, it makes the construction of the protocol more challenging. The studies of PSI-CA are mainly focused on the two-party setting, and over the last several years, two-party PSI-CA protocols have become practical with extremely fast cryptographically secure implementations [1], [2], [4],

This work is supported by National Key R&D Program of China (No. 2021YFB2701304).

[6], [9]. However, there have been only a small number of results dealing with the multi-party setting. Although the state-of-the-art protocol for multi-party PSI-CA [8] only relies on fast symmetric-key primitives, which is much more efficient than prior multi-party PSI-CA protocols that require expensive public-key operations for each item [10] or computation on secret-shared data [11], it might not scale well for a large number of parties over the low and middle range of bandwidth networks due to its massive communication overhead. In this work, we present a newly efficient multi-party PSI-CA protocol that is secure against semi-honest adversaries and achieves a better balance between computation and communication trade-offs.

A. State-of-the-Art for Multiparty PSI-CA

The first multi-party PSI-CA protocol was proposed by Kissner et al. [10]. They leveraged the oblivious polynomial evaluation [12] and additively homomorphic cryptosystem technique to construct a PSI-CA protocol with $O(n^2)$ computation cost and $O(nt^2)$ communication cost, where t is the number of parties, and n is the private set size of each party. Vaidya et al. [13] proposed another protocol from commutative one-way hash functions [14] and reduced the computation cost to $O(nt)$. Mohassel et al. [15] utilized an oblivious switching network [16] to construct a three-party PSI-CA protocol in the honest-majority setting with $O(n)$ communication cost. They then extended their protocol to a multi-party protocol, where each party secretly shared their private inputs with those three parties. Fenske et al. [17] presented a server-aided multi-party PSI-CA protocol from additively homomorphic encryption. Their protocol takes advantage of multiple cloud servers and assumes that at least one cloud server does not collude with all the participants. Jolfaei et al. [18] also proposed a server-aided EO-PSI-CA (fficient outsourced private set intersection cardinality) protocol with one cloud server. Their protocol requires each party to construct an encrypted Bloom filter [19] with ElGamal encryption [20], resulting in high computation costs.

Recently, based on OKVS (Oblivious Key-Value Store) data structure [21] and OPPRF (Oblivious Programmable PRF) [22], Gao et al. [8] present a multi-party PSI-CA protocol paradigm with an assumption that a subset of particular parties does not collude. They offer two variants, the first of which is a server-aided protocol that relies on a non-colluding semi-honest server with no input, while the second is a server-less one that removes the need for an outside server by converting the problem of t-party PSI-CA to the problem of server-aided $(t-1)$-party PSI-CA with the use of an untrusted participant (say P_t) who has a private input. Consequently, their server-less t-party PSI-CA is reduced into a two-party PSI-CA where P_1 acts as a receiver and P_t acts as a sender with n times calculations of zero-sharing function for each party and $(t-2)$ times evaluations of OPPRF. In the implementation of Gao et al.'s server-less t-party PSI-CA, the two-party PSI-CA is a server-aided protocol (described in Protocol 10 by [8]) in which P_2 is assigned the role of the server and required to be non-colluding with both P_1 and P_t. For completeness, we provide their server-aided two-Party PSI-CA, server-aided OPPRF, and server-less multiparty PSI-CA protocols in Section II-E, and Appendix A, B, respectively.

Gao et al. [8] demonstrate that their multi-party PSI-CA protocol is much more efficient than prior protocols. When there are four parties with 2^{23} items each, the total run time of their protocol is 27.2 seconds, while protocol from [15] takes 74 seconds. They also reduce the communication cost by a factor up to $4.76\times$ compared to the protocol proposed by [15].

Although Gao et al.'s protocol [8] takes advantage of symmetric-key techniques and outperforms existing protocols, it relies on a server-aided OPPRF and a zero-sharing protocol, which requires the computation of n times the zero-sharing function and $O(n)$ times PRF for each party. We find out that it can be improved further. We propose a simpler and more efficient protocol that does not need OPPRF, cutting about half of the total communication overhead. Moreover, our proposed protocol only requires parties to agree on a single time zero-sharing in contrast with Gao et al.'s protocol [8] with n times zero-sharing, thereby reducing the computational cost.

B. Our Contributions and Techniques

Our contribution in this work is twofold. First, we propose a novel and efficient multi-party PSI-CA protocol that relies on the OKVS data structure and zero-sharing technique without incorporating components such as OPPRF, which is the main building block of multi-party PSI-CA protocol by Gao et al. [8]. We prove that our protocol is secure in the presence of a semi-honest adversary who may passively corrupt any $(t-2)$-out-of-t parties once two specific participants are non-colluding.

The second contribution is the implementation and evaluation of our protocol. We conduct an extensive experiment over both LAN and WAN and up to 16 parties with up to 2^{20} items each. We provide a comparison of the performance of our protocol to the state-of-the-art [8]. Our experiments show that in all settings we considered, our protocol has less communication and computational cost than the state-of-the-art [8]. For example, in the case of 16 parties with a private set of 2^{20} items each, our protocol reduces communication by a factor up to $2.4\times$ compared to their server-less multi-party PSI-CA protocol, which implies that our protocol offers greater advantages under real-world settings with moderate network bandwidth. In the 20 Mbps network, our protocol runs in 144.06 seconds, which is $2.38\times$ faster than Gao et al.' s [8] that runs in 343.46 seconds.

The main idea of our protocol is shown below.

Main idea: Assume that there are t parties $P_1, \ldots, P_t$, and each party owns a private set with n items, denoted as $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$ for all $i \in [t]$.

In the first step, $P_2, \ldots, P_t$ use the linear OKVS scheme to encode their private set into vectors $T_i \in \mathbb{F}^m$ such that for $i \in [3, t]$, P_i's items are encoded to 0, whereas

P_2's items are encoded to random values. Specifically, for $i \in [3,t]$, P_i's OKVS T_i satisfies that $\mathsf{Decode}(T_i, x_{i,j}) = 0$ ($j \in [n]$), whereas for P_2's OKVS T_2 it satisfies that $\mathsf{Decode}(T_2, x_{2,j}) \in \Gamma$ ($j \in [n]$) where Γ is a set of random values selected by P_2.

In the next step, $P_2, \ldots, P_t$ will send and aggregate their OKVS in P_1. However, sending $\{T_i\}_{i \in [2,t]}$ directly to P_1 is not secure, so each party P_i ($i \in [2,t]$) needs to invoke the zero-sharing protocol to generate r_i, which satisfies $\sum_{i=2}^{t} r_i = 0$. Thus, they can send the masked OKVS vector $T_i + r_i$ to P_1. After P_1 receives $T_i + r_i$, it will aggregate those OKVS together as $T = \sum_{i=2}^{t}(T_i + r_i) = \sum_{i=2}^{t} T_i$.

Then, we can transform the multi-party PSI-CA problem into a two-party PSI-CA problem. Notice that for $x \in X_1$ and $x \in \bigcap_{i=2}^{t} X_i$, it holds that $\mathsf{Decode}(T, x) = \sum_{i=2}^{t} \mathsf{Decode}(T_i, x) \in \Gamma$ according to the homomorphic property of linear OKVS. Thus, P_1 can decode the OKVS T with his private set as $V = \{\mathsf{Decode}(T, x_{1,j})_{j \in [n]}\}$. Then, P_1 and P_2 can invoke a two-party PSI-CA protocol where P_1 is the sender with input set V and P_2 is the receiver with input set Γ. As a result, P_1 obtains $|V \cap \Gamma|$, which equals $|\bigcap_{i=1}^{t} X_i|$ except for a negligible probability.

We can prove that our protocol is secure in the presence of a semi-honest adversary who may corrupt any subset of $\{P_2, \ldots, P_t\}$ or a proper subset of $\{P_1, P_3, \ldots, P_t\}$.

Similar to the protocol in [8], we employ a server-aided two-party PSI-CA protocol, which can be constructed efficiently by viewing another party $P_i \in \{P_3, \ldots, P_t\}$ as the server.[1] We will introduce the detailed version of our protocol in Section III-A. The server-aided two-party PSI-CA protocol $\Pi_{\mathsf{SA\text{-}2\text{-}PSI\text{-}CA}}$ proposed by [8] will be introduced in Section II-E. Our protocol can be easily transformed into a multi-party PSI protocol by replacing the last step with a two-party PSI protocol instead of a PSI-CA protocol.

C. Paper Organization

Section II provides the security model of PSI-CA and introduces the OKVS and zero-sharing technique we used to construct our protocol. In Section III-A, we present our multi-party private set intersection cardinality protocol and the proof of security for it in Section III-B. We present theoretical communication and computation cost in Section III-C and compare our protocol with the state-of-the-art work through code implementation in Section IV.

II. Preliminary

A. Notation

Throughout the paper, we use the following notation: We denote the parties as $P_1, \ldots, P_t$, and their respective input sets as X_i ($i \in [t]$). We use κ, λ to denote the computational and statistical security parameters, respectively. We use $[a]$ to denote the set $\{1, 2, \ldots, a\}$ and $[a,b]$ to denote the set $\{a, a+1, \ldots, b\}$. For some set S, the notation $s \leftarrow S$ means that s is assigned a uniformly random element from S. By $\mathrm{negl}(\kappa)$, we denote a negligible function, i.e., a function f such that $f(\kappa) < 1/p(\kappa)$ holds for any polynomial $p(\cdot)$ and sufficiently large κ.

B. Private Set Intersection Cardinality and Security Model

Private set intersection cardinality (PSI-CA) allows t parties, each holding a set of n items, to learn the intersection size of their private sets without revealing anything else. The ideal functionality for PSI-CA $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$ is presented in Fig. 1. We denote the ideal functionality for 2-party PSI-CA as $\mathcal{F}_{\mathsf{2\text{-}PSI\text{-}CA}}$ specially.

Functionality 1. (PSI Cardinality - $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$)

- **Parameters:** t parties $P_1, \ldots, P_t$; the set size n.
- **Inputs:** P_i's input set $X_i = (x_{i,1}, \ldots, x_{i,n})$.
- **Outputs:** Give P_1 the intersection set size $\left|\bigcap_{i=1}^{t} X_i\right|$.

Fig. 1. Ideal functionality for PSI-CA $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$

Security Model. A semi-honest adversary is one who corrupts parties but follows the protocol as specified. In other words, the corrupt parties run the protocol honestly, but they may try to learn as much as possible from the messages they receive from other parties. Semi-honest adversaries are also considered passive in that they cannot take any actions other than attempting to learn private information by observing a view of protocol execution.

We work in a multi-party setting where the semi-honest corrupt parties may collude. This is modeled by considering a single monolithic adversary that obtains the views of all corrupt parties. The protocol is secure if the joint distribution of those views can be simulated.

Similar to the security assumption in [8], in this work, we present a PSI-CA protocol in the multi-party setting, assuming that a particular subset of parties refrains from collusion. This is a reasonable assumption for real-life applications, especially when performance is critical, so a weaker security guarantee is applied as a trade-off. Specifically, our PSI-CA protocol ensures security in the presence of a semi-honest adversary who may corrupt any subset of $\{P_2, \ldots, P_t\}$ or a proper subset of $\{P_1, P_3, \ldots, P_t\}$.[2]

C. Zero Sharing

In zero-sharing, t parties generate shares of zero. Specifically, it provides the parties with a sharing function $S : \{0,1\}^{\kappa} \times \{0,1\}^{l} \rightarrow \{0,1\}^{l'}$ and a key K_i for party P_i, and satisfies the following properties:

- Correctness. For common seed $x \in \{0,1\}^{l}$, the P_i's random share $s_i = S(K_i, x)$ will satisfy $\oplus_{i=1}^{t} s_i = 0$.
- Privacy. Any coalition of $\tau < t-1$ corrupt parties will not reveal any information about honest party P_i's share s_i.

[1] In this case, for security reasons, we need additional conditions that P_i does not collude with P_1 or P_2.

[2] That means $\{P_1, P_2\}$ do not collude, and $\{P_1, P_3, \ldots, P_t\}$ are not simultaneously corrupted.

The zero-sharing protocol from [22] is given in Fig. 2.

PROTOCOL 1. (Zero-Sharing - Π_{ZS} [22])

- **Parameters:** t parties $P_1, \ldots, P_t$; a PRF $F : \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^\kappa$, a common seed $x \in \{0,1\}^l$.
- **Protocol:**
  1) Each party P_i picks random seeds $r_{i,j}$ for $j = i+1, \ldots, t$ and sends seed $r_{i,j}$ to P_j. The party P_i's key $K_i = (r_{1,i}, \ldots, r_{i-1,i}, \ldots, r_{i,i+1}, \ldots, r_{i,t})$.
  2) To obtain its share, each P_i computes: $S(K_i, x) = \left(\bigoplus_{j=1}^{i-1} F\left(r_{j,i}, x\right)\right) \oplus \left(\bigoplus_{j=i+1}^{t} F\left(r_{i,j}, x\right)\right)$

Fig. 2. The zero-sharing protocol

D. Oblivious Key-Value Store (OKVS)

Definition 1. *A **Key-Value Store (KVS)** is parameterized by a set $\mathcal{K}$ of keys, a set $\mathcal{V}$ of values, and consists of two algorithms:*

- Encode*: Takes as input a set of (k_i, v_i) key-value pairs and outputs an object T (or, with statistically small probability, an error indicator $\perp$).*
- Decode*: Takes as input an object T, a key k, and outputs a value v.*

A KVS is correct if, for all $A \subseteq \mathcal{K} \times \mathcal{V}$ with distinct keys:

$$(k, v) \in A \text{ and } \perp \neq T \leftarrow \mathsf{Encode}(A) \Longrightarrow \mathsf{Decode}(T, k) = v$$

An oblivious key-value store (OKVS) is a data structure that, when the v_i values are random, conceals the k_i values that were used to generate them.

Definition 2. *[21] A KVS is an **oblivious KVS (OKVS)** if, for all distinct $\{k_1^0, \ldots, k_n^0\}$ and all distinct $\{k_1^1, \ldots, k_n^1\}$, if* Encode *does not output $\perp$ for $(k_1^0, \ldots, k_n^0)$ or $(k_1^1, \ldots, k_n^1)$, then the output of $\mathsf{Exp}^{\mathcal{A}}\left(\mathcal{K} = (k_1^0, \ldots, k_n^0)\right)$ is computationally indistinguishable to that of $\mathsf{Exp}^{\mathcal{A}}\left(\mathcal{K} = (k_1^1, \ldots, k_n^1)\right)$, where:*

$\underline{\mathsf{Exp}^{\mathcal{A}}\left(\mathcal{K} = (k_1, \ldots, k_n)\right):}$
(1) for $i \in [n]$: choose uniform $v_i \leftarrow \mathcal{V}$
(2) return $\mathcal{A}\left(\mathsf{Encode}\left(\{(k_1, v_1), \ldots (k_n, v_n)\}\right)\right)$

In our construction, we need an OKVS with homomorphic properties. Specifically, we need $\mathsf{Decode}(\cdot, k)$ to be a linear function for all k.

Definition 3. *An OKVS is **linear** (over a field $\mathbb{F}$) if $\mathcal{V} = \mathbb{F}$ ("values" are elements of $\mathbb{F}$), the output of* Encode *is a vector T in $\mathbb{F}^m$, and the* Decode *function is defined as:*

$$\mathsf{Decode}(T, k) = \langle \mathrm{d}(k), T \rangle \overset{def}{=} \sum_{j=1}^{m} \mathrm{d}(k)_j T_j$$

for some function $\mathrm{d} : \mathcal{K} \to \mathbb{F}^m$. Hence Decode *$(\cdot, k)$ is a linear map from $\mathbb{F}^m$ to $\mathbb{F}$.*

For a linear OKVS, one can view the Encode function as generating a solution to the linear system of equations:

$$\begin{bmatrix} -\mathrm{d}\left(k_1\right)- \\ -\mathrm{d}\left(k_2\right)- \\ \vdots \\ -\mathrm{d}\left(k_n\right)- \end{bmatrix} T^\top = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

We will use the homomorphic property that: If T_1 and T_2 have the same dimension, then we have:

$$\mathsf{Decode}(T_1 \oplus T_2, k) = \mathsf{Decode}(T_1, k) \oplus \mathsf{Decode}(T_2, k)$$

A binary OKVS over a field $\mathbb{F}$ is a special case of a linear OKVS, where the $\mathrm{d}(k)$ vectors are restricted to $\{0,1\}^m \subseteq \mathbb{F}^m$. Then $\mathsf{Decode}(T, k)$ is simply the sum of some positions in T. We generally restrict our attention to $\mathbb{F} = GF\left(2^\ell\right) \cong \{0,1\}^\ell$, in which case the addition operation over $\mathbb{F}$ is XOR of strings. In [23], a binary OKVS is called a probe and XOR of strings (PaXoS) data structure.

Definition 4. *An OKVS is **doubly oblivious** if, for all sets of n distinct keys $\{k_1, \ldots, k_n\} \subseteq \mathcal{K}$ and n values $v_1, \ldots, v_n$ each drawn uniformly at random from $\mathcal{V}$, the encoding $T = \mathrm{Encode}\left(\{\left(k_1, v_1\right), \ldots, \left(k_n, v_n\right)\}\right)$ is statistically indistinguishable from an uniformly random element in $\mathbb{F}^m$.*

To facilitate easier proof of security, we need the additional property denoted as doubly oblivious. Note that being doubly oblivious directly implies being oblivious as, if the output encoding is a uniformly random element, no adversary may distinguish two different output encodings. The OKVS schemes in [24], [25] can both satisfy this property.

E. Server-Aided Two-Party PSI-CA [8]

Gao et al. [8] introduced an efficient server-aided two-party PSI-CA protocol only utilizing symmetric key technique. The protocol runs between a sender $\mathcal{S}$, a receiver $\mathcal{R}$, and a cloud server $\mathcal{C}$. The sender and receiver have a private set $X = \{x_1, \ldots, x_{m_1}\}$ and $Y = \{y_1, \ldots, y_{m_2}\}$, respectively. Both parties will agree on a PRF $F : \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$ before the protocol. The sender will first choose two random PRF keys $(k_1,\ k_2) \in \{0,1\}^\kappa$ and send k_1, k_2 to $\mathcal{R}$, $\mathcal{C}$, respectively. For $y_i \in Y$, the receiver will compute $Y' = \{F(k_1, y_i)\} = \{y_1', \ldots, y_{m_2}'\}$ and send the set Y' to the cloud server. The cloud server will compute $F(k_2, y_j')$ for every $y_j' \in Y'$. Then, it will permute those PRF values with a random permutation and send the permuted set Y'' to the receiver. The sender will also compute $x_i' = F(k_2, F(k_1, x_i))$ for $x_i \in X$. It will also permute those PRF values with a random permutation and send the permuted set X' to the receiver. The receiver can obtain the intersection set size by comparing the PRF values in sets Y'' and X'. It will output $|X' \cap Y''|$ at the end of the protocol. The detailed protocol is shown in Fig. 3.

PROTOCOL 2. (Server-Aided Two-Party PSI-CA $\Pi_{\mathsf{SA\text{-}2\text{-}PSI\text{-}CA}}$ [8])

- **Parameters:** The protocol runs between a sender $\mathcal{S}$, a receiver $\mathcal{R}$ and a server $\mathcal{C}$. $\mathcal{S}$ and $\mathcal{R}$ have input size of m_1 and m_2. A PRF $F : \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$.
- **Inputs:**
  - Sender $\mathcal{S}$ has input $X = \{x_1, \ldots, x_{m_1}\}$.
  - Receiver $\mathcal{R}$ has input $Y = \{y_1, \ldots, y_{m_2}\}$.
  - Cloud $\mathcal{C}$ has no input.
- **Protocol:**
  1) $\mathcal{S}$ chooses random keys $(k_1, k_2) \in \{0,1\}^\kappa$ and send k_1, k_2 to $\mathcal{R}$, $\mathcal{C}$, respectively.
  2) $\mathcal{R}$ computes $Y' = F(k_1, Y)$ and sends Y' to $\mathcal{C}$.
  3) $\mathcal{C}$ computes $Y'' = F(k_2, Y')$ and sends a random permutation π of Y'' to $\mathcal{R}$.
  4) $\mathcal{S}$ sends to $\mathcal{R}$ a random permutation of $X' = \{F(k_2, F(k_1, X))\}$.
  5) $\mathcal{R}$ output $|X' \cap Y''|$.

Fig. 3. Server-Aided Two-Party PSI-CA protocol from [8]

III. Multi-party Private Set Intersection Cardinality

A. Multi-Party PSI-CA Protocol

In this section, we propose a multi-party PSI-CA protocol built on OKVS and zero-sharing techniques. We assume that there are t parties and each party holds a private set of n items $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, for $i \in [t]$. P_1 will act as the receiver who receives the cardinality of the intersection set $|\bigcap_{i=1}^t X_i|$. Each party agrees on a linear OKVS scheme with $\mathrm{d} : \{0,1\}^\ell \to \mathbb{F}^m$ beforehand. Then parties $P_1, \ldots, P_t$ will act as follows: At the beginning, party $P_2, \ldots, P_t$ will agree on a zero sharing function $S : \{0,1\}^\kappa \times \{0,1\}^l \to \mathbb{F}^m$ and a zero sharing seed $z \in \{0,1\}^l$. Then $P_2, \ldots, P_t$ invoke the zero-sharing protocol Π_{ZS} such that P_i receives a key $K_i \in \{0,1\}^\kappa$ for all $i \in [2,t]$. P_i can calculate its zero-shares $r_i := S(K_i, z) \in \mathbb{F}^m$. Then P_2 will generate n random values $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ from $\mathbb{F}$, and construct an OKVS by solving the system below:

$$\begin{bmatrix} -\mathrm{d}(x_{2,1}) - \\ -\mathrm{d}(x_{2,2}) - \\ \vdots \\ -\mathrm{d}(x_{2,n}) - \end{bmatrix} T_2^\top = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_n \end{bmatrix}$$

Then P_2 sends $T_2' = T_2 + r_2$ to P_1. Whereas, for $i \in [3,t]$, P_i will solve the system below:

$$\begin{bmatrix} -\mathrm{d}(x_{i,1}) - \\ -\mathrm{d}(x_{i,2}) - \\ \vdots \\ -\mathrm{d}(x_{i,n}) - \end{bmatrix} T_i^\top = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Then each P_i sends $T_i' = T_i + r_i$ to P_1.

Upon receiving those masked OKVS vectors, P_1 will calculate $T = \sum_{i=2}^t T_i'$. Remind that r_i is a zero-sharing satisfying $\sum_{i=2}^t r_i = 0$, so $T = \sum_{i=2}^t T_i$. Then P_1 will decode the aggregated OKVS T with his private set $\{x_{1,j}\}_{j\in[n]}$, namely $V = \{\mathsf{Decode}(T, x_{1,j})_{j\in[n]}\} = \{v_j\}_{j\in[n]}$. After that, P_1, P_2 will invoke the two-party PSI-CA protocol where P_2 acts as a sender with input Γ and P_1 acts as a receiver with input V. P_1 outputs $|V \cap \Gamma|$ as the intersection set cardinality of t parties. The Multi-Party PSI-CA construction is presented formally in Fig. 4.

PROTOCOL 3. (Multi-Party PSI-CA)

- **Parameters:**
  - The protocol runs between $P_1, \ldots, P_t$ for $t > 2$.
  - A finite field $\mathbb{F} = GF(2^{\lambda + 2\log n})$.
  - Linear OKVS scheme $(\mathsf{Encode}, \mathsf{Decode})$ mapping n items to m slots.
  - Zero-sharing $S : \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^{m\cdot(\lambda+2\log n)}$.
- **Inputs:** P_i's set $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$.
- **Protocol:**
  1) Parties $P_2, \ldots, P_t$ agree on a zero-sharing seed $z \in \{0,1\}^l$ and invoke zero-sharing protocol Π_{ZS}, each party P_i obtains the key K_i and computes $r_i := S(K_i, z) \in \mathbb{F}^m$ such that $\sum_{j=2}^t r_j = 0$.
  2) P_2 generates n random values $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ from $\mathbb{F}$.
  3) P_2 generates an OKVS over $T_2 \leftarrow \mathsf{Encode}(\{x_{2,j}, \gamma_j\}_{j\in[n]})$ and sends $T_2^{'} = T_2 + r_2$ to P_1.
  4) For all $i \in [3,t]$, P_i constructs an KVS over $T_i \leftarrow \mathsf{Encode}(\{x_{i,j}, 0\}_{j\in[n]})$, and sends $T_i^{'} = T_i + r_i$ to P_1.
  5) P_1 computes $T = \sum_{i=2}^t T_i^{'} = \sum_{i=2}^t T_i$ and $V = \{\mathsf{Decode}(T, x_{1,j})_{j\in[n]}\}$.
  6) P_1 and P_2 invoke the two-party PSI-CA functionality $\mathcal{F}_{\mathsf{2\text{-}PSI\text{-}CA}}$ where P_2 acts as a sender with input Γ and P_1 acts as a receiver with input V.
  7) P_1 obtains and outputs $|V \cap \Gamma|$.

Fig. 4. Multi-party PSI-CA protocol

Our Multi-Party PSI-CA protocol can also be transformed easily into a Multi-Party PSI protocol. We only need to change the Step 6 that P_1 and P_2 invoke the two-party $\mathcal{F}_{\mathsf{PSI}}$ instead of $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$. P_1 can determine that $x_{1,j}$ is in the intersection set $\bigcap_{i=2}^t X_i$ iff. $v_j \in \Gamma$ according to our correctness proven below.

B. Security Proof

Theorem 1. *Assume that the two-party PSI-CA protocol and the zero-sharing protocol are secure in the semi-honest model, and the OKVS scheme is doubly oblivious. Then the protocol in Fig. 4 securely implements the functionality $\mathcal{F}_{\mathsf{PSI\text{-}CA}}$ for arbitrary t, in the presence of a semi-honest adversary who may corrupt any subset of $\{P_2, \ldots, P_t\}$ or a proper subset of $\{P_1, P_3, \ldots, P_t\}$.*

Proof. **Correctness.** We now show if $P_1, \ldots, P_t$ run the protocol honestly, P_1 will know the intersection set cardinality of t parties after the protocol with high probability. If item $x_{1,j} \in \bigcap_{i=2}^{t} X_i$, without loss of generality we assume that $x_{1,j} = x_{2,k}$. Thus, in Step 5, the decode result of P_1 will be $\mathsf{Decode}(T, x_{1,j}) = \mathsf{Decode}(\sum_{i=2}^{t} T_i, x_{1,j}) = \sum_{i=2}^{t} \mathsf{Decode}(T_i, x_{1,j}) = \gamma_k$ according to the homomorphic property of linear OKVS. Thus, $x_{1,j}$ will be counted as a member of the intersection set in Step 7. Otherwise, we claim that $v_j = \mathsf{Decode}(T, x_{1,j})$ be a random value in $\mathbb{F}$ since T_2 is oblivious. Then, if we set $|\mathbb{F}| \geq 2^{\lambda+2\log n}$, by union bound, the probability that any $\mathsf{Decode}(T, x_{1,j}) \in \Gamma$ for $x_{1,j} \notin \bigcap_{i=2}^{t} X_i$ is less than $\frac{n^2}{|\mathbb{F}|} \leq \frac{1}{2^{\lambda}}$. In this case, the item $x_{1,j} \notin \bigcap_{i=2}^{t} X_i$ will not be counted in Step 7 with high probability.

Privacy. We separate the proof into the maximal collusion case, from which security for non-maximal collusions can be derived. We exhibit simulators in two different cases and argue the indistinguishability of the produced transcripts from the real execution.

• Case 1: *$P_2, \ldots, P_t$ are passively corrupted.* In this case, the view of $P_2, \ldots, P_t$ consists only of the transcripts in two-party PSI-CA protocol. Thus, the simulator is the same as the one in the two-party PSI-CA protocol. Since the two-party PSI-CA protocol is secure in the semi-honest model, the joint view of the parties $P_2, \ldots, P_n$ is identically distributed in the simulation.

• Case 2: *P_1 is passively corrupted, and $P_3, \ldots, P_t$ are not all corrupted.* Without loss of generality, assume that $P_3, \ldots, P_{t-1}$ are passively corrupted, but P_t is not corrupted. In this case, the corrupt parties' view consists of the following:

1) The transcripts in Π_{ZS} (Step 1).
2) The masked OKVS $T_2' = T_2 + r_2$ and $T_t' = T_t + r_t$ (Step 3, 4).
3) The transcripts in $\mathcal{F}_{\mathsf{2\text{-}PSI\text{-}CA}}$ (Step6).

We construct simulator $\mathcal{S}$ as follows. $\mathcal{S}$ runs the protocol honestly to generate its view with the following exceptions: In step 1, $\mathcal{S}$ runs the zero-sharing simulator to simulate the view. In step 3, 4, $\mathcal{S}$ use random matrix to simulate T_2' and T_t'. In step 6, $\mathcal{S}$ runs the two-party PSI-CA simulator to simulate the view. We argue the indistinguishability via following hybrid:

- **Hyb**$_0$: The $\{P_1, P_3, \ldots, P_{t-1}\}$'s view in the real protocol.
- **Hyb**$_1$: Same as Hyb$_0$ except that in step 1, $\mathcal{S}$ runs the zero-sharing simulator to simulate the view. This hybrid is computationally indistinguishable from Hyb$_0$ by security of the zero-sharing protocol.
- **Hyb**$_2$: Same as Hyb$_0$ except that in step 3, $\mathcal{S}$ use random matrix to simulate T_2'. Since the OKVS T_2 is encoded by the random values Γ, according to the doubly oblivious property of OKVS, the matrix T_2 is pseudorandom. Thus, $T_2' = T_2 + r_2$ is computationally indistinguishable from the random matrix.
- **Hyb**$_3$: Same as Hyb$_0$ except that in step 4, $\mathcal{S}$ use random matrix to simulate T_t'. By the privacy of zero-sharing, the coalition of $\{P_1, P_3, \ldots, P_{t-1}\}$ will not reveal any information about the P_t's zero share s_t. Thus, the masked OKVS $T_t' = T_t + r_t$ is computationally indistinguishable from the random matrix. Furthermore, the joint distribution of (T_2', T_t') is indistinguishable to the uniform distribution on $\{0,1\}^{m\times(\lambda+2\log n)} \times \{0,1\}^{m\times(\lambda+2\log n)}$. Specifically, consider the conditional distribution. Given the value of $T_t^{'} = T_t + r_t$, consider the distribution of $T_2^{'}$. Since $T_2^{'} = T_2 + r_2$ which T_2 is pseudorandom and independent of $r_t + T_t$, the conditional distribution is uniform and matches the marginal distribution of $T_2^{'}$. Therefore, $T_2^{'}$ and $T_t^{'}$ are independent. Thus, this hybrid is computationally indistinguishable from Hyb$_2$.
- **Hyb**$_4$: Same as Hyb$_3$ except that in step 6, $\mathcal{S}$ runs the two-party PSI-CA simulator to simulate the view. This hybrid is computationally indistinguishable from Hyb$_3$ by security of the two-party PSI-CA protocol.

□

C. Complexity Analysis

In this section, we denote P_1 as the two-party PSI-CA receiver and P_2 as the sender in Step 6 of our protocol. The communication cost of our protocol mainly consists of the $m \times |\mathbb{F}|$ bits masked OKVS that $P_2, \ldots, P_t$ send to P_1 in steps 4 and 5, and the communication cost of the two-party PSI-CA protocol in step 6. We assume that there is a protocol $\Pi_{\mathsf{2\text{-}PSI\text{-}CA}}$ realizes the functionality $\mathcal{F}_{\mathsf{2\text{-}PSI\text{-}CA}}$ and we denote $Comm(\Pi_{\mathsf{2\text{-}PSI\text{-}CA}})$ as the communication cost of the protocol. We set $|\mathbb{F}| = 2^{\lambda+2\log n}$, so the total communication cost in our protocol is $(t-1)m(\lambda + 2\log n) + Comm(\Pi_{\mathsf{2\text{-}PSI\text{-}CA}})$ bits. The total computation cost of our protocol mostly comes from calculating n times zero-sharing function S, encoding $t-1$ OKVS, P_1 decoding an OKVS with n keys, and the cost of the two-party PSI-CA protocol $Comp(\Pi_{\mathsf{2\text{-}PSI\text{-}CA}})$ in step 6.

We claim that our protocol is much more efficient than the state-of-the-art multi-party PSI-CA protocol proposed by [8]. In their protocol, all parties must invoke $t-2$ times server-aided OPPRF, which will bring $(t-2)(m+2n)(\lambda + 2\log n)$ bits communication cost according to the server-aided OPPRF protocol in Appendix A. Their protocol must also decode for $n(t-2)$ times and calculate $4nt - 7n$ times PRF function. The zero-sharing process of their protocol is also much more complicated than ours. The comparison of the two protocols is shown in Table I. We also compare each party's communication and computation costs in Table II and Table III.

We instantiate our protocol using the OKVS scheme proposed by [25], which satisfies $m = 1.23n$. We utilize the server-aided two-party PSI-CA protocol proposed by [8], and party P_3 will act as the server in the two-party PSI-CA protocol. The total computation cost of our protocol is $((t-1)m + 3n)(\lambda + 2\log n)$ bits while all parties need to send $((t-2)(m+2n) + 3n)(\lambda + 2\log n)$ bits in total in the protocol proposed by [8]. Our computation cost is about 0.45 times that of theirs when $t = 16$. The computation cost of our protocol is also less than theirs.

TABLE I

TOTAL COMMUNICATION COST (IN BITS) OF ALL PARTIES AND COMPUTATION COST FOR OUR MULTI-PARTY PSI-CA PROTOCOL AND PROTOCOL [8]. WE ONLY ACCOUNTED FOR THE TOTAL AMOUNT OF DATA SENT BY EACH PARTY. WE DENOTED $Comm(\Pi_{\text{2-PSI-CA}})$, $Comp(\Pi_{\text{2-PSI-CA}})$ AS THE COMMUNICATION AND COMPUTATION COST OF THE TWO-PARTY PSI-CA PROTOCOL, RESPECTIVELY. WE ASSUME $|\mathbb{F}| = 2^{\lambda+2\log n}$.

Scheme	Total Communication Cost	Total Computation Cost
Our protocol	$(t-1)m(\lambda+2\log n) + Comm(\Pi_{\text{2-PSI-CA}})$	1 execution of Π_{ZS} $(t-1)$ times Encode n times Decode $Comp(\Pi_{\text{2-PSI-CA}})$
[8]	$(t-2)(m+2n)(\lambda+2\log n) + Comm(\Pi_{\text{2-PSI-CA}})$	1 execution of Π_{ZS} $(t-2)$ times Encode $n(t-2)$ times Decode $4nt-7n$ PRF $Comp(\Pi_{\text{2-PSI-CA}})$

TABLE II

COMMUNICATION COST (IN BITS) OF EVERY PARTY IN OUR MULTI-PARTY PSI-CA PROTOCOL AND PROTOCOL BY [8]. WE ACCOUNTED FOR THE AMOUNT OF DATA SENT AND RECEIVED FOR EACH PARTY. WE DENOTE P_1 AS THE TWO-PARTY PSI-CA RECEIVER AND P_2 AS THE SENDER, $Comm(\Pi_{\text{2-PSI-CA}})$ AS THE COMMUNICATION COST OF THE TWO-PARTY PSI-CA PROTOCOL. WE ASSUME $|\mathbb{F}| = 2^{\lambda+2\log n}$.

Scheme	P_1	P_2	P_i $(\forall\ i \in [3,t])$
Our protocol	$Comm(\Pi_{\text{2-PSI-CA}})+$ $(t-1)m(\lambda+2\log n)$	$m(\lambda+2\log n)+$ $Comm(\Pi_{\text{2-PSI-CA}})$	$m(\lambda+2\log n)$
[8]	$2(t-2)n(\lambda+2\log n)$ $+Comm(\Pi_{\text{2-PSI-CA}})$	$(t-2)(m+2n)$ $(\lambda+2\log n)+$ $Comm(\Pi_{\text{2-PSI-CA}})$	$m(\lambda+2\log n)$

TABLE III

COMPUTATION COST FOR OUR MULTI-PARTY PSI-CA PROTOCOL AND PROTOCOL [8]. WE DENOTE P_1 AS THE TWO-PARTY PSI-CA RECEIVER AND P_2 AS THE SENDER, $S(K_i,\cdot)$ AS THE ZERO-SHARING FUNCTION, $Comp_r(\Pi_{\text{2-PSI-CA}})$, $Comp_s(\Pi_{\text{2-PSI-CA}})$ AS THE COMPUTATION COST OF THE RECEIVER AND THE SENDER, RESPECTIVELY.

Scheme	P_1	P_2	P_i $(\forall\ i \in [3,t])$
Our protocol	n times Decode $Comp_r(\Pi_{\text{2-PSI-CA}})$	1 calculation of $S(K_i,\cdot)$ 1 times Encode $Comp_s(\Pi_{\text{2-PSI-CA}})$	1 calculation of $S(K_i,\cdot)$ 1 times Encode
[8]	$(t-1)n$ PRF $Comp_r(\Pi_{\text{2-PSI-CA}})$	$(t-2)n$ times Decode $(t-2)n$ PRF n calculation of $S(K_i,\cdot)$ $Comp_s(\Pi_{\text{2-PSI-CA}})$	n calculation of $S(K_i,\cdot)$ 1 times Encode $2n$ PRF

IV. IMPLEMENTATION AND PERFORMANCE COMPARISON

We implement our protocol in C++ using libOTe [26]. Our implementation is available on GitHub: https://github.com/lzjluzijie/mpsi. Our implementation uses the OKVS code from [25], where Encode and Decode are based on the PaXoS data structure. The expansion parameter of the OKVS code we used in our protocol is 1.23, namely $m = 1.23n$.

We use the zero sharing protocol in Section II-C with AES as PRF. All evaluations were performed with each item input length of 128 bits, the statistical security parameter is $\lambda = 40$, and the computational security parameter is $\kappa = 128$. Our implementation uses the server-aided two-party PSI-CA protocol from [8], where P_3 acts as the server.

A. Benchmark

The experiments were run on a desktop computer with 16 cores AMD 3950X CPU and 32GiB RAM. We considered the localhost environment and simulated WAN network settings using the Linux tc command, with 200 Mbps bandwidth and 96 ms round-trip latency. The implementation is single-threaded, so we expect an approximately linear speedup for computation when using multithreading.

In the same environment, we run the state-of-the-art multiparty PSI-CA code provided in [8]. Note that the printed running time of their program did not include the time of zero sharing, so we measured the duration directly from the beginning of execution to termination. Although our experiment of their program didn't match the data in [8], the data in Table. IV confirms our previous analysis in Section III-C.

To understand the efficiency of our protocol, we evaluate it on the range of the number of parties $t \in \{3, 4, 8, 16\}$ on the set size $n \in \{2^{12}, 2^{16}, 2^{20}\}$. Due to the different roles of parties, our protocol is asymmetric with respect to the receiver P_1, sender P_2, and the server P_3 of the server-aided two-party PSI-CA as well as other parties P_i for $i \in [4, t]$. Thus, we report the performance results of these parties separately. In our protocol, the workload of the receiver is to compute n times Decode of OKVS and n times PRF evaluations (i.e., AES instances). The majority of the receiver's running time is to wait for other parties to finish their work. For example, P_1 takes 5.84 seconds to compute PSI-CA with $t = 16$ and $n = 2^{20}$ in the localhost setting. Furthermore, as P_1 receives encoded OKVS table from all other parties (i.e., $P_2, \ldots, P_t$), P_1's communication cost is highest amongst other participants. For $t = 16$ and $n = 2^{20}$, the protocol PSI-CA requires 326.6 MiB communication cost in total, where there are 310 MiB on P_1's side. Our protocol's communication and computational overhead are better than the state-of-the-art protocol proposed by [8]. The detailed experiment results are shown below.

Computation Improvement. Table. IV presents the running time of protocols in both LAN and WAN settings. We separately report the running time of P_1, P_2, P_3, and other parties P_i for $i \in [4, t]$. In the Localhost setting, where the running time is dominated by computation, our protocol requires less computation cost than Gao et al.' s [8]. For example, to

TABLE IV
RUNNING TIME (IN SECONDS).

n	t	Protocol	Running Time											
			localhost				200 mbps				20 mbps			
			P_1	P_2	P_3	$P_{i\in[4,t]}$	P_1	P_2	P_3	$P_{i\in[4,t]}$	P_1	P_2	P_3	$P_{i\in[4,t]}$
2^{12}	3	Ours	0.01	0.01	0.01	-	0.68	0.20	0.53	-	0.74	0.21	0.57	-
		[8]	0.07	0.07	0.07	-	1.13	1.22	1.22	-	1.18	1.28	1.28	-
	4	Ours	0.01	0.01	0.01	0.01	0.68	0.20	0.53	0.14	0.75	0.22	0.58	0.14
		[8]	0.07	0.07	0.07	0.07	1.56	1.66	1.75	1.70	1.65	1.74	1.84	1.79
	8	Ours	0.02	0.01	0.01	0.01	0.69	0.20	0.54	0.14	0.84	0.22	0.67	0.15
		[8]	0.08	0.08	0.08	0.08	2.46	2.56	2.65	2.75	2.80	2.89	2.97	3.08
	16	Ours	0.04	0.04	0.04	0.04	0.70	0.20	0.55	0.14	1.26	0.22	1.09	0.15
		[8]	0.10	0.10	0.10	0.10	4.07	4.17	4.26	4.36	5.17	5.26	5.36	5.46
2^{16}	3	Ours	0.07	0.04	0.04	-	1.46	0.23	0.90	-	2.93	0.24	2.07	-
		[8]	0.17	0.16	0.16	-	3.06	3.16	3.16	-	4.86	4.96	4.95	-
	4	Ours	0.07	0.04	0.04	0.04	1.50	0.33	0.93	0.18	3.44	0.26	2.58	0.18
		[8]	0.19	0.19	0.19	0.19	3.38	3.48	3.57	3.52	6.47	6.56	6.66	6.61
	8	Ours	0.08	0.05	0.06	0.05	1.67	0.33	1.11	0.20	5.61	0.30	4.75	0.19
		[8]	0.21	0.21	0.21	0.21	4.58	4.68	4.77	4.87	13.05	13.15	13.25	13.34
	16	Ours	0.13	0.09	0.11	0.08	2.05	0.33	1.49	0.20	9.91	0.61	9.05	0.20
		[8]	0.25	0.25	0.25	0.25	7.02	7.12	7.21	7.31	26.01	26.11	26.20	26.30
2^{20}	3	Ours	2.90	1.52	2.09	-	7.99	5.63	7.10	-	34.19	25.49	32.65	-
		[8]	5.07	5.07	5.07	-	10.74	10.83	10.83	-	47.68	47.77	47.77	-
	4	Ours	3.20	1.74	2.36	1.74	8.90	6.52	7.99	4.86	42.59	33.82	40.94	25.97
		[8]	5.10	5.09	5.09	5.09	13.62	13.70	13.80	13.75	70.54	70.63	70.73	70.68
	8	Ours	3.83	2.41	3.02	2.21	12.27	9.88	11.35	7.61	76.10	67.22	74.44	52.74
		[8]	5.55	5.54	5.54	5.54	25.65	25.74	25.84	25.93	161.22	161.32	161.41	161.51
	16	Ours	5.84	4.00	4.96	3.67	20.37	17.96	19.44	14.70	144.06	135.25	142.40	123.14
		[8]	6.71	6.71	6.70	6.70	50.71	50.80	50.90	50.99	343.46	343.55	343.64	343.74

compute PSI-CA with $t = 16$ parties and set size $n = 2^{20}$, our protocol runs in 5.84 seconds, which is $1.15\times$ faster than Gao et. al.'s [8] that runs in 6.71 seconds.

In the WAN setting, as shown in the Table. IV, in moderate bandwidth (i.e., 20 $\sim$ 200 Mbps), our protocol takes much less time than that in the localhost setting because we have lower communication overhead than Gao et al.'s [8]. For example, for $t = 16$ parties and set size $n = 2^{20}$, in the 20 Mbps network, our protocol runs in 144.06 seconds, which is $2.38\times$ faster than Gao et al.' s [8] that runs in 343.46 seconds.

Communication Improvement. Table. V shows the communication overhead of the protocols, measured in one-way cost. The total communication cost of our protocol is about $\frac{m+2n}{m}\times$ smaller than that of Gao et al.' s [8]. For example, to compute PSI-CA with $t = 16$ parties and set size $n = 2^{20}$, our protocol requires 326.6 MiB communication, which is a $2.38\times$ improvement of Gao et. al.'s [8] that requires 780.48 MiB communication.

V. CONCLUSIONS

In this paper, we propose an efficient multi-party PSI-CA protocol only from OKVS and zero-sharing techniques. We compare the communication and computation with state-of-the-art multi-party PSI-CA protocol [8] theoretically and experimentally. We measured our protocol on a WiFi connection of 200 Mbps bandwidth. It only needs 20.37 seconds and 310.6 MiB communication when computing the intersection between 16 parties with 2^{20} private items each. In this case, our protocol reduces the total communication by a factor up to $2.4\times$ compared to that of [8]. We also demonstrate that our protocol has higher computational efficiency than their protocol from theoretical analysis. However, our protocol still has a limitation. Our protocol requires two specific non-colluding participants instead of the setting of non-colluding participants. Thus, future research will focus on constructing PSI-CA protocols under stronger security settings. We also observe the potential adaptation of other OKVS schemes to our protocol. Our future work will also focus on the specific implementation.

ACKNOWLEDGMENT

This work is supported by National Key R&D Program of China (No. 2021YFB2701304).

REFERENCES

[1] S. Dittmer, Y. Ishai, S. Lu, R. Ostrovsky, M. Elsabagh, N. Kiourtis, B. Schulte, and A. Stavrou, "Function secret sharing for psi-ca: With ap-

TABLE V
COMMUNICATION COST (IN MIB).

n	t	Ours					[8]				
		P_1	P_2	P_3	$P_{i\in[4,t]}$	Total	P_1	P_2	P_3	$P_{i\in[4,t]}$	Total
2^{12}	3	0.23	0.15	0.21	-	0.30	0.31	0.27	0.21	-	0.40
	4	0.31	0.15	0.21	0.08	0.38	0.44	0.47	0.21	0.08	0.60
	8	0.64	0.15	0.21	0.08	0.70	0.94	1.29	0.21	0.08	1.42
	16	1.31	0.15	0.21	0.08	1.36	1.94	2.93	0.21	0.08	3.06
2^{16}	3	3.49	2.24	3.24	-	4.49	5.00	4.27	3.27	-	6.27
	4	4.73	2.24	3.24	1.24	5.73	7.00	7.54	3.27	1.27	9.54
	8	9.71	2.24	3.24	1.24	10.70	15.00	20.62	3.27	1.27	22.62
	16	19.67	2.24	3.24	1.24	20.64	31.00	46.78	3.27	1.27	47.78
2^{20}	3	55.28	35.64	51.64	-	71.28	80.00	68.32	52.32	-	100.32
	4	74.92	35.64	51.64	19.64	90.92	112.00	120.64	52.32	20.32	152.64
	8	153.48	35.64	51.64	19.64	169.48	240.00	329.92	52.32	20.32	361.92
	16	310.60	35.64	51.64	19.64	326.60	496.00	748.48	52.32	20.32	780.48

plications to private contact tracing," *arXiv preprint arXiv:2012.13053*, 2020.

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

[3] K. Michael and R. Abbas, "Behind covid-19 contact trace apps: The google–apple partnership," *IEEE Consumer electronics magazine*, vol. 9, no. 5, pp. 71–76, 2020.

[4] J. Gao, C. Surana, and N. Trieu, "Secure contact tracing platform from simplest private set intersection cardinality," *IET information security*, vol. 16, no. 5, pp. 346–361, 2022.

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

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

[7] K. Chida, K. Hamada, A. Ichikawa, M. Kii, and J. Tomida, "Communication-efficient inner product private join and compute with cardinality," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023, pp. 678–688.

[8] J. Gao, N. Trieu, and A. Yanai, "Multiparty private set intersection cardinality and its applications," *Proceedings on Privacy Enhancing Technologies*, 2024.

[9] M. Wu and T. H. Yuen, "Efficient unbalanced private set intersection cardinality and user-friendly privacy-preserving contact tracing," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 283–300.

[10] L. Kissner and D. Song, "Privacy-preserving set operations," in *Annual International Cryptology Conference*. Springer, 2005, pp. 241–257.

[11] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty psi and extensions to circuit/quorum psi," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1182–1204.

[12] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 245–254.

[13] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *Journal of Computer Security*, vol. 13, no. 4, pp. 593–622, 2005.

[14] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[15] P. Mohassel, P. Rindal, and M. Rosulek, "Fast database joins and psi for secret shared data," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1271–1287.

[16] P. Mohassel and S. Sadeghian, "How to hide circuits in mpc an efficient framework for private function evaluation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 557–574.

[17] E. Fenske, A. Mani, A. Johnson, and M. Sherr, "Accountable private set cardinality for distributed measurement," *ACM Transactions on Privacy and Security*, vol. 25, no. 4, pp. 1–35, 2022.

[18] A. A. Jolfaei, H. Mala, and M. Zarezadeh, "Eo-psi-ca: Efficient outsourced private set intersection cardinality," *Journal of Information Security and Applications*, vol. 65, p. 102996, 2022.

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[21] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *CRYPTO 2021, Proceedings, Part II 41*. Springer, 2021, pp. 395–425.

[22] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of ACM CCS 2017*, 2017, pp. 1257–1272.

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

[24] A. Bienstock, S. Patel, J. Y. Seo, and K. Yeo, "{Near-Optimal} oblivious {Key-Value} stores for efficient {PSI},{PSU} and {Volume-Hiding}{Multi-Maps}," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 301–318.

[25] S. Raghuraman and P. Rindal, "Blazing fast psi from improved okvs and subfield vole," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2505–2517.

[26] L. R. Peter Rindal, "libOTe: an efficient, portable, and easy to use Oblivious Transfer Library," https://github.com/osu-crypto/libOTe.

APPENDIX

A. Server-Aided OPPRF protocol from [8]

A server-aided OPPRF protocol runs between a sender, a receiver, and a cloud server. The sender and receiver have a private set $X = \{(x_1, v_1), \ldots, (x_{m_1}, v_{m_1})\}$ and $Y = \{y_1, \ldots, y_{m_2}\}$, respectively. For $y_i \in Y$, the receiver will receive OPPRF output v_j if $y_i = x_j$ after the protocol. Otherwise, it will receive a random value as the OPPRF output. The detailed protocol is shown in Protocol 4.

B. Multi-party PSI-CA from [8]

The original protocol proposed by [8] involved n parties, where P_1, P_n, and P_2 act as the receiver, sender, and server of $\Pi_{\mathsf{SA\text{-}2\text{-}PSI\text{-}CA}}$, respectively. We change the numbering of the

PROTOCOL 4. (Server-Aided OPPRF Π_{sopprf} [8])

- **Parameters:**
  - The protocol runs between a sender $\mathcal{S}$, a receiver $\mathcal{R}$ and a server $\mathcal{C}$. $\mathcal{S}$ and $\mathcal{R}$ have input size of m_1 and m_2. A PRF $F : \{0,1\}^\kappa \times \{0,1\}^l \rightarrow \{0,1\}^l$.
- **Inputs:**
  - Sender $\mathcal{S}$ has input $X = \{(x_1, v_1), \ldots, (x_{m_1}, v_{m_1})\}$ with (pseudo) random v_i's
  - Receiver $\mathcal{R}$ has input $Y = \{y_1, \ldots, y_{m_2}\}$.
  - Cloud $\mathcal{C}$ has no input.
- **Protocol:**

1) $\mathcal{S}$ chooses random keys $(k_1, k_2) \in \{0,1\}^\kappa$ and send k_1, k_2 to $\mathcal{R}$, $\mathcal{C}$, respectively.
2) $\mathcal{R}$ computes $Y' = F(k_1, Y)$ and sends Y' to $\mathcal{C}$.
3) $\mathcal{C}$ computes $Y'' = F(k_2, Y') = \{y''_1, \ldots, y''_{m_2}\}$ and sends Y'' to $\mathcal{R}$.
4) $\mathcal{S}$ constructs an OKVS $T \leftarrow \mathsf{Encode}(\{x_i, F(k_2, F(k_1, x_i)) \bigoplus v_i\}_{i\in[m_1]})$ and sends T to $\mathcal{R}$.
5) For every $j \in [m_2]$, $\mathcal{R}$ outputs $v'_j = y''_j \bigoplus \mathsf{Decode}(T, y_j)$.

Fig. 5. Server-Aided OPPRF protocol from [8]

parties in their protocol, adapting to our notation such that P_1, P_2, and P_3 act as the receiver, sender, and server of $\Pi_{\mathsf{SA\text{-}2\text{-}PSI\text{-}CA}}$. The detailed protocol is shown in Protocol 2.

PROTOCOL 5. (Multi-party PSI-CA [8])

- **Parameters:**
  - The protocol runs between parties $P_1, \ldots, P_t$ for $t > 2$. A PRF $F : \{0,1\}^\kappa \times \{0,1\}^l \rightarrow \{0,1\}^l$.
- **Inputs:** P_i has $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$.
- **Protocol:**

1) Parties $P_2, \ldots, P_t$ invoke Π_{ZS} and each party P_i obtains the key K_i for a sharing function S.
2) Parties P_1 and P_3 agree on a random PRF key s.
3) Parties $P_3, \ldots, P_t$ agree on a random PRF key k.
4) Party P_i for $i \in [3, t]$ computes the set of points $\mathcal{P}_i$ where:
   - $\mathcal{P}_3 = \{(F(k, x_{3,j}), S(K_3, x_{3,j}) \bigoplus F(s, x_{3,j}))\}_{j\in[n]}$.
   - For $i \in [4, t]$ $\mathcal{P}_i = \{(F(k, x_{i,j}), S(K_i, x_{i,j}) \bigoplus F(s, x_{i,j}))\}_{j\in[n]}$.
5) P_2 and P_i (for every $i \in [3, t]$) invoke an instance of the server-aided OPPRF Π_{sopprf} where:
   - P_i acts as a sender with input $\mathbf{P}_i$.
   - P_1 acts as a cloud server with no input.
   - P_2 acts as a receiver with input X_2. P_2 obtains the result $y_{i,j}$ on the query $x_{2,j}$.
6) For every $j \in [n]$, P_2 computes $w_j = \bigoplus_{i=3}^{t} y_{i,j} \bigoplus S(K_2, x_{2,j})$. Then, P_2 sets W to be $\{w_1, \ldots, w_n\}$.
7) P_1 and P_2 invoke the server-aided $\Pi_{\mathsf{SA\text{-}2\text{-}PSI\text{-}CA}}$ protocol with P_3 as a server, where:
   - P_2 acts as a sender with input W.
   - P_3 acts as a cloud server with no input.
   - P_1 acts as a receiver with input $V = F(s, X_1)$, and obtains $|W \cap V|$.

Fig. 6. Multi-party PSI-CA protocol from [8]