



A NEW MULTI-PARTY PRIVATE SET INTERSECTION PROTOCOL BASED on OPRFs

Aslı BAY*, Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Antalya Bilim University, asli.bay@antalya.edu.tr

( <https://orcid.org/0000-0002-3820-1778>)

Anıl KAYAN, Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Antalya Bilim University, anil.kayan@antalya.edu.tr

( <https://orcid.org/0000-0002-6531-046X>)

Received: 18.02.2022, Accepted: 16.06.2022

Research Study

*Corresponding author

DOI: 10.22531/muglajsci.1075788

Abstract

In many crucial real-world applications, parties must jointly perform some secure multi-party computation (MPC) while keeping their inputs hidden from other parties. Private Set Intersection (PSI), the specific area of Multi-Party Computation, let the parties learn the intersection of their private data sets without sharing their secret data with others. For instance, a smartphone user downloads a messaging application, naturally, he wants to discover who are the other contacts that are using the same application. The naive and insecure solution is to send all contacts to the server to discover them. However, the user does not want to share his contacts with the application for privacy issues. To handle this, in recent years, companies and organizations start to use PSI to enhance privacy and security with a little cost of communication and computation. In this paper, we introduce a novel method to compute Private Set Intersection with multi parties where there are at least three or more parties participating in the protocol. By employing the Zero-Secret Sharing scheme and Oblivious Pseudo-Random Functions (OPRFs), parties securely calculate the intersection with computational and communication complexities which are both linear in the number of parties.

Keywords: Private set intersection, multi-party private set intersection, multi-party computation, oblivious transfer, oblivious pseudorandom function, zero sharing

OPRF'LERE DAYALI YENİ ÇOKLU KULLANICILI ÖZEL SET KESİŞİMİ PROTOKOLÜ

Özet

Birçok önemli gerçek dünya uygulamasında, taraflar girdilerini diğer taraflardan gizli tutarken bazı güvenli çok taraflı hesaplama (MPC) işlemlerini birlikte yapmalıdır. Çok Taraflı Hesaplamanın özel alanı olan Özel Set Kesişimi (PSI), tarafların gizli verilerini başkalarıyla paylaşmadan veri kümelerinin kesişimini öğrenmelerini sağlar. Örneğin, bir akıllı telefon kullanıcısı bir mesajlaşma uygulaması indirir, doğaç olarak aynı uygulamayı kullanan diğer kişilerin kim olduğunu keşfetmek ister. Naif ve güvensiz çözüm, tüm kişileri, sunucuya göndermek ve kim olduklarını keşfetmektir. Ancak kullanıcı, gizlilik sorunları için uygulama ile temaslarını paylaşmak istemezler. Bunu halletmek için, son yıllarda şirketler ve kuruluşlar, küçük bir iletişim ve hesaplama maliyetiyle gizliliği ve güvenliği artırmak için PSI kullanmaya başladılar. Bu makalede, protokole en az üç veya daha fazla tarafın katıldığı çoklu taraflarla Özel Set Kesişimi hesaplamak için yeni bir yöntem tanıtıyoruz. Taraflar, Sıfır Gizli Paylaşım ve Habersiz Söзде Rastgele Fonksiyonları kullanarak, her ikisi de kullanıcı sayısı ile doğrusal olan hesaplama ve iletişim karmaşıklıklarıyla kesişimi güvenli bir şekilde hesaplar.

Anahtar Kelimeler: özel set kesişimi, çoklu kullanıcıli özel set kesişimi, çok taraflı hesaplama, habersiz transfer, habersiz söзде rastgele fonksiyonlar, sıfır gizli paylaşım.

Cite

BAY, A., KAYAN, A., (2022). "A New Multi-Party Private Set Intersection Protocol Based on OPRFs", Mugla Journal of Science and Technology, 8(1), 69-75.

1. Introduction

Private Set Intersection (PSI) is a way of finding the intersection of two secret data sets without disclosing their elements other than the intersection. In a formal

way, two parties P_1 and P_2 holding their data sets S_1 and S_2 are willing to compute the intersection $S_1 \cap S_2$ without revealing their data sets. As our lives are getting digital day by day, as a consequence of this, people inevitably demand more secure applications from companies or organizations. Therefore, in the past years,

PSI has been enthusiastically researched and found various practical and fast applications [1-6].

Traditionally PSIs are designed for two parties. At certain setups, we can derive much more information than the two-party intersections and it is only possible to get these meaningful inferences with multiple parties simultaneously joined together. In this setting, there are more than two parties, say t parties P_1, \dots, P_t having their own sets S_1, \dots, S_t respectively, are interested in finding $S_1 \cap \dots \cap S_t$ without revealing any other information.

Both, the usages of PSIs and Multi-party Private Set Intersections (MPSIs) are very broad in real-life applications and can be exemplified as follows:

Contact Tracing: In the ongoing COVID-19 pandemic, the researchers from UC Berkeley introduced a lightweight way for contact tracing. Users are alerted by an application if they contact any diagnosed people with the disease while protecting private information [7].

Password Checkup: Based on the numbers from the paper 1.5% of logins on the web involve breached credentials. Another use of PSI is checking whether your password is leaked or not. Users can compare their credentials with millions of entries in breached databases without revealing any part of it [8].

Ad Efficiency: Facebook, Datalogix, Epsilon, and Acxiom, consumer data collection companies measure how well an ad is performing. Rather than using naive hashing solutions to compare merchant's lists of customers and advertiser's lists, they are using PSI methods to increase security [9].

Genome Discovery: In the field of paternity and ancestry testing, it can be possible to perform these tasks without revealing any further individual genomic information [10].

1.1. Related Work

The naive solution of PSI is where all the parties deal with a hashing algorithm and then apply this algorithm to their sets. By comparing resulting hashes, parties easily come up with an intersection but the problem with this approach is when the input domains are small, the brute force attack can be applied. Early researches are based on public-key cryptography techniques that have computational challenges rather than symmetric key cryptography [11-13].

There are different types of techniques that are used to build PSIs and MPSIs. The well-known ones are permutation-based hashing [5], circuit-based computations [14-16], oblivious transfer [1,3,17], Bloom filters [18], cuckoo hashing [2], oblivious programmable hashing [19]. To the best of the author's knowledge, the fastest PSI protocols are designed by Pinkas et al. [5,19] where the former is based on oblivious transfer and permutation-based hashing and the latter uses a generic circuit-based multi-party computation.

The earliest MPSI protocol by Freedman et al. [20] is based on Oblivious Polynomial Evaluation (OPE) which makes use of a homomorphic encryption scheme. In this

scheme, users' private data sets are represented as polynomials and the coefficients are encrypted by a homomorphic encryption scheme and obviously evaluated on the other party's data sets. The same OPE technique is also used by Kissner et al. [21] which has a quadratic complexity in the number of parties. Another work by Hazay and Venkatasubramanian is similar to work [22] which uses an additively homomorphic public-key encryption scheme with threshold decryption. The computational complexity of this protocol is linear in the input data sets. Recently, Kolesnikov et al. proposed an MPSI which is based on Oblivious Transfer which makes their protocol faster due to the usage of symmetric algorithms. The computational complexity is quadratic in the number of parties while it is independent of the size of the sets.

1.2. Our Contribution

Inspired by Chase-Miao's PSI protocol [3] which is based on Oblivious Pseudorandom Functions (OPRFs), we design an efficient and secure multi-party PSI. In our design, we consider t parties each of which has private data sets of the same size n , and a Trusted Dealer D who has no clue about the secret key and data sets of the parties. The effectiveness of our protocol is that any party can compute the intersection, however, for the sake of simplicity, we consider the first party P_1 as the server who outputs the intersection. Our scheme is based on again Oblivious Pseudorandom Functions and zero secret sharing to protect the privacy of the parties. Our protocol is efficient in terms of both computational and communication complexities which are both linear in the number of parties.

2. Preliminaries

In this section, we will provide the necessary background for our protocol. We start with the notations, then we explain the security model of the protocol. Finally, we briefly explain the Chase-Miao PSI protocol that we inspired from [3].

2.1. Notations

We use λ, κ, σ and l_1 to denote the computational and statistical security parameters.

t : Number of parties.

m : The number of rows in the matrices.

w : The number of columns in the matrices.

l_1 : The output length of the Hash function H .

s : A random string to be used in Oblivious Transfer operations.

P_i : The i -th party, where P_1 is the server and the rest is the clients.

X_i : The dataset of P_i .

D : The Trusted Dealer.

n_i : The size of the dataset of P_i , we assume that the all the parties have data set of the same size.

$sh_i^{a,b}$: The i -th secret share part at index (a,b) .

C_i : The matrix constructed via OT operations of P_i .

C: The final matrix constructed by trusted dealer D

PSI: Private Set Intersection.

MPSI: Multi-Party Private Set Intersection.

S: The intersection of S_i 's, $S = \cap_{i=1}^t S_i$.

κ : The computational security parameter.

r : The length of secret shares in bits.

2.2. Security Model

The security of the protocol is based on the semi-honest model with static adversaries. To explain briefly, the parties follow the protocol honestly without deviating from the flow of the protocol and the number of corrupted parties is determined before the start of the protocol.

As we will omit the proof, we refer the reader to the security definition of the semi-honest security for deterministic functionalities defined by Goldreich[23]. The protocol has a trusted Dealer D, who contacts with the clients through a secure communication channel. The security definition for our MPSI protocol matches up with that of Miyaji et al. [24] which can be explained as follows.

An MPSI is player-private secure under the existence of a trusted Dealer **D** if the following two conditions are satisfied:

- The clients can only learn the other client's elements only if the elements are in the intersection $\cap_i S_i$.
- The trusted dealer D cannot learn any information about the data sets of the clients.

Note that the trusted Dealer has no access to the data set of the clients, also he cannot obtain the mutual or full intersection client's data sets by trying the elements from the domain. The reason is that he does not know the secret key **k** of PRF and cannot infer anything from the parties' matrices C_i 's which will be clarified in Sec. 3.

2.3. Technical Background

The Zero-Secret Sharing Scheme: To extend two-party protocols to multiple parties, we usually make use of secret sharing schemes. The idea behind secret sharing schemes is as follows: we divide the secret into several shares in such a way that when a sufficient number of shares are available, which is greater than a threshold number, then the secret can be reconstructed. This means that if there is not enough shares, no one can be able to reconstruct the secret, nor any part of it. In this work, we make use of an additive XOR-based (t, t) secret sharing scheme [25] which can be briefly explained as follows: we create $t - 1$ shares of the same size as the secret s , the last share becomes the Exclusive OR (XOR) of all $t - 1$ shares and the secret s . To reconstruct the secret s , it is simply needed to XOR all t shares. In our case, we use a fixed secret as $s = 0$ which makes the scheme $(t - 1, t)$ as s is already known by the parties.

Oblivious Transfer: Oblivious transfer (OT), proposed by Rabin [26], is a cryptographic protocol executed among two parties: the sender has two inputs m_0 and m_1

and the receiver has a bit b , at the end of the protocol, while the receiver learns m_b while no parties learn any additional information. When there are n OTs are needed, there is an efficient extension of this technique called *OT extension* which requires $\mathcal{O}(\kappa)$ public-key operations instead of $\mathcal{O}(n)$, where κ is a computational security parameter [27].

Single-Point OPRF: As proposed in Kolesnikov et. al. [4], the single point oblivious PRF (OPRF) is defined as follows: let the PRF key k be the two-bit strings $q, s \in \{0,1\}^\lambda$. Let H be a hash function and $F(\cdot)$ be a pseudorandom code that produces a pseudorandom string. The oblivious pseudorandom function is defined as

$$OPRF_k(x) = H(q \oplus [F(x) \cdot s]) \quad (1)$$

where ' \cdot ' is bitwise-AND and ' \oplus ' is bitwise-XOR operands. Here, the OPRF is assumed to be pseudorandom if H is a collision-resistant hash function and F is a pseudorandom generator function, lastly s should be a randomly generated string.

Let's see how we can evaluate the single-point OPRF on the receiver's input y . First of all, the receiver chooses a random r_0 from $\{0,1\}^\lambda$ and computes $r_1 = r_0 \oplus F(y)$. The sender then samples a random string s from $\{0,1\}^\lambda$ where each bit of s is one of the λ choice bits for OT. Then, these two parties execute λ oblivious transfers where the sender acts as a receiver in the OT and inputs single bits $s[1], s[2], \dots, s[\lambda]$. The receiver acts as a sender in the OT and inputs single bits $\{r_0[i], r_1[i]\}_{i \in \{1, \dots, \lambda\}}$. After all OTs are executed, the sender obtains a set of λ bits and sets $q = r_{s[1]}[1] || \dots || r_{s[\lambda]}[\lambda]$. The sender also sets the PRF key as $k = (q, s)$. Then, he chooses x and computes $q \oplus [F(x) \cdot s]$. Therefore, the PRF value on y learned by the receiver is $H(r_0)$ no matter s is chosen. That is, for $x = y$, we have $H(q \oplus [s \cdot f(x)]) = H(r_0 \oplus [s \cdot (F(x) \oplus F(y))]) = H(r_0)$. Otherwise, the receiver has no clue about $OPRF_k(x)$.

Multi-Point OPRF: Instead of executing the single-point OPRF for every value of the receiver which is not efficient, the single-point OPRF is extended to Multi-Point OPRF [3][28]. According to Chase and Miao [3], instead of a vector PRF key k , here they define the key as $m \times w$ matrix. Similarly, we use a hash function H and pseudorandom code $F(\cdot)$ that produces a pseudorandom vector $v \in [m]^w$. The pseudorandom function is defined as

$$OPRF_M(x) = H(M_1[v[1]] || \dots || M_w[v[w]]) \quad (2)$$

We evaluate OPRF on input y , the sender picks a random string s from $\{0,1\}^w$. The receiver prepares (in a way that it will be explained in the PSI protocol) two matrices A and B , where $A_i \in \{0,1\}^m$ and $B_i \in \{0,1\}^m$ are column matrices of A and B respectively, where $1 \leq i \leq w$. There will be w number of execution of OTs where the sender behaves as the receiver and the receiver behaves as the sender as in the single-point OPRF. After all, OTs are executed, the sender gets w columns vectors which will

be assigned to have the PRF key M . As for all $x \in Y$, $OPRF_M(x) = OPRF_A(x)$ is independent of chosen s .

2.4. The Chase-Miao's PSI Protocol

In their protocol [3], Chase and Miao use the multi-point OPRF to find the intersection of two private data sets. Their protocol is as follows:

- **Input:** Let λ, σ be security parameters, $H_1: \{0,1\}^* \rightarrow \{0,1\}^{l_1}$ and $H_2: \{0,1\}^* \rightarrow \{0,1\}^{l_2}$ and pseudorandom function $F: \{0,1\}^\lambda || \{0,1\}^{l_1} \rightarrow \{0,1, \dots, m-1\}^w$ agreed by two parties P_1 and P_2 .
- **Precomputation:**
 1. P_1 chooses a random string $s \in_R \{0,1\}^w$.
 2. P_2 constructs a matrix D which have all entries are set to 1. Let D_1, \dots, D_w be column vectors of D .
 3. P_2 chooses a random key k of length λ for $F, k \in_R \{0,1\}^\lambda$.
 4. P_2 computes $v = F_k(H_1(y))$ for each element $y \in Y$ and update D as $D_i[v[i]] = 0$ for all $i \in \{1, \dots, w\}$.
 5. P_2 constructs a random matrix A of size $m \times w$ and computes another matrix B by $B = A \oplus D$.
- **Oblivious Transfer:** P_1 as the receiver with input choice bits $s[i]$'s and P_2 as the sender with inputs $\{A_i, B_i\}_i$ execute w oblivious transfers (OTs) and P_1 obtains an $m \times w$ matrix C at the end.
- **Computing Intersections with OPRF:**
 1. P_2 sends k to P_1 .
 2. P_1 computes $v = F_k(H_1(x))$ for each element x in his data set.
 3. Then, P_1 computes OPRF value of each element $X \in X$ by $\psi = H_2(C_1[v[1]] || \dots || C_w[v[w]])$ constructs the set of ψ 's as Ψ and send Ψ to P_2 .
 4. P_2 computes $v = F_k(H_1(y))$ for all $y \in Y$, and their OPRF values $OPRF_M(y) = H_2(A_1[v[1]] || \dots || A_w[v[w]])$. For each $y \in \Psi$, outputs y .

3. Our MPSI Protocol

Assume that there are t parties as $P = \{P_1, \dots, P_t\}$ each of which has a private data set X_i of size n_i , respectively, and a trusted Dealer D who helps parties to compute the intersection. Although any party can compute the intersection, for simplicity, we consider P_1 as the server who computes and outputs the intersection.

They agree on parameters $\lambda, \sigma, m, w, n, l_1$ and a hash function $H_1: \{0,1\}^* \rightarrow \{0,1\}^{l_1}$, a pseudorandom function $F: \{0,1\}^\lambda || \{0,1\}^{l_1} \rightarrow \{0,1, \dots, m-1\}^w$. The parties generate a secret $k = \{0,1\}^\lambda$ and keep it hidden from the dealer.

1. Precomputation

- Each P_i samples $t-1$ random strings $s = \{0,1\}^w$.
- Each P_i chooses a random matrix A^i of size $m \times w$, each entry of A^i is a random number of

sizes r , where A^i_j denotes the j -th column of A^i , where $1 \leq j \leq w$.

- Each P_i , for each $x_{i,l} \in X_i$, computes $v = F_k(H_1(x_{i,l}))$, where $1 \leq l \leq n_i$, updates $A_c[v[c]] = 0$ for all $c \in [w]$ by keeping rest of the entries the same.
- Each party generates another random matrix B^i , where B^i shares zero entries with A^i (that is, each zero entry in A^i appears in the same place B^i). The rest of the entries in B^i are chosen at random.

2. Zero Sharing

- Each P_i generates $t-1$ matrices of size $m \times w$ called A^{i,r^*} , $1 \leq r < t-1$, and fills with random numbers, does the same thing again, and is named as B^{i,r^*} .
- For each 0 appears in A^i matrix at index (a,b) , the party i does the following:
 - The party i generates t shares that satisfy:

$$0 = sh_1^{a,b} \oplus sh_2^{a,b} \dots \oplus sh_t^{a,b} \quad (3)$$

- P_i keeps the first share to itself by updating $A_a^i[b] = sh_1^{a,b}$ and $B_a^i[b] = sh_1^{a,b}$.

- P_i sets rest of the shares to A^{i,r^*} and B^{i,r^*} at index (a,b) respectively to be used in Oblivious Transfer later.

3. Oblivious Transfer

- Each P_i does OT with the rest $P \setminus P_i$'s.
- The OT interaction, while P_i will be taking the role of the sender, the rest $P_r \in P \setminus P_i$ takes the role of the receiver, happens as follows:
 - Party P_r uses $s_{i,c}[1], \dots, s_{i,c}[w]$, while P_i has the column vectors (A^{i,r^*}, B^{i,r^*}) as inputs. Here, we have (A^{i,r^*}, B^{i,r^*}) , these matrices are composed of the shares of entries of (A^i, B^i) . Of course every (A^{i,r^*}, B^{i,r^*}) is different for different P_r 's (which denotes the shares of entries to be sent to P_r).

-Now, each party has a matrix $C_{i,r}$ from their OT interactions. Namely, P_i has $C_{i,r}$ whose columns are either from A^{i,r^*} or B^{i,r^*} , where $r \neq i$ and has $C_{i,i} = B^{i,i^*}$ consisting of the shares holding for himself. That is, for example, P_1 keeps $sh_1^{a,b}$ for himself, sends other sh's to the other parties where

$$0 = sh_1^{a,b} \oplus sh_2^{a,b} \dots \oplus sh_t^{a,b} \quad (4)$$

- At the end each P_i element-wise XOR their obtained matrices and A^i to construct C_i , after computation each P_i sends C_i to the Dealer, where

$$C_i = A^i \oplus C_{i,j} \oplus C_{i,j+1} \oplus \dots \oplus C_{i,t-1} \quad (5)$$

4. Dealer Side and Requests

- The trusted dealer gathers all C 's and XOR them to construct the final C by

$$C = C_1 \oplus C_2 \oplus \dots \oplus C_t. \quad (6)$$

- Any party can act as a server and check an item is whether inside the intersection or not by sending previously calculated $v = F_k(H_1(x_{i,l}))$ values to the Dealer. For simplicity, we think P_1 acts like a server.

- The Dealer D checks $C_c[v[c]] = 0$ for all $c \in [w]$, if it encounters an entry other than 0 it means the item is not inside the intersection. On the other hand, if Dealer D checks that all items are 0, that means the item is inside the intersection. In the end, Dealer D sends the output vector (d_1, \dots, d_n) with the same order to Server P_1 where bits represent an item whether inside the intersection or not.

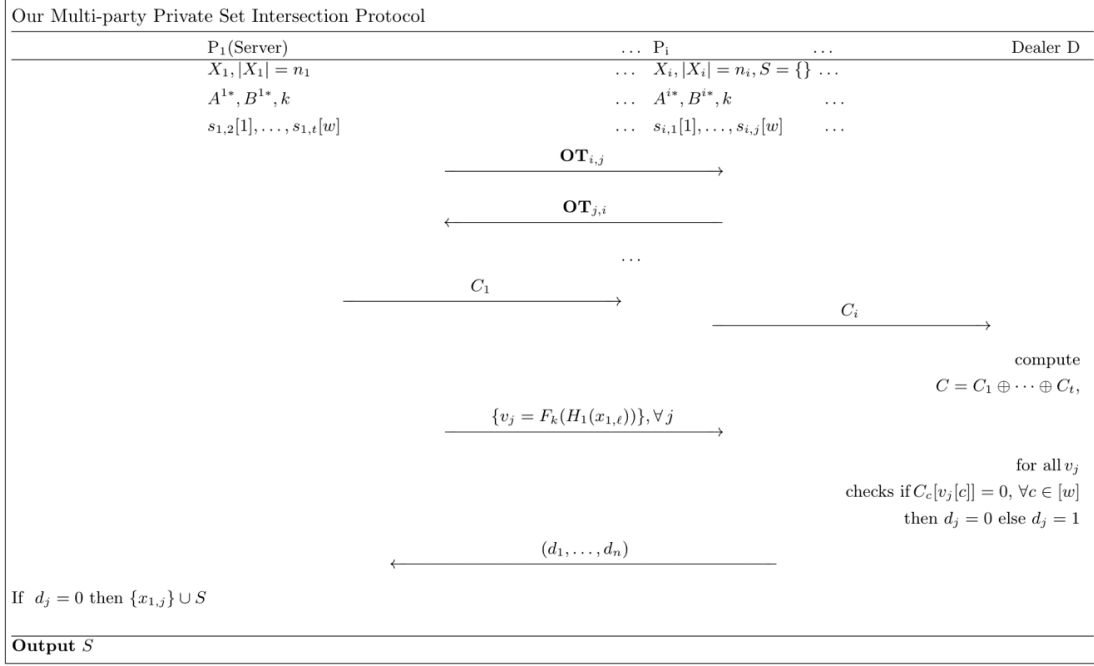


Figure 1: Our Multi-party private set intersection protocol when only P_1 outputs the intersection.

4. Complexity Analysis and Security of Our Mpsi Protocol

4.1 Communication Complexity

There are two major contributions to the communication complexity which are one from OT and the other from communications with the server.

-There is $t(t - 1)$ number of OT interactions executed during this protocol. Each party receives a matrix from all other members. These interactions include $m * w$ matrix and matrix and if there the shares have length of r bits, the communication cost from OT will be $(t - 1) * m * w * r$.

-Each party sends its matrix C_i of size $m * w$ to the server to construct the final C . Same as before, if there are the shares have a length of r bits, the communication cost from communications with the server will be $m * w * r$.

In the case that all the parties compute the intersection, then for each party, the communicational complexity will be $t * m * w * r$.

Choosing m and w play a crucial role in our construction in the sense of security leaks and communication overhead. As in [3], we set row size(m) to number of data(n). Also, we fix column size(w) to statistical parameter λ for all parties. We refer to reader the Miao's statistical explanations of how to choose m and w [3,30]. This makes our complexity $t * n * \lambda * r$.

4.2. Computational Complexity

In the precomputation phase, for every element x in each data set, each party has to compute $v = F_k(H_1(x))$ which requires n calls of F function, where $n = \max \{n_i\}$. Each party creates shares for at most $m * w$ entries. Then, each party has to do $2(t - 1)$ OT interactions. Finally, each party does $(t - 1)$ matrix addition. The server makes t matrix addition.

4.3. Security

Our protocol is secure according to the following Theorem.

Theorem 5.1: If F is a PRF, H is modelled as a random oracle, and the underlying OT protocol is secure, for any

coalition of fewer than t clients including the server, our MPSI has semi-honest security against an honest-but-curious adversary.

The security of the protocol is due to the use of OT and the zero-secret sharing protocol, as long as the parties and the Dealer D follow the instruction of the protocol which is aligned with the semi-honest security, no one gets any information rather than the intersection. The full proof will be provided later in the full version of the paper.

5. Final Remarks and Comparison with Other Protocols

5.1. Remarks

Remark 1: Our multi-party private set intersection protocol relies on computationally fast primitives such as hashing, oblivious transfers based on symmetric keys, and bitwise-XOR operations.

Remark 2: As the number of parties participating in the protocol increases, the communication and computational complexity increases linearly for each party. This makes our protocol scale well particularly when there is a large number of participants in the setup.

Remark 3: The bottleneck of our protocol is the mesh topology that arises from its design. To execute the oblivious transfer section, every party needs to communicate with the rest of the parties. The authors in [29] proposed a multi-party PSI that is an extension of the work of Chase and Miao’s [3] private set intersection protocol. They employed Garbled Bloom Filters to their protocol and this way parties interact with path-like communication. Naturally, from the communication perspective path-like topology is better than mesh topology.

5.2. Comparison

Table 1: Comparison with the other protocols

Protocol	Communication		Computation	
	#bits		#operations	
	Server	Client	Server	Client
[30]	$\mathcal{O}(\lambda t^2 n)$		$\mathcal{O}(tn)$	
[22]	$\mathcal{O}(tn\lambda)$	$\mathcal{O}(n\lambda)$	$\mathcal{O}(tn\log(n))$	$\mathcal{O}(n)$
[2]	$\mathcal{O}(tn\lambda)$	$\mathcal{O}(n\lambda)$	$\mathcal{O}(tk)$	$\mathcal{O}(lk)$
[17]	$\mathcal{O}(tnkk)$	$\mathcal{O}(tnkk)$	$\mathcal{O}(tnkk)$	$\mathcal{O}(tnkk)$
Sec. 3	$\mathcal{O}(tn\lambda r)$	$\mathcal{O}(tn\lambda r)$	<i>Negl</i>	$\mathcal{O}(t\lambda)^*$

* In our construction, computational complexity heavily depends on OT interactions. Oblivious transfers are cryptographic primitives introduced by Rabin[26]. It relies on public-key operations and this makes OTs computationally expensive. However, Ishai [31] proposed a method (OT-extension) that you can do a number of oblivious transfers by using a few public-key operations.

6. Implementation

We implemented and tested our protocol in Python in a primary way. Our implementation confirms our

theoretical approach logically works. In order to do a benchmark comparison with other protocols which have already been implemented and are publicly available, we plan to implement our code in C++ in the near future. In this way, we can use the Oblivious Transfer Protocol developed by Peter Rindal [32] where symmetric primitives are used. For concrete analysis, the full code will be available later.

7. Conclusion

In this paper, a novel MPSI protocol based on OPRFs and the zero-secret sharing scheme is proposed. The key idea of the construction is to extend OPRF-based Chase-Miao’s PSI protocol [3] to multi parties. That is to say, we take OPRFs idea that is used in two-party Chase-Miao’s PSI protocol and use it in a multi-party set intersection while the zero-secret sharing protects the privacy of the parties. Also, the developed construction uses fast primitives such as hashing, oblivious transfers based on symmetric keys, and bitwise-XOR operations. These primitives make the computational and communication complexities of our protocol efficient which are both linear in the number of parties ($t\lambda$ and $tn\lambda r$ respectively). As future work, we plan to do benchmark implementation of the scheme in C++ and remove the Trusted Dealer D safely by keeping the parties’ input secure.

8. References

- [1] Pinkas, B., Schneider, T. and Zohner, M., “Faster private set intersection based on {OT} extension”, *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, 797-812.
- [2] Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., and Trieu, N., “Practical multi-party private set intersection from symmetric-key techniques”, *2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, 1257-1272.
- [3] Chase, M., and Miao, P., “Private set intersection in the internet setting from lightweight oblivious PRF”, *Annual International Cryptology Conference*, 2020, 34-63.
- [4] Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N., “Efficient batched oblivious PRF with applications to private set intersection”, *2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, 818-829.
- [5] Pinkas, B., Schneider, T., Segev, G., and Zohner, M., “Phasing: Private set intersection using permutation-based hashing”, *24th USENIX Security Symposium (USENIX Security 15)*, 2015, 515-530.
- [6] Pinkas, B., Rosulek, M., Trieu, N. and Yanai, A., “PSI from PaXoS: fast, malicious private set intersection”, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2020, 739-767.
- [7] Trieu, N., Shehata, K., Saxena, P., Shokri, R. and Song, D., “Epione: Lightweight contact tracing with strong privacy”. arXiv preprint arXiv:2004.13293., 2020.
- [8] Thomas, K., Pullman, J., Yeo, K., Raghunathan, A., Kelley, P. G., Invernizzi, L., ... and Bursztein, E., “Protecting accounts from credential stuffing with password breach

- alerting”, *28th USENIX Security Symposium (USENIX Security 19)*, 2019, 1556-1571.
- [9] Internet: K. Opsahl, R. Reitman, The Disconcerting Details: How Facebook Teams Up With Data Brokers to Show You Targeted Ads, <https://www.eff.org/deeplinks/2013/04/disconcerting-details-how-facebook-teams-data-brokers-show-you-targeted-ads>, 03.02.2022
- [10] Shen, L., Chen, X., Wang, D., Fang, B. and Dong, Y., “Efficient and private set intersection of human genomes”, *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2018, 761-764.
- [11] Freedman, M. J., Nissim, K. and Pinkas, B., “Efficient private matching and set intersection”. *International Conference on the Theory and Applications of Cryptographic Techniques*, 2004, 1-19.
- [12] De Cristofaro, E. and Tsudik, G., “Experimenting with fast private set intersection”, *International Conference on Trust and Trustworthy*, 2012, 55-73.
- [13] Sang, Y. and Shen, H., “Privacy preserving set intersection based on bilinear groups”, *The Thirty-first Australasian conference on Computer science*, 2008, 47-54.
- [14] Huang, Y., Evans, D. and Katz, J., “Private set intersection: Are garbled circuits better than custom protocols?”, *NDSS. 19th Annual Network & Distributed System Security Symposium*, 2012.
- [15] Yao, A. C. C., “How to generate and exchange secrets”, *27th Annual Symposium on Foundations of Computer Science*, 1986, 162-167.
- [16] Kiss, Á., Liu, J., Schneider, T., Asokan, N. and Pinkas, B., “Private Set Intersection for Unequal Set Sizes with Mobile Applications”, *Proceedings on Privacy Enhancing Technologies*, Vol. 4, 177-197, 2017.
- [17] Inbar, R., Omri, E. and Pinkas, B., “Efficient scalable multiparty private set-intersection via garbled bloom filters”. *International Conference on Security and Cryptography for Networks*, 2018, 235-252.
- [18] Debnath, S. K. and Dutta, R., “Secure and efficient private set intersection cardinality using bloom filter”, *International Conference on Information Security*, 2015, 209-226.
- [19] Pinkas, B., Schneider, T., Tkachenko, O. and Yanai, A., “Efficient circuit-based PSI with linear communication”, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2019, 122-153.
- [20] Freedman, M. J., Nissim, K. and Pinkas, B., “Efficient private matching and set intersection”. *International Conference on the Theory and Applications of Cryptographic Techniques*, 2004, 1-19.
- [21] Kissner, L. and Song, D., “Privacy-preserving set operations”, *Annual International Cryptology Conference*, 2005, 241-257.
- [22] Hazay, C. and Venkatasubramanian, M., “Scalable multi-party private set-intersection”. *IACR International Workshop on Public Key Cryptography*, 2017, 175-203.
- [23] Goldreich O., “Secure multi-party computation”, Manuscript. Preliminary version 78, 1998.
- [24] Miyaji, A., Nakasho, K. and Nishida, S., “Privacy-preserving integration of medical data”, *Journal of Medical Systems*, Vol. 41(3), 1-10, 2017.
- [25] Binu V. P. and Sreekumar A., “Simple and efficient secret sharing schemes for sharing data and image.”, *International Journal of Computer Science and Information Technologies*, Vol. 6 (1), 404-409, 2015.
- [26] M. O. Rabin, “How To Exchange Secrets with Oblivious Transfer.” IACR Eprint archive 2005/187, 2005.
- [27] Kolesnikov, V. and Kumaresan, R., “Improved OT extension for transferring short secrets”, *Annual Cryptology Conference*, 2013, 54-70.
- [28] Pinkas, B., Rosulek, M., Trieu, N. and Yanai, A., “SpOT-light: lightweight private set intersection from sparse OT extension”, *Annual International Cryptology Conference*, 2019, 401-431.
- [29] Alireza K., Mohajeri J. and Mahmoud S., “Efficient scalable multi-party private set intersection using oblivious prf”, *International Workshop on Security and Trust Management*, 2021, 81-99.
- [30] Cheon, J. H., Jarecki, S. and Seo, J. H., “Multi-party privacy-preserving set intersection with quasi-linear complexity”, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 95(8), 1366-1378, 2012.
- [31] Ishai, Y., Kilian, J., Nissim, K. and Petrank, E.. “Extending oblivious transfers efficiently”, *Annual International Cryptology Conference*, 2013, 145-161.
- [32] Internet: P. Rindal, A fast, portable, and easy to use Oblivious Transfer Library, <https://github.com/osu-crypto/libOTe>, 01.02.2022.