



## Article A Novel Edge Cache-Based Private Set Intersection Protocol via Lightweight Oblivious PRF

Jing Zhang, Li Yang, Yongli Tang \*, Minglu Jin 🕑 and Shujing Wang

College of Software, Henan Polytechnic University, Jiaozuo 454000, China

\* Correspondence: yltang@hpu.edu.cn

**Abstract:** With the rapid development of edge computing and the Internet of Things, the problem of information resource sharing can be effectively solved through multi-party collaboration, but the risk of data leakage is also increasing. To address the above issues, we propose an efficient multi-party private set intersection (MPSI) protocol via a multi-point oblivious pseudorandom function (OPRF). Then, we apply it to work on a specific commercial application: edge caching. The proposed MPSI uses oblivious transfer (OT) together with a probe-and-XOR of strings (PaXoS) as the main building blocks. It not only provides one-sided malicious security, but also achieves a better balance between communication and computational overhead. From the communication pattern perspective, the client only needs to perform OT with the leader and send a data structure PaXoS to the designated party, making the protocol extremely efficient. Moreover, in the setting of edge caching, many parties hold a set of items containing an identity and its associated value. All parties can identify a set of the most frequently accessed common items without revealing the underlying data.

Keywords: private set intersection; edge computing; multi-party cooperative cache; concrete efficiency

# check for **updates**

Citation: Zhang, J.; Yang, L.; Tang, Y.; Jin, M.; Wang, S. A Novel Edge Cache-Based Private Set Intersection Protocol via Lightweight Oblivious PRF. *Entropy* **2023**, *25*, 1347. https:// doi.org/10.3390/e25091347

Academic Editors: Eirik Rosnes and Hsuan-Yin Lin

Received: 12 June 2023 Revised: 1 September 2023 Accepted: 12 September 2023 Published: 16 September 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).

## 1. Introduction

Co-creation and sharing gained significance in the transition from the era of information technology to the era of digital technology. While information sharing brings convenience, the risk of privacy breaches also rises. The private set intersection (PSI) protocol is a widely used approach to distributed set computation. It is devoted to the joint intersection calculation of data from two or more parties. The PSI protocol guarantees that all parties can collaboratively calculate the intersection of the sets without disclosing anything beyond that intersection. PSI plays an important role in improving pattern matching [1], private contact discovery [2], advertisement conversion rate [3], and edge caching [4]. Edge caching is a key technology for communication networks. In order to utilize cache resources more efficiently, individual operators tend to keep their public items in a shared cache that can be accessed by all parties. However, since the cache is shared among multiple parties, these parties aim to identify the set of most frequently visited common data items and add them to the network edge cache. Their objective is to achieve this without revealing the actual underlying data. This is known as the multi-party shared cache problem, where determining the common term is a typical private set intersection problem.

Most of the current efficient PSI protocols are built on OT [5–7]. The OT-based PSI protocols offer greater advantages in terms of communication and computation when compared with PSI based on public key encryption [8,9] and PSI based on a garbled circuit [10–12]. Efficient OT extension techniques allow parties to generate many OT protocol instances at a low computational cost through a few public key operations. Chase et al. [5] implemented a two-party PSI protocol with one-sided malicious security. This protocol uses OT and a multi-point OPRF to achieve a good balance between computational and communication overhead. The protocol can only interact between two parties, and multiple runs are required to accomplish the intersection computation with multiple parties

involved. Kavousi et al. [13] proposed a MPSI based on OT and multi-point OPRF. This protocol can only be implemented in the semi-honest model. Inbar et al. [14] presented an enhanced semi-honest MPSI protocol based on OT and a garbled Bloom filter (GBF). However, the protocols [13,14] require the transmission of the GBF for communication, creating a certain degree of communication burden.

In response to the above issues, we constructed an MPSI protocol for malicious actors that combines a PaXoS and multi-point OPRF based on OT. The protocol relies only on symmetric keys, hashing, coding techniques, and bitwise operations, thus providing good computational performance. This protocol can solve the problem of privacy-preserving edge cooperative cache sharing by making a simple transformation of this protocol. We show the following contributions:

- Multi-party PSI protocol: We propose a specifically efficient MPSI protocol utilizing OT and a PaXoS. The PaXoS can be seen as a corresponding Encode/Decode algorithm achieving a constant rate. Therefore, our protocol has good computational performance. The protocol has low communication overhead since the clients only need to send a data structure. Theoretical analysis shows that the protocol leads to a better balance between communication and computational cost.
- Security against malicious clients: We present that our protocol uses the data structure PaXoS to hide the key during encoding to resist malicious adversaries, which can achieve one-sided malicious security against the clients with almost no additional overhead. At the same time, we prove that the protocol can also resist any possible collusion attack from malicious clients.
- Multi-party cooperative cache: Our MPSI protocol can be applied to edge caching scenarios by using cuckoo hashing and simple hashing. The protocol supports having data associated with each input and the extension of payloads to multi-party. In a multi-party cooperative cache (MPCCache) setting, the MPCCache protocol allows parties to compute a sum depending on the data associated with the intersection items. Compared with [4], our MPCCache protocol eliminates the computing burden associated with polynomial interpolation and improves computational efficiency.

## 2. Related Work

PSI. The development of efficient constructions for PSI functionality has received considerable research attention in the last decade or more. Some of the recent relevant works on PSI are illustrated in Table 1. Ghosh et al. [15] presented a MPSI protocol using oblivious linear function evaluation (OLE) with optimal asymptotic communication complexity. However, the balance between communication and computation cost is not good. Kolesniko [16] proposed a two-party PSI protocol against semi-honest adversaries. The protocol is mainly based on OT techniques for security string equivalence testing and is computationally efficient. Pinkas [17] proposed a two-party semi-honest PSI protocol based on OT and a GBF. The parallelized processing of the protocol allows for some improvement in protocol efficiency. Nevo [18] proposed a malicious PSI protocol utilizing oblivious programmable PRF (OPPRF) and oblivious key-value store (OKVS) technology, which solves the problem of multi-party PSI against malicious adversaries. However, this protocol does not lead to a better trade-off between communication and computational overhead. Pinkas [19] also proposed a PSI protocol for two parties in the malicious model which uses a PaXoS to implement, for the first time, a malicious secure PSI using cuckoo hashing. Ben-Efraim et al. [20] implemented malicious MPSI based on a GBF for multiple parties. However, GBFs suffer from a certain false positive rate and their high communication overhead. Bui et al. [21] constructed an optimized semi-honest PSI based on a pseudorandom correlation generator (PCG). Additionally, they can use the PCG to construct protocols with fully malicious security in the standard model.

Protocol	Technical	Number of Parties	Security Model
[15]	OLE	multi-party	Malicious
[16]	OT	two-party	Semi-Honest
[17]	GBF + OT	two-party	Semi-Honest
[18]	OPPRF + OKVS	multi-party	Malicious
[19]	PaXoS	two-party	Malicious
[20]	GBF	multi-party	Malicious
[21]	PCG	two-party	Semi-Honest

Table 1. The related work of PSI.

Function-based PSI. Many studies have focused on developing efficient techniques for PSI construction. In addition, these studies have explored the output results of computing a function over intersections, allowing for potential extensions to various business scenarios. Table 2 shows recent related works on function-based PSI. Ion et al. [3] proposed a PI-Sum Protocol utilizing Diffie-Hellman (DDH) and homomorphic encryption (HE). Thinking about the advertising (Ad) conversion problem: Ad providers want to analyze Ad effectiveness by age, which obviously cannot be solved using the PI-Sum. Chida [22] proposed a new function based on OPRF and DDH assumptions to calculate the weighted sum of two-party privacy sets (PIW-sum), which has more practical application value. Pinkas et al. [11] proposed an idea of calculating payloads based on the circuit, OPPRF, and cuckoo hash constructs, which allows each input item from one party to have payload data attached to it, and finally to calculate some specific functions of the payloads in the intersection set. Based on a new shuffled distributed oblivious PRF (DOPRF), Miao et al. [23] constructed a two-party PSI cardinality (PSI-CA) protocol for malicious settings which achieves a good computation and communication cost. In the above protocols, only one party can own the payload data, which can be applied in limited practical scenarios. Nguyen et al. [4] extended payload data to the multi-party setting and proposed an MPCCache sharing framework based on polynomial interpolation and OPPRF, which enables multiple parties to calculate a sum of data payloads on each of common data items and can identify the most frequently accessed data items.

Protocol	Technical	Number of Parties	Protocol Type
[3]	DDH + HE	two-party	PI-Sum
[4]	OPPRF	multi-party	MPCCache
[11]	OPPRF + Circuit	multi-party	PSI- payload
[22]	OPRF + DDH	two-party	PIW-Sum
[23]	DOPRF	two-party	PSI-CA

Table 2. The related work of function-based PSI.

#### 3. Preliminaries

3.1. Notions

The computational and statistical security parameters are denoted by  $\lambda$  and  $\sigma$ . [n] stands in for the set  $\{1, \ldots, n\}$ .  $\stackrel{R}{\leftarrow}$  indicates uniformly random selection. The notation || denotes concatenation between strings.  $\{0, 1\}^*$  denotes the set of strings consisting of 0 and 1, where \* means that the strings in the set can be of any length. We use  $\stackrel{C}{\approx}$  to indicate that the real world is indistinguishable from the ideal world. We denote with v[i] the *i*-th element of a vector  $\mathbf{v}$  of length *l*. The *i*-th column vector  $i \in [n]$  of the matrix  $M_{n \times m}$  is denoted by the symbol  $M_i$ . The Hamming weight of the binary string *x* is represented by  $||x||_{\mathrm{H}}$ 

#### 3.2. One-Sided Malicious Security

One-sided malicious security [5] is a security property found in cryptographic protocols wherein one party is allowed to engage in arbitrary malicious behavior in an attempt to compromise security while the other parties follow specified behavioral guidelines. In this context, only the targeted party is vulnerable to malicious action, whereas the other parties maintain their assigned roles and responsibilities. Our MPSI protocol achieves unilateral malicious security against the clients, as they are considered as a whole. We further prove that the proposed MPSI is secure against malicious clients.

#### 3.3. Security Model

MPSI is a unique instance of secure multi-party computation (MPC). We adhere to the MPC standard security definition. The ideal functionality of MPSI is defined in Figure 1.

**Inputs:** Party  $P_{i \in [n]}$  respectively input a set of elements  $X_i = \{x_1, ..., x_{n_i}\}$  where  $x_i \in \{0, 1\}^*$ . **Output:** Party  $P_n$  receives the set intersection  $I = X_1 \cap ... \cap X_n$ .

**Figure 1.** Ideal functionality of MPSI  $\mathcal{F}_{MPSI}$ .

The security models [24] of secure multi-party computation are divided into semihonest and malicious models. For the semi-honest model, an adversary can completely obey the protocol execution process, yet might record all the data in the protocol execution process and try to learn more from the data generated during the protocol execution process. The adversary under the malicious model can not only infer the sensitive information through the data of the protocol process but also refuse to participate in the protocol, alter the private input set information, or prematurely stop the protocol from running. Our protocol can achieve one-sided malicious security.

**Definition 1.** (Malicious security against the clients) If there is a PPT adversary A who might unilaterally depart from the protocol in the real world, there exists a PPT adversary S who could modify the input to the ideal functionality and terminate the output in an ideal world. Then, the protocol  $\Pi$  can protect from malicious clients, such that for each input  $X_1, \ldots, X_n$ :

$$\operatorname{Real}_{A}^{\prod}(X_{1},\ldots,X_{n}) \stackrel{c}{\approx} \operatorname{Ideal}_{S}^{\mathcal{F}}(X_{1},\ldots,X_{n}).$$
(1)

#### 3.4. Oblivious Transfer

Rabin et al. [25] proposed a crucial cryptographic primitive OT. In a 1-out-of-2 OT configuration, the receiver can have a choice bit  $b \in \{0, 1\}$ , while the sender can have input strings  $(m_0, m_1)$ . The OT acts to prevent the receiver from knowing nothing regarding  $m_{1-b}$  and prevent the sender from learning anything about b. OT necessitates costly public-key operations. Ishai et al. [26] described an OT extension technique that permits many OT executions at the cost of doing few public-key procedures. We can use the instantiation OT in [15]. The ideal functionality of OT is defined in Figure 2.

**Input:** The sender inputs two strings  $(m_0, m_1)$  and the receiver inputs a choice bit  $b \in \{0, 1\}$ . **Output:** The functionality returns only  $m_b$  to the receiver and returns nothing to the sender.

**Figure 2.** Ideal functionality of OT  $\mathcal{F}_{OT}$ .

#### 3.5. PaXoS

The following is a way to encode key-value mapping into a brief data structure using a PaXoS [19]. The associated Encode/Decode methods are frequently more convenient to describe when describing a PaXoS than the u mapping.

Encode( $(x_1, y_1), \ldots, (x_t, y_t)$ ): Given *t* items  $(x_i, y_i)$ , where  $x_i \in \{0, 1\}^*$  and  $y_i \in \{0, 1\}^w$ , indicate via *M* the *t* × *m* matrix where the *i*-th row is  $u(x_i)$ . Note that u(x) is the result of using the mapping u to *x*. It is possible to find a data structure (matrix)  $D = (d_1, \ldots, d_m)^T \in (\{0, 1\}^w)^m$ 

satisfying  $M \times D = (y_1, \dots, y_t)^T$ . In particular, the subsequent linear system of equations is fulfilled when the  $u(x_i)$ 's are linearly independent:

$$\begin{bmatrix} -\mathbf{u}(x_1) - \\ -\mathbf{u}(x_2) - \\ \vdots \\ -\mathbf{u}(x_t) - \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{bmatrix}.$$
 (2)

Decode(D, x): Given  $D \in (\{0, 1\}^w)^m$  and  $x \in \{0, 1\}^*$ , we can extract the corresponding "value" via  $y = \langle u(x), D \rangle \stackrel{\text{def}}{=} \bigoplus_{j:v(x)_i=1} d_j$ .

#### 3.6. Multi-Point OPRF

Chase [5] presented a PSI protocol for two parties based on multi-point OPRF. The sender chooses a pseudorandom seed  $s \stackrel{R}{\leftarrow} \{0,1\}^w$ , and the receiver computes a pseudorandom function  $v = F_k(x_i)$  based on its set elements to construct two matrices:  $A_{m \times w}$  and  $B_{m \times w}$ . For each  $x_i \in X_1$ , the corresponding bits in matrices are the same, while others are different. The sender obtains a matrix  $C_{m \times w}$  depending on seed s and runs w OTs with the receiver. Each column of the matrix is either  $A_j$  or  $B_j$  for all  $j \in [w]$ . Then, the sender computes  $v = F_k(x_i)$  according to each element  $x_i \in X_2$  to obtain all the resulting OPRFs  $\varphi = H(C_1[v[1]||...||C_w[v[w]])$  and sends them to the receiver. Eventually, the receiver can find the intersection of the two sets based on its computed OPRF value.

#### 3.7. Hamming Correlation Robustness

Under the assumption of correlation robustness for the underlying hash function, our MPSI structure is demonstrated to be secure.

**Definition 2.** (Hamming Correlation Robustness [5]) If the distribution produced by the sampling of  $s \leftarrow \{0,1\}^n$  at random is pseudorandom for  $a_1, \ldots, a_m, b_1, \ldots, b_m \in \{0,1\}^n$ , and has  $||b_i||_{H} \ge d$  for each  $i \in [m]$ , H is d-Hamming correlation robust. Namely:

$$(H(a_1 \oplus [b_1 \cdot s]), \dots, H(a_m \oplus [b_m \cdot s])) \stackrel{\sim}{\approx} (F(a_1 \oplus [b_1 \cdot s]), \dots, F(a_m \oplus [b_m \cdot s])),$$
(3)

where  $\oplus$  denotes bitwise-AND and bitwise-XOR, respectively, and F is a random function.

#### 3.8. Cuckoo Hashing and Simple Hashing

Hash technology is one of the essential tools for optimizing communication and computational complexity in PSI protocols. There are two commonly used construction methods for hash technology: simple hashing and cuckoo hashing [10]. Simple hashing can map elements to k positions in a hash table using k hash functions, with each bucket being capable of storing multiple elements. Cuckoo hashing can map elements to a specific location in a hash table using a hash function, and its basic idea is to use multiple hash functions to handle collisions. When collisions occur, cuckoo hashing evicts the element occupying the original position, which can be rehomed to alternative positions. If alternative positions are already occupied, the process repeats until all elements can find their homes. Typically, cuckoo hashing and simple hashing are combined to achieve optimal results in PSI protocols.

#### 4. Our MPSI Protocol

### 4.1. Overview

In this section, we show the MPSI protocol. A couple of parties  $P_1, \ldots, P_n$  with respective private input sets  $X_1, \ldots, X_n$  desire to collectively compute  $X_1 \cap \ldots \cap X_n$  without disclosing any more information. Note that we regard *t* as the set sizes for parties,  $P_n$  as the leader, and  $P_{i \in [n-1]}$  as the client. The system model of the MPSI protocol is shown in Figure 3.



Figure 3. System model.

 $P_n$  constructs a random matrix  $A_{m \times w}$  and chooses strings for  $P_{i \in [n-1]}$  to generate the  $A_j^i \stackrel{R}{\leftarrow} \{0,1\}^m$  and sets  $A_j^n = A_j^1 \oplus \ldots \oplus A_j^{n-1}$ , where  $j \in [w]$ . For each  $i \in [n-1]$ , from its input elements,  $P_n$  constructs unique matrices  $B_{m \times w}$ .  $P_n$  first initializes a matrix  $E_{m \times w}$  to all 1's. For  $x \in X_n$  computing  $v = F_k(H_1(x))$ ,  $B_{m \times w}$  is designed such that  $E_j[v[j]] = 0$  for all  $j \in [w]$ , and hence  $A_j^i[v[j]] = B_j^i[v[j]] = C_j^i[v[j]]$  for all  $i \in [n-1]$ and  $j \in [w]$ . Then,  $P_{i \in [n-2]}$  locally encode a data structure PaXoS of their input sets  $D^i \leftarrow \text{Encode}(\{(x, C_1^i[v[1]]) | \dots | | C_w^i[v[w]])\})$  using the entries of the received matrix and send  $D^i$  to  $P_{n-1}$ .  $P_{n-1}$  decodes all the  $D^i$ . Then, they compute and sends the OPRF values  $\varphi = H_2(\bigoplus_{i=1}^{n-2} \text{Decode}(D^i, x) \oplus (C_1^{n-1}[v[1]]) | \dots | | C_w^n [v[w]]))$  to  $P_n$ . After receiving the OPRF values,  $P_n$  computes  $\varphi = H_2(C_1^n[v[1]]) | \dots | | C_w^n[v[w]])$  according to its input set, which allows  $P_n$  to find the intersection. This implies that, if  $x \in I$ , the hash function's input from  $P_{n-1}$  and  $P_n$  will be equal. While the output of the PRF would be pseudorandom to  $P_n$  if  $x \notin I$ , the hash function's input from  $P_{n-1}$  will be dramatically different from any  $P_n$ 's input.

#### 4.2. Our Protocol

We show our MPSI protocol in Figure 4. The selection of m, w,  $l_1$ , and  $l_2$  in our MPSI protocol follows [5] and they show how to choose the parameters concretely.

**Parameters:** security parameters  $\lambda$  and  $\sigma$ , protocol parameters m, w,  $l_1$ ,  $l_2$ , hash functions  $H_1: \{0,1\}^* \to \{0,1\}^l$ ,  $H_2: \{0,1\}^w \to \{0,1\}^{l_2}$ , pseudorandom function  $F: \{0,1\}^d \times \{0,1\}^l \to [m]^w$ .

## Precomputation:

•  $P_{i \in [n-1]}$  sample  $s_i \leftarrow {}^{R} \{0,1\}^{w}$ .

• *P<sub>n</sub>* performs following operations:

(i) Initialize a matrix  $E_{m \times w}$  to all 1's  $E_1 = ... = E_w = 1^m$ .

(ii)  $P_n$  randomly samples a binary matrix  $A_{m\times w}^n \leftarrow \{0,1\}$ . For  $j \in [w]$ ,  $P_n$  chooses n-1 shares uniformly at randomly under the constraint that  $A_j^n = A_j^1 \oplus ... \oplus A_j^{n-1}$ . He also uniformly samples a PRF key  $k \leftarrow [0,1]^{\lambda}$ .

(iii) For  $x \in X_n$ , computes  $v = F_k(H_1(x))$  and sets  $E_j[v[j]] = 0$  for all  $j \in [w]$ .

#### **Oblivious Transfer:**

(i)  $P_n$  computes matrix  $B^i = A^i \oplus E$  for  $i \in [n-1]$ .

(ii)  $P_{i\in[n-1]}$  and  $P_n$  run *w* OTs where  $P_n$  is the sender with input  $\{A_j^i, B_j^i\}$  and  $P_{i\in[n-1]}$  is the receiver with inputs  $s_i[j]$ .  $P_{i\in[n-1]}$  obtain *w* number of m-bit strings as the column vectors of  $C_{m_{NW}}^i$ .

#### **Concluding the Intersection:**

(i)  $P_n$  sends k to  $P_{i \in [n-1]}$ .

(ii) For  $x \in X_i$ ,  $P_{i \in [n-2]}$  compute  $v = F_k(H_1(x))$  and encode a PaXos data structure  $D^i$  to  $P_{n-1}$ , where  $D^i$  is encoded as  $D^i \leftarrow \text{Encode}\left\{\left\{(x, C_1^i[v[1]] \| ... \| C_w^i[v[w]])\right\}\right\}$ .

(iii) For each  $x \in X_{n-1}$ ,  $P_{n-1}$  decode  $D^{i \in [n-2]}$  according to  $v = F_k(H_1(x))$ , and computes its OPRF values  $\varphi = H_2\left(\bigoplus_{i=1}^{n-2} \text{Decode}(D^i, x) \oplus \left(C_1^{n-1}[v[1]] \parallel ... \parallel C_w^{n-1}[v[w]]\right)\right)$  according to  $v = F_k(H_1(x))$  and send  $\varphi$  to  $P_n \circ$ 

(iv) Let  $\psi$  represent the set of OPRF values obtained from  $P_{n-1}$ . For  $x \in X_n$ ,  $P_n$  computes its OPRF value  $\varphi = H_2(C_1^n[v[1]] \parallel ... \parallel C_w^n[v[w]])$  according to  $v = F_k(H_1(x))$  and outputs x if  $\varphi \in \psi$ .

Figure 4. Our MPSI protocol.

## 4.3. Protocol Correctness

 $P_n$  constructs the special matrices  $A^i$  and  $B^i$  for  $P_{i \in [n-1]}$  such that  $v = F_k(H_1(x))$ computed for each  $x \in X_n$  satisfies  $A^i_j[v[j]] = B^i_j[v[j]]$  for all  $j \in w$ . Let x be the intersection element. Since each column of matrix  $A^n_j$  is composed of uniform random shares as  $A^n_j = A^1_j \oplus \ldots \oplus A^{n-1}_j$  for  $j \in w$ , after the client  $P_{i \in [n-1]}$  runs OTs with  $P_n$ , the matrix  $C^i_j$  is obtained, satisfying  $A^i_i[v[j]] = C^i_i[v[j]]$ . It holds that  $A^n_i[v[j]] = \bigoplus_{i=1}^{n-1} C^i_i[v[j]]$  for each  $x \in I$ .

Based on the nature of the constructed data structure, we have  $Decode(D^i, x) = C_1^i[v[1]] || \dots || C_w^i[v[w]]$ . So, for  $x \in I$ , let  $v = F_k(H_1(x))$ , and we can always satisfy  $H_2\left(\bigoplus_{i=1}^{n-1} (C_1^i[v[1]] || \dots || C_w^i[v[w]])\right) = H_2(A_1^i[v[1]] || \dots || A_w^i[v[w]])$ .

#### 4.4. Protocol Security

**Theorem 1.** If F is a PRF,  $H_1$  and  $H_2$  are random oracles, and the underlying OT is protected against malicious receivers, then our MPSI protocol has one-sided malicious security which can be secure against malicious clients when  $m, w, l_1$ , and  $l_2$  are chosen appropriately.

**Proof of Theorem 1.** We consider any client  $P = \{P_1, \ldots, P_{n-1}\}$  corrupted by an adversary  $\mathcal{A}$ . Let l clients  $P_1, \ldots, P_l$  be corrupted, making the number of uncorrupted clients (n-l-1). Given  $\{X_i\}_{i \in [l]}$ , the simulator  $\mathcal{S}$  interacts with  $\{P_i\}_{i \in [l]}$  as follows.  $\mathcal{S}$  samples random matrices  $\{C_i\}_{i \in [l]} \in \{0, 1\}^{m \times w}$  and performs malicious OT simulator on  $\{P_i\}_{i \in [l]}$  with outputs  $C_1^i, \ldots, C_w^i$ .  $\mathcal{S}$  honestly chooses PRF key k and sends k to  $\{P_i\}_{i \in [l]}$ . The simulator  $\mathcal{S}$  constructs random data structures representing honest parties according to the

randomness of the matrices.  $T_1$  and  $T_2$  are initialized to an empty table. In  $P_{i\in[n-1]}$ 's query x to  $H_1$ , S records  $(x, H_1(x))$  in table  $T_1^i$ . In  $P_{n-1}$ 's query y to  $H_2$ , S records  $(y, H_2(y))$  in table  $T_2$ . When  $P_n$  receives OPRF value  $\Psi$ , S finds all  $\varphi \in \Psi$  such that  $\varphi = H_2(y)$  for some y in  $T_2$ , and  $y = \bigoplus_{i \in [l]} (C_1^i[v[1]] | | \dots | | C_w^i[v[w]]) \oplus (\bigoplus_{j \in [n-t-1]} (C_1^j[v[1]] | | \dots | | C_w^j[v[w]]))$  where  $v = F_k(H_1(x))$  for  $x \in T_1^1 \cap \dots \cap T_1^{n-1}$ . Finally, S can send these x to ideal functionality.

Let  $Q_1^i, Q_2$  be a set of queries  $P_{i \in [n-1]}$  and  $P_{n-1}$  make to  $H_1$  and  $H_2$ , respectively, and

let  $Q = \bigcap_{i=1}^{n-1} Q_1^i$ ,  $Q_1^i = |Q_1^i|$ , and  $Q_2 = |Q_2|$ . We will misuse notation: for matrix  $C_{m \times w}$  and vector  $u \in [m]^w$ , C[v] means  $C_1[v[1]]||...||C_w[v[w]]$ . For the set V of vectors in  $[m]^w$ , the set  $\{C[V]|v \in V\}$  is denoted by C[V].

We prove  $\operatorname{Real}_{\mathcal{A}}^{\prod}(X_1,\ldots,X_n) \stackrel{c}{\approx} \operatorname{Ideal}_{\mathcal{S}}^{\mathcal{F}}(X_1,\ldots,X_n)$ .

- Hyb0 The outputs of parties in the real world.
- Hyb1 Similar to Hyb0, but S performs OT simulator on  $\{P_i\}_{i \in [l]}$  to obtain  $s_i$ . If  $s_i[j] = 0$ , it randomly chooses string  $A_j^i$  of length m and constructs matrix  $B_j^i = A_j^i \oplus D_j$ , and it randomly chooses string  $B_j^i$  of length m and constructs matrix  $A_j^i = B_j^i \oplus D_j$ ; otherwise, it gives  $C_1^i, \ldots, C_w^i$  to OT simulator as output. Hyb1 is computationally indistinguishable from Hyb0 due to OT security against malicious receiver.
- Hyb2 Similar to Hyb1 except that the protocol terminates if there exists  $x^a$ ,  $x^b \in X_1 \cup X_2 \cup \dots \cup X_n$ ,  $x^a \neq x^b$  such that  $H_1(x^a) = H_1(x^b)$ . Since  $H_1$  is a random oracle, the protocol is aborted with negligible probability.
- Hyb3 Same as Hyb2, but, for each OPRF value  $\varphi$  received by  $P_n$ , if  $\varphi \notin H_2(Q_2)$ , then  $P_n$  ignores  $\varphi$ . Since  $H_2$  is a random oracle, the probability of changing  $P_n$ 's output is negligible.  $\varphi$  equals the output of  $H_2$  on one of  $P_n$ 's elements with negligible probability.
- Hyb4 Same as Hyb3 except that the protocol terminates if there exists  $y \in Q_2$ ,  $y' \in A[F_k(H_1(X_n))]$  with  $y \neq y'$  and  $H_2(y) = H_2(y')$ . Since  $H_2$  is a random oracle, the protocol is aborted with negligible probability.
- Hyb5 Same as Hyb4, but, for each OPRF value  $\varphi$  received by  $P_n$ ,  $P_n$  ignores  $\varphi$  when calculating the set intersection if  $\varphi = H_2(y)$  for some  $y \in Q_2$ , where  $y \notin \left( \bigoplus_{i \in [t]} (C^i[F_k(H_1(Q))]) \right) \oplus \left( \bigoplus_{j \in [n-t-1]} (C^j[F_k(H_1(Q))]) \right)$ . This hybrid changes output only if there exist  $x \in X_n$  satisfying  $\varphi = H_2(A[F_k(H_1(x))])$ , which implies  $y = A[F_k(H_1(x))]$  via the terminate condition added in Hyb4. Note that if  $x \in X_n$  and  $x \in Q$ , because of the construction of E, we then have  $y = A[F_k(H_1(x))] = \bigoplus_{i \in [l]} (C^i[F_k(H_1(x))]) \oplus \left( \bigoplus_{j \in [n-l-1]} (C^j[F_k(H_1(x))]) \right) \in \left( \bigoplus_{i \in [t]} (C^i[F_k(H_1(Q))]) \right)$ . Therefore, we need only think about  $x \in X_n \setminus Q$ . For all  $x \in X_n$ ,  $A[F_k(H_1(x))] = \bigoplus_{i \in [l]} (C^i[F_k(H_1(x))]) \oplus \left( \bigoplus_{j \in [n-l-1]} (C^j[F_k(H_1(x))]) \right)$ , the output of Hyb5 changes only if there exist  $x \in X_n \setminus Q$ ,  $y \in Q_2$  satisfying  $y = \bigoplus_{i \in [l]} (C^i[F_k(H_1(x))]) \oplus \left( \bigoplus_{j \in [n-l-1]} (C^j[F_k(H_1(x))]) \right)$ . Suppose there is a PPT adversary  $\mathcal{A}$  that, with non-negligible probability, produces Q,  $Q_2$ , and  $X_n$  such that there exist  $y \in Q_2$ ,  $x \in X_n \setminus Q$  satisfying  $y = \bigoplus_{i \in [l]} (C^i[F_k(H_1(x))]) \right)$ . Then, [5] shows we can break the security of the PRF.
- Hyb6 Same as Hyb5 except that the protocol terminates if there exists  $x^a \in Q$ ,  $x^b \in X_n$  such that,  $y = \bigoplus_{i \in [l]} (C^i[F_k(H_1(x^a))]) \oplus (\bigoplus_{j \in [n-l-1]} (C^j[F_k(H_1(x^a))])) = A[F_k(H_1(x^b))]$  but  $x^a \neq x^b$ . The protocol is aborted with negligible probability because of the security of the PRF.
- Hyb7 Same as Hyb6 except that  $P_n$ 's outputs are substituted by its outputs in the ideal world. Hyb7 can change  $P_n$ 's outputs if and only if there exists a value  $\varphi$  received by  $P_n$  and considered by  $P_n$  such that  $\varphi = H_2\left(\bigoplus_{i \in [l]} \left(C^i[F_k(H_1(x^a))]\right) \oplus \left(\bigoplus_{j \in [n-l-1]} \left(C^j[F_k(H_1(x^a))]\right)\right)\right)$  for

some  $x^a \in Q$ , and  $\varphi = A\left[F_k(H_1(x^b))\right]$  for some  $x^b \in X_n$ ,  $x^a \neq x^b$ . Because  $H_2$  is a random oracle,  $\bigoplus_{i \in [l]} \left(C^i[F_k(H_1(x^a))]\right) \oplus \left(\bigoplus_{j \in [n-l-1]} \left(C^j[F_k(H_1(x^a))]\right)\right) \neq A\left[F_k(H_1(x^b))\right]$  is aborted via terminate condition in Hyb6 with negligible probability.

- Hyb8 Same as Hyb7 except that the protocol does not terminate. Hyb7 and Hyb8 are computationally indistinguishable since  $H_1$  and  $H_2$  are random oracles and  $F_k$  is a PRF.
- Hyb9 The output in the ideal world. The difference between Hyb9 and Hyb8 is that S samples a random matrix C and encodes a data structure PaXoS, which is identically distributed.

## 5. Performance Evaluation

## 5.1. Complexity Analysis

To better evaluate the complexity of the protocol, we first need to perform a simple analysis of the overall protocol process. It is important to note that this protocol uses only inexpensive tools such as OTs and bitwise operations, making it concretely efficient. We treat *t* as the set sizes and set m = t as in [5]. So, *w* can be viewed as a value dependent on  $\lambda$  by fixing *m* and *t*.

Party  $P_n$  is referred to as the leader carrying the majority overhead of the protocol, while the others are referred to as clients. Regarding the complexity of the protocol,  $P_n$  designs matrices of a particular form, requiring linear complexity in t. Then, they perform w OTs for clients independently, resulting in linear complexity in the number of OTs. Moreover,  $P_{i \in [n-2]}$  just do encoding operations for data structure  $D^i$ , and  $P_{n-1}$  does hashing, bitwise-XOR, and decoding operations, which require linear communication and computation complexities. Although the computational overhead of  $P_{n-1}$  is larger than that of other clients, they do not need to encode and send a data structure. From this, we can regard the overall communication and computation costs as uniformly distributed across all clients.

Note that our protocol can be divided into offline and online phases. Only lightweight procedures are required in the online phase, and communication and computation costs associated with performing OT can be handled in the offline phase. In addition, the bits exchanged among the parties concerning the random OT and the optimized malicious OT extension are summarized in Table 3.

Table 3. Bits sent for leader and client.

Communication Party	Total Bit Transmission
$P_n \to P_{i \in [n-1]}$	tw
$P_{i\in[n-1]} \to P_n$	$w(\lambda-1)$
$P_{i\in[n-2]} \to P_{n-1}$	tw
$P_{n-1} \rightarrow P_n$	$tl_2$

#### 5.2. Comparison

It should be noted that, due to the variations in architectures and security levels, making a fair comparison is challenging. Nevertheless, we have endeavored to include some recent studies pertaining to diverse security models (e.g., semi-honest, malicious, etc.). So, we contrast the complexity of communication and computation with [13–15] in Table 4, where *n* is the number of parties, *k* is the number of hash functions, *t* is the size of input sets, and  $\lambda$  is the security parameter. In our MPSI protocol, the communication and computation complexity of the leader are  $O(tn\lambda)$ , which is linear in the number of parties. Meanwhile, the complexity for the client remains constant regardless of the number of parties involved (namely,  $O(t\lambda)$ ) because the client  $P_{i \in [n-1]}$  only needs to compute and

send a data structure  $D^i$  and does not need to perform additional data transfers with other parties. Therefore, our protocol achieves a good trade-off between communication and computation overhead.

Dricks col	Communication		Computation		Socurity Model
Protocol	Leader	Clients	Leader	Clients	- Security would
[13]	$O(tn\lambda)$ $O(log(n)tn\lambda k)$	$O(t\lambda k)$ $O(\log(n)tn\lambda k)$	$O(tn\lambda)$ $O(tn\lambda k)$	$O(tn\lambda)$ $O(tn\lambda k)$	Semi-Honest
[15]	$O(\log(n)tn\lambda k)$ $O((n^2 + tn)\lambda)$	$O(\log(n)tn\lambda k)$ $O(t\lambda)$	$O(tn \log(t))$	$O(t\log(t))$	Malicious
Ours	$O(tn\lambda)$	$O(t\lambda)$	$O(tn\lambda)$	$O(t\lambda)$	One-sided Malicious

Table 4. Complexity of MPSI protocols.

Figure 5 shows the security levels of the discussed protocols. Compared with [13], our protocol achieves a stronger security model without sacrificing communication and computation costs. We implement one-sided malicious security and [14] implements the Aug semi-honest model. It is difficult to define which security model is more practical, but our protocol has better computation and communication performance. Although the security model in [15] is higher-performing, our protocol has greater communication performance and achieves a better trade-off between communication and computation.



Figure 5. Comparison of security levels.

#### 5.3. Experimental Evaluation

In order to compare the runtime overhead of each protocol more intuitively, simulation experiments and a results analysis were performed. It should be noted that the time consumed by this protocol is the average time of multiple experiments. The experimental platform was Windows 10, Intel (R) Core (TM) i5-8250U CPU @ 1.60 GHz 1.80 GHz, 8.00 GB of RAM, and a compiled environment of Dev-C++5.11.

We first consider the total time required for each protocol to execute with different numbers of set elements. It is assumed that n = 100,  $k = \lambda = 128$ , and  $t = 2^{10}, 2^{11}, 2^{12}, 2^{13}$  are chosen for the comparison experiment, and Figure 6 shows the total running time of the protocol as a function of the number of elements contained in the set.



Figure 6. Running time vs. set cardinality.

From Figure 6, the total time overhead in each protocol grows essentially linearly as the number of set elements continues to increase. However, the time of our MPSI protocol increases the slowest when the fixed set cardinality is small. Our MPSI protocol has the slowest time growth rate.

In addition, the effect of the change in the number of parties on the running time of the protocol is further considered. Suppose that the maximum number of elements contained in the set is t = 1000, the security parameters are kept fixed at  $k = \lambda = 128$ , and the number of parties  $n = 10^1, 10^2, 10^3, 10^4$  is selected for the comparison experiment. The total protocol runtime as a function of the number of parties is shown in Figure 7.



Figure 7. Running time vs. the number of parties.

From Figure 7, the running time of all protocols increases gradually with the number of parties. The time overheads of our MPSI protocol are lower than those of the other protocols when n is fixed. In addition, our MPSI protocol has the slowest time growth rate.

#### 6. MPCCache in Edge Computing

This section aims to address the problem of edge collaborative content caching, wherein all parties can jointly cache the most frequently accessed common data items in shared caches. Figure 8 shows the difference between the traditional cache model and edge cache model. Our challenge is to find how to determine a set of the most frequently accessed common items without revealing any underlying data.



Figure 8. Traditional cache model and edge cache model.

## 6.1. Our MPCCache

We describe how to use our MPCCache protocol to handle the edge cache case. The network operators  $P_{i \in [n]}$  respectively own set  $K_i = \{(x_1^i, z_1^i), \dots, (x_t^i, z_t^i)\}$ , where  $x^i \in \{0, 1\}^*$  denotes an identify element and  $z^i \in \{0, 1\}^w$  denotes its associated value. Note that the latter may represent the anticipated frequency of content being accessed or the value to network operators of the cached content. Let the common items  $I = \bigcap_{i=1}^n X_i = \{x_1, x_2 \dots\}$  be the intersection of the identifiers, where  $X_i = \{x_1^i, \dots, x_t^i\}$  is the set of identity for  $P_{i \in [n]}$ . For each common item  $x \in I$ , calculate a sum of the associated values z; that is,  $sum^{(x)} = \sum_{i=1}^n z^{(x)}$ . The sum of a common item is determined as the total of the individual values of the operators for the item.

We present the MPCCache protocol in Figure 9.  $P_{i \in [n-1]}$  conduct simple hashing and  $P_n$  conducts cuckoo hashing that maps common items to the same bucket. According to the PaXoS, all the buckets are compressed into a data structure so that  $P_n$  can efficiently compute the MPCCache. In detail,  $P_{i \in [n-1]}$  choose  $q_j^n$  and  $s_j^n$  uniformly at random for  $j \in [\beta]$ . Notice that  $A_j^n[v[j]] = \bigoplus_{i=1}^{n-1} C_j^i[v[j]]$  for each  $x \in I$ . For  $(x, z) \in K_i$  and  $v = F_k(H_1(x))$ ,  $P_{i \in [n-1]}$  compute  $f_{x_j}^i \stackrel{def}{=} (C_1^1[v[1]]||, \dots, ||C_w^1[v[w]]) \oplus q_j^i$  and  $g_{x_j}^i \stackrel{def}{=} z - s_j^i$ , and send the encoding Encode  $\left(x \mid j, f_{x_j}^i\right)$  and Encode  $\left(x \mid j, g_{x_j}^i\right)$  to  $P_n$ , where  $x_j = (\text{HT}[j]||j)$  means that x is in j – th bucket.  $P_n$  can use  $x_j^n \in \{(x \mid |j) \mid x \in \text{GT}_n[j]\}_{j \in [\beta]}$  to obtain the correct decoding  $f_{x_j}^i$  and  $g_{x_j}^i$  if  $x_j^n = x_j^i$ ; it is otherwise random. Then,  $P_n$  computes  $q_j^n \stackrel{def}{=} \bigoplus_{i=1}^{n-1} f_{x_j}^i \oplus (A_1^n[v[1]]||\dots||A_w^n[v[w]])$  and  $s_j^n \stackrel{def}{=} \sum_{i=1}^{n-1} g_{x_j}^i + z$ . Finally,  $P_{i \in [n]}$  input  $q_{j \in [\beta]}^n$  and  $s_{j \in [\beta]}^n$ , respectively, to check whether  $\bigoplus_{i=1}^n q_j^i = 0$  is based on a garbled circuit, and, if so, obtain the sum of the corresponding common item  $\bigoplus_{i=1}^n s_i^i$ .

**Parameters:** security parameters  $\lambda$  and  $\sigma$ , protocol parameters m, w,  $l_1$ ,  $l_2$ , hash function  $H_1: \{0,1\}^* \to \{0,1\}^{l_1}$ , pseudorandom function  $F: \{0,1\}^{\lambda} \times \{0,1\}^{l_1} \to [m]^w$ , Cuckoo, Simple hashing, and number of bins  $\beta$ .

Input of  $P_{i \in [n]}$ :  $K_i = \{(x_1^i, z_1^i), ..., (x_t^i, z_t^i)\} \subset (\{0,1\}^*, \{0,1\}^w)^t$ .

#### Precomputation:

•  $P_{i \in [n-1]}$  do Simple hashing which maps elements in  $X_{i \in [n-1]}$  to the hash table  $HT_i$ with  $\beta$  bins and sample random strings  $q_j^i \leftarrow \frac{R}{2} \{0,1\}^w$  and  $s_j^i \leftarrow \frac{R}{2} \{0,1\}^w$  for  $j \in [\beta]$ .

•  $P_n$  does Cuckoo hashing which maps elements in  $X_n$  to the hash table  $GT_n$ . Then  $P_n$  perform the same Precomputation as in Figure 4.

#### **Oblivious Transfer:**

 $P_n$  and  $P_{i \in [n-1]}$  run the same Oblivious Transfer as in Figure 4.

**Compute:** 

(i)  $P_n$  sends key k to  $P_{i \in [n-1]}$ .

(ii) For each  $(x,z) \in K_i$ ,  $P_{i \in [n-1]}$  compute  $v = F_k(H_1(x))$  and encode two data structures PaXoS  $D_x^i$  and  $D_z^i$  based on  $X_i^i = \{(x \parallel j) \mid x \in HT_i[j]\}_{j \in [\beta]}$  and sent to  $P_n$ , where  $D_x^i$  is encoded as  $D_x^i \leftarrow \text{Encode}(\{(x \parallel j, (C_1^1[v[1]] \parallel ... \parallel C_w^1[v[w]]) \oplus q_j^i)\}))$  and  $D_z^i$  is encoded as  $D_z^i \leftarrow \text{Encode}(\{(x \parallel j, z - s_j^i)\})$ .

(iii) After  $P_n$  receives  $D_x^i$  and  $D_z^i$  sent by  $P_{i\in[n-1]}$ , compute  $v = F_k(H_1(x))$  for each  $(x,z) \in K_n$  and generate query set  $X_n^i = \{(x \parallel j) \mid x \in GT_n[j]\}_{j\in[\beta]}$  according to  $GT_n$ .  $P_n$  finally computes  $q_j^n \stackrel{\text{def}}{=} \left( \bigoplus_{i=1}^{n-1} \text{Decode}(D_x^i, x \parallel j) \right) \oplus \left( A_1^n[v[1]] \parallel \dots \parallel A_w^n[v[w]] \right)$  and  $s_j^n \stackrel{\text{def}}{=} \sum_{i=1}^{n-1} \text{Decode}(D_z^i, x \parallel j) + z$ . **Garbled Circuit:** -Input from  $P_{i\in[n]}$  is  $q_{j\in[\beta]}^n$  and  $s_{j\in[\beta]}^n$ .

-Output to  $P_{i\in[n]}$  is  $u_j$  where  $u_j = \sum_{i=1}^n s_j$  if  $\bigoplus_{i=1}^n q_j^i = 0$ , otherwise  $u_j = 0$ .

Figure 9. Our MPCCache protocol.

## 6.2. Correctness and Security

Correctness: Section 4.3 proves that  $A_j^n[v[j]] = \bigoplus_{i=1}^{n-1} C_j^i[v[j]]$  for each  $x \in I$  and  $v = F_k(H_1(x))$ ; that is,  $A_1^n[v[1]] || \dots || A_w^n[v[w]] = \bigoplus_{i=1}^{n-1} (C_1^1[v[1]]) || \dots || C_w^n[v[w]])$ . Via the property of the data structure PaXoSs  $D_x^i$  and  $D_z^i$  constructed by  $P_{i\in[n-1]}$ , for  $x \in I, j \in [\beta]$ , and  $v = F_k(H_1(x))$ , we always have  $\bigoplus_{i=1}^{n-1} \text{Decode}(D_x^i, x || j) = \bigoplus_{i=1}^{n-1} ((C_1^i[v[1]]) || \dots || C_w^i[v[w]]) \oplus q_j^i)$ ,  $\bigoplus_{i=1}^{n-1} \text{Decode}(D_z^i, x || j) = \sum_{i=1}^{n-1} (z^i - s_j^i)$ . At the same time,  $P_n$  defines  $q_j^n \stackrel{def}{=} (\bigoplus_{i=1}^{n-1} \text{Decode}(D_x^i, x || j) \oplus (A_1^n[v[1]]) || \dots || A_w^n[v[w]])$  and  $s_j^n \stackrel{def}{=} \sum_{i=1}^{n-1} \text{Decode}(D_z^i, x || j) + z$  in terms of the  $D_x^i$  and  $D_z^i$  they receive from  $P_{i\in[n-1]}$ . That is, when  $x \in I$ , it always satisfies that  $\bigoplus_{i=1}^n q_i^i = 0$  and  $\bigoplus_{i=1}^n s_i^i = \sum_{i=1}^n z^i$ .

**Theorem 2.** If F is a PRF and  $H_1$  is a random oracle, then the construction of our MPCCache protocol has colluding semi-honest security, given the OT, PaXoS, GC, and appropriate parameters.

**Proof of Theorem 2.** If we consider l parties  $\{P_i\}_{i \in [l]}$  to be corrupted by an adversary  $\mathcal{A}$ , then the number of uncorrupted parties is (n - l). Given  $\{K_i\}_{i \in [l]}$ , the simulator  $\mathcal{S}$  interacts with  $\{P_i\}_{i \in [l]}$  as follows.  $\mathcal{S}$  samples random matrices, performs OT, chooses the PRF key k and sends k to  $\{P_i\}_{i \in [l]}$ . The simulator  $\mathcal{S}$  constructs random data structures representing honest parties according to the randomness of the matrices.  $\mathcal{S}$  sends two data structures  $D_x^i$  and  $D_z^i$  constructed on a PaXoS to ideal functionality. We prove  $\text{Real}_{\mathcal{A}}^{\Pi}(K_1, \ldots, K_n) \stackrel{c}{\approx} \text{Ideal}_{\mathcal{S}}^{\mathcal{F}}(K_1, \ldots, K_n)$ .

- Hyb0 The outputs of parties in the real world.
- Hyb1 Same as Hyb1, Hyb2, and Hyb6 in Section 4.4.
- Hyb2 Similar to Hyb1 except that the decoding executions of the PaXoS are replaced as follows. When  $\{P_i\}_{i \in [l]}$  does not contain  $P_n$ , S receives nothing from the data structure PaXoS. When  $\{P_i\}_{i \in [l]}$  contains  $P_n$ , if  $x \in I$ ,  $P_n$  receives  $D_x^i$  and  $D_z^i$ , thus  $(C_1^i[v[1]]||...||C_w^i[v[w]]) \oplus q_j^i$ ,  $(z^i - s_j^i)$  for the PaXoS involving the non-colluding party  $\{P_i\}_{i \in [n-l]}$  and  $j \in [\beta]$ . Note that  $q_j^i$  and  $s_j^i$  are used in the above expression for each bin  $j \in [\beta]$ . Since these values are uniform, so are  $D_x^i$  and  $D_z^i$ . Therefore, we replace the decoding outputs of the PaXoS with random ones. Otherwise, all the decoding outputs of the PaXoS are uniformly random from the perspective of  $P_n$  and  $\{P_i\}_{i \in [l]}$ . Hyb2 is computationally indistinguishable from Hyb1 due to the PaXoS's security.
- Hyb3 The output in the ideal world. The only difference between Hyb3 and Hyb2 is that S executes the output of the circuit.

### 7. Conclusions

In this work, we design an efficient MPSI protocol and the MPCCache protocol to better solve the information leakage problem in resource sharing. The proposed MPSI protocol derived from multi-point OPRF demonstrates concrete efficiency in achieving one-sided malicious security. The protocol also leads to a better trade-off between communication and computational overhead. It is based on OT and a data structure PaXoS and achieves linear computation and communication complexity concerning the input set size of each party. In our MPSI protocol, the asymptotic communication and computational complexity of the clients are largely determined by the size of the input sets rather than the number of parties (namely,  $O(t\lambda)$ ). Overall, this research has contributed to the development of efficient MPSI protocols for multiple parties in practice. In fact, we apply the MPCCache protocol to edge caching scenarios using a simple transformation of the MPSI protocol. The MPCCache protocol under the semi-honest model can support the computation of specific functions on intersections. It is our belief that future work can improve the fairness of the MPSI protocol, as well as propose more application scenarios with practical application value.

**Author Contributions:** Conceptualization, J.Z., L.Y. and Y.T.; methodology, L.Y. and Y.T.; validation, J.Z., L.Y. and Y.T.; formal analysis, L.Y. and M.J.; writing—original draft preparation, L.Y.; writing—review and editing, Y.T., S.W. and M.J.; supervision, Y.T. and S.W.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by Henan Key Laboratory of Network Cryptography Technology (No. LNCT2022-A11), the Henan Province Key R&D and Promotion Special Project (No. 212102210166), and the PhD Foundation of Henan Polytechnic University (No. B2021-41).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

#### References

- Wei, X.; Xu, L.; Cai, G.; Wang, H. Secure approximate pattern matching protocol via Boolean threshold private set intersection. *Int. J. Intell. Syst.* 2022, 37, 9245–9266. [CrossRef]
- Kales, D.; Rechberger, C.; Schneider, T.; Senker, M.; Weinert, C. Mobile private contact discovery at scale. In Proceedings of the 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, USA, 14–16 August 2019; pp. 1447–1464.
- Ion, M.; Kreuter, B.; Nergiz, E.; Patel, S.; Saxena, S.; Seth, K.; Shanahan, D.; Yung, M. Private intersection-sum protocol with applications to attributing aggregate ad conversions. *Cryptol. ePrint Arch.* 2017. preprint. Available online: https://eprint.iacr.org/ 2017/738 (accessed on 11 September 2023).

- 4. Nguyen, D.T.; Trieu, N. MPCCache: Privacy-preserving multi-party cooperative cache sharing at the edge. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada*; Springer International Publishing: Berlin/Heidelberg, Germany, 2022; pp. 80–99. [CrossRef]
- Chase, M.; Miao, P. Private set intersection in the internet setting from lightweight oblivious PRF. In Proceedings of the Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2020; pp. 34–63. [CrossRef]
- Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. SpOT-light: Lightweight private set intersection from sparse OT extension. In Proceedings of the Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; pp. 401–431. [CrossRef]
- Pinkas, B.; Schneider, T.; Zohner, M. Scalable private set intersection based on OT extension. ACM Trans. Priv. Secur. TOPS 2018, 21, 1–35. [CrossRef]
- Cong, K.; Moreno, R.C.; da Gama, M.B.; Dai, W.; Iliashenko, I.; Laine, K.; Rosenberg, M. Labeled PSI from homomorphic encryption with reduced computation and communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 1135–1150. [CrossRef]
- Chen, H.; Huang, Z.; Laine, K.; Rindal, P. Labeled PSI from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1223–1237. [CrossRef]
- Pinkas, B.; Schneider, T.; Weinert, C.; Wieder, U. Efficient circuit-based PSI via cuckoo hashing. In Proceedings of the Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018; pp. 125–157. [CrossRef]
- Pinkas, B.; Schneider, T.; Tkachenko, O.; Yanai, A. Efficient circuit-based PSI with linear communication. In Proceedings of the Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019; pp. 122–153. [CrossRef]
- 12. Chandran, N.; Gupta, D.; Shah, A. Circuit-PSI With Linear Complexity via Relaxed Batch OPPRF. *Proc. Priv. Enhancing Technol.* 2022, 1, 353–372. [CrossRef]
- Kavousi, A.; Mohajeri, J.; Salmasizadeh, M. Efficient scalable multi-party private set intersection using oblivious PRF. In Proceedings of the Security and Trust Management: 17th International Workshop, STM 2021, Darmstadt, Germany, 8 October 2021; pp. 81–99. [CrossRef]
- Inbar, R.; Omri, E.; Pinkas, B. Efficient scalable multiparty private set-intersection via garbled bloom filters. In Proceedings of the Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, 5–7 September 2018; pp. 235–252.
- Ghosh, S.; Nilges, T. An algebraic approach to maliciously secure private set intersection. In Proceedings of the Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019; pp. 154–185. [CrossRef]
- Kolesnikov, V.; Kumaresan, R.; Rosulek, M.; Trieu, N. Efficient batched oblivious PRF with applications to private set intersection. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 818–829.
- Pinkas, B.; Schneider, T.; Zohner, M. Faster private set intersection based on {OT} extension. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14), San Diego, CA, USA, 20–22 August 2014; pp. 797–812.
- Nevo, O.; Trieu, N.; Yanai, A. Simple, fast malicious multiparty private set intersection. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, 15–19 November 2021; pp. 1151–1165. [CrossRef]
- Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. PSI from PaXoS: Fast, malicious private set intersection. In Proceedings of the Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; pp. 739–767. [CrossRef]
- Ben-Efraim, A.; Nissenbaum, O.; Omri, E.; Paskin-Cherniavsky, A. Psimple: Practical multiparty maliciously-secure private set intersection. In Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, 30 May–2 June 2022; pp. 1098–1112.
- 21. Bui, D.; Couteau, G. Private Set Intersection from Pseudorandom Correlation Generators. IACR Cryptol. ePrint Arch. 2022, 2022, 334.
- Chida, K.; Hamada, K.; Ichikawa, A.; Kii, M.; Tomida, J. Communication-Efficient Inner Product Private Join and Compute with Cardinality. Cryptol. ePrint Arch. 2022. preprint. Available online: https://eprint.iacr.org/2022/338 (accessed on 11 September 2023).
- Miao, P.; Patel, S.; Raykova, M.; Seth, K.; Yung, M. Two-sided malicious security for private intersection-sum with cardinality. In Proceedings of the Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2020; pp. 3–33. [CrossRef]
- 24. Goldreich, O. Foundations of Cryptography: Volume 2, Basic Applications; Cambridge University Press: New York, NY, USA, 2009.

- 25. Rabin, M.O. How to exchange secrets with oblivious transfer. *Cryptol. ePrint Arch.* 2005. *preprint*. Available online: https://eprint.iacr.org/2005/187 (accessed on 11 September 2023).
- 26. Ishai, Y.; Kilian, J.; Nissim, K.; Petrank, E. Extending Oblivious Transfers Efficiently. Crypto 2003, 2729, 145–161. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.