Efficient Scalable Multi-Party Private Set Intersection Using Oblivious PRF

Alireza Kavousi, Javad Mohajeri, and Mahmoud Salmasizadeh

[1] Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
`kavousi.alireza@ee.sharif.edu`

[2] Electronics Research Institute, Sharif University of Technology, Tehran, Iran
{`mohajer,salmasi`}`@shairf.edu`

Abstract. In this paper, we present a concretely efficient protocol for private set intersection (PSI) in the multi-party setting using oblivious pseudorandom function (OPRF). In fact, we generalize the approach used in the work of Chase and Miao [CRYPTO 2020] towards deploying a lightweight multi-point OPRF construction for two-party PSI. Our protocol only includes oblivious transfer (OT) extension and garbled Bloom filter as its main ingredients and avoids computationally expensive operations. From a communication pattern perspective, the protocol consists of two types of interactions. The first type is performed over a star-like communication graph in which one designated party interacts with all other parties via performing OTs as the sender. Besides, parties communicate through a path-like communication graph that involves sending a garbled Bloom filter from the first party to its neighboring party following the last one. This design makes our protocol to be highly scalable due to the independence of each party's complexity from the number of participating parties and thus causes a communication and computation complexities of $O(n\lambda k)$, where n is the set size, k is the number of hash functions, and λ is the security parameter. Moreover, the asymptotic complexity of the designated party is $O(tn\lambda)$ which linearly scales with the number of parties t. We prove security of the proposed protocol against semi-honest adversaries.

Keywords: Secure Multi-Party Computation · Private Set Intersection · Oblivious Pseudorandom Function · Concrete Efficiency.

1 Introduction

Secure multi-party computation (MPC) has been the focus of an extensive amount of scientific works over the last few decades. It deals with the general problem of enabling a group of distrustful parties to jointly compute a function of their private inputs without revealing anything but the result. Due to the considerable progress in making the MPC protocols more and more efficient, they have become truly practical and therefore found much more applications in recent years.

Private set intersection (PSI) is one of the important and well-studied MPC protocols which allows a set of parties, each holding an input set, to compute their intersection without leaking any other information beyond their intersection. There exist many privacy-preserving potential applications for PSI such as advertising conversion [Mia+20], private contact discovery [Kal+19; Dem+18], and more. Recently and due to the spread of the COVID-19 pandemic, there has been an interdisciplinary quest to develop private contact tracing systems to contain the outbreak. In this case, PSI also plays a crucial role in building privacy-preserving solutions [DPT20; Dit+20].

During the last decade or so, many research-oriented works have been dedicated to proposing efficient constructions for the PSI functionality. In general, there are two main approaches to the design of these constructions. The first is using *generic circuit-based* protocols that deal with the computation of logical or arithmetic circuits by parties [HEK12; Pin+18; Pin+19b]. Although circuit-based protocols often yield computationally efficient constructions and are flexible to be adapted for different variants of PSI functionality, having high communication complexity for the PSI problem which requires the evaluation of large circuits is a big hurdle in making them to be practically useful. It should be remarked that it is now widely believed that communication and not computation, is the principal bottleneck in MPC protocols like PSI [Ash+13; Hal18]. Another approach is related to *special purpose* protocols that mainly rely on cryptographic primitives and various assumptions. Since this type of PSI protocols can achieve better performance compared to the previous one, it has gained significant attention among researchers.

Loosely speaking, existing special purpose PSI protocols can be categorized in the following way. PSI protocols built from oblivious polynomial evaluation [FNP04; HV17; Haz18; GS19], hard cryptographic assumptions [DT10; DT12], and oblivious transfer (OT) and hashing structures [Kol+16; Pin+20; CM20]. There has also been a branch of works on server-aided setting [KMS20; ATD20]. Since the OT-based PSI protocols achieve a good balance between communication and computation costs and indeed mainly benefit from cheap cryptographic tools, are often regarded as the fastest *concretely efficient* solutions in which by this term we refer to those constructions which do not use computationally too expensive tasks like polynomial evaluation and interpolation or vast public-key operations (see [PSZ18] for an overview on different PSI settings).[1]

Notice that a large body of literature on two-party OT-based PSI uses a primitive named *oblivious pseudorandom function* (OPRF) which is often instantiated efficiently by means of symmetric-key techniques. Particularly, the recent work of Chase and Miao [CM20] aims to investigate the trade-offs between communication and computation costs and enjoy the best of both worlds. By introducing an interesting lightweight *multi-point* OPRF protocol, they propose a highly efficient semi-honest secure two-party PSI protocol that assuming

[1] Although to perform OT one needs to use public-key operations, in [Ish+03] a method was introduced which enables to do quite a large number of OTs utilizing only efficient symmetric-key primitives.

random oracle model its security can be enhanced to one-sided malicious security. The idea of considering a multi-point OPRF construction instead of the common single-point version results in decreasing the communication complexity of the protocol by a constant factor due to the fact that evaluation of each element in the set will be required only for once.

Multi-Party PSI. While two-party setting encompasses the majority of existing works, multi-party PSI has not attracted that much attention in the literature. This might account for the fact that there is a seemingly inevitable need to have interactions among parties which incurs an extreme communication cost to the protocol and hence makes it practically infeasible. More recently, however, few works including [Kol+17; HV17; IOP18; GN19] have come up with asymptotically efficient constructions for multi-party PSI in different security models. Regardless of the tools and primitives used, the core idea underlying all these constructions is considering a *designated* party who individually interacts with all other parties throughout the protocol execution (i.e., star topology network). This attitude towards multi-party PSI protocol appears to be useful since it results in a reduction in *intermediate* exchanges between parties but has the weakness of putting a high workload on the designated party which may be very problematic in practical scenarios. Very recently, the approaches of [Kol+17; IOP18], which led to concretely efficient constructions are extended by [Efr+21] in a maliciously secure model.

Additional Related Work. The authors in [DCW13] present a two-party OT-based PSI protocol using a variant of Bloom filter called garbled Bloom filter. [RR16] follows the approach of the aforementioned protocol and presents a maliciously secure protocol employing the cut-and-choose technique. A few constructions like [Yin+20; Bud+20] concentrate on some variants of PSI in which they study the problem of performing different sets of computations on the intersection. There are also a few works on threshold PSI such as [ZC18; GS19; Bad+20; BDP20]. In [ZC18], authors introduce a protocol based on oblivious polynomial evaluation for threshold PSI. The exciting work of [GS19] demonstrates a lower bound on the communication complexity of two-party threshold PSI. The most recent work of [Bad+20] takes this a stage further and extends the results to the multi-party setting.

1.1 Our Contribution

We study the problem of PSI in the case that there are more than two parties involved in the execution of the protocol, namely *multi-party* PSI. When practicality comes into play, most of the current protocols on multi-party PSI fail to meet the need because of suffering from either high communication or computational overhead for the considerable number of participating parties or the large set sizes. In this work, we aim to present a concretely efficient multi-party PSI protocol following the idea of [CM20] in employing an efficient multi-point

OPRF construction that through leading to a better balance between communication and computation costs causes [CM20] to be the fastest two-party PSI in moderate bandwidth compared to the state-of-the-art protocols.

Our protocol leverages a combination of so-called *star* and *path* communication graphs which in the former, a designated party as the sender runs OTs with all other parties, and in the latter, each party only sends a garbled Bloom filter to his adjacent party in the direction of the last party. In light of this design, the construction can be very scalable since the communication and computation complexities of each party (except the designated party) only depend on his own input set size and not on the number of parties involved in the protocol. So, while the designated party has the asymptotic complexity of $O(tn\lambda)$ which linearly scales with the number of parties t, the complexity of each other party is $O(n\lambda k)$ where n is the party's set size, k is the number of hash functions (used in garbled Bloom filter), and λ is the security parameter. Also, thanks to this fusion of star and path communication graphs, instead of having a designated party with significantly high communication overhead compared to others, the distribution of cost is rather fair with respect to the number of parties t and the number of hash functions k and therefore this prevents the designated party from taking a lot of bandwidth. We consider semi-honest security and prove the security of our protocol in this model.

2 Preliminaries

2.1 Notation

Throughout this paper, we consider t parties $P_1, \dots, P_t$ who each owns an input set $X_1, \dots, X_t$, respectively. We may refer to P_t as the leader and all the other parties as clients. λ and σ are used to denote the computational and statistical security parameters which the former deals with the hardness of problems in the face of computationally bounded adversaries and the latter is concerned with the attacks that may occur during protocol interactions. $[n]$ concisely shows a set of n items $\{1, \dots, n\}$. By $v[i]$, we refer to the i-th element of the vector v. In an $n \times m$ matrix M, the i-th column is denoted as M_i where $i \in [m]$. $\|x\|$ denotes the hamming weight of a string x. We consider $\mathrm{negl}(\lambda)$ as a negligible function that proceeds asymptotically towards zero faster than any inverse polynomial for appropriately large inputs. Finally, we use $s \xleftarrow{R} S$ to show that s is sampled uniformly at random from S.

2.2 Secret Sharing Scheme

Secret sharing [Sha79] is one of the pivotal tools in cryptography which has numerous applications in constructing secure computation protocols. In an (t, n) secret sharing scheme, a secret s is distributed among n parties in a way that by having up to $t-1$ shares no information about the secret is revealed. The simplest form of an (n, n) secret sharing scheme can be achieved by means of bitwise-XOR operation. In fact, one chooses $n-1$ random strings $(v_1, \ldots, v_{n-1})$, and then selects the last share by computing $v_n = s \oplus v_1 \oplus \ldots \oplus v_{n-1}$. This scheme has perfect security and for reconstructing the secret s having all shares are required.

2.3 Bloom Filter

Bloom filter (BF) [Blo70] is a probabilistic compact data structure which is used as a tool for efficient set membership checking. It randomly maps a set X containing n items to a binary array of size M, where every single element is mapped to a different subset of indices in the array. Bloom filter includes k independent uniform hash functions $H = \{h_1, \ldots, h_k\}$ that $h_i : \{0,1\}^* \rightarrow [M]$. At first, all bits in the array are set to zero. To insert each element $x \in X$, one sets $BF[h_i(x)] = 1$ for all $i \in [k]$. To see whether the set X consists of an element x', one simply needs to check all the $BF[h_i(x')]$ are equal to one. Even if one of the corresponding bits in the array be equal to zero, then it can be concluded that the element x' is not in the set. On the other hand, if all of the corresponding bits in the array are equal to one, then x' is in the set but for a determined false-positive probability ϵ. The computed upper bound on ϵ is given by $p^k(1 + O(\frac{k}{p}\sqrt{\frac{\ln M - k \ln p}{M}}))$ where $p = 1 - (1 - \frac{1}{M})^{nk}$. It is shown in [DCW13] that the optimal values to accomplish the best performance are $k = \frac{M}{n} \ln 2$ and $M \geq n \log_2(e). \log_2(\frac{1}{\epsilon})$.

Garbled Bloom Filter. A different version of BF was introduced in [DCW13] which is called garbled Bloom Filter (GBF). To give a concise description of GBF, we can refer to it as an extended Bloom filter that instead of having an array consisting of single bits, it is an array consisting of bit strings where the length of the bit string is determined by security parameter. Like BF, insertion is done in GBF by computing the hash functions for each input element x with regard to a set of uniform hash functions $H = \{h_1, \ldots, h_k\}$, but instead of dealing with just single bits, some randomly chosen shares of x are placed in those indices corresponded to x subject to the constraint that $\bigoplus_{i=1}^{k} GBF[h_i(x)] = x$. Garbled property of GBF makes it computationally impossible to know whether a given element x is in the set, unless one queries GBF on all the indices related to x. In this manner, the false-positive probability in GBF is equal to $2^{-\lambda}$.

2.4 Oblivious Transfer

A foundational cryptographic primitive used as a building block in many secure computation protocols is oblivious transfer (OT) [Rab05] whose functionality is presented in Figure 1. In an 1-out-of-2 OT, there exist a sender and a receiver, where the sender has two strings (x_0, x_1) and the receiver has a choice bit c as their inputs, respectively. After the execution of OT, the sender learns nothing and the receiver learns x_c without obtaining any information about x_{1-c}. As shown in [IR89], it cannot be possible to do OT without relying on public-key operations. Thus, this was considered as the main constraint when it comes to

Input: The sender inputs two strings (x_0, x_1) and the receiver inputs a choice bit $c \in \{0, 1\}$.
Output: The functionality returns x_c to the receiver and returns nothing to the sender.

Fig. 1: The functionality of oblivious transfer ($\mathcal{F}_{OT}$)

doing a great number of OTs which is a typical task in PSI protocols. However, [Ish+03] proposed a method called OT extension which makes it possible to do an extensive number of OTs while using only a limited number of public-key operations for initial OTs (known as base-OTs). Also, some variants of OT are available. Random OT (ROT) refers to a setting in which the sender and receiver do not choose their inputs and they are chosen by the functionality itself. By using ROT, the protocol can be performed with much less communication overhead compared to OT.

2.5 Security Model

Definition 1. (Computational Indisguishability) *Let $X = \{X(\lambda)\}_{\lambda \in N}$ and $Y = \{Y(\lambda)\}_{\lambda \in N}$ be two probability distribution ensembles, we say that X and Y are computationally indistinguishable, denoted as $X \approx Y$, if for every probabilistic polynomial time (PPT) algorithm D, there exists a negligible function negl(λ) such that for all sufficiently large λ*

$$|Pr[D(\lambda, X(\lambda))] - Pr[D(\lambda, Y(\lambda))]| \leq negl(\lambda).$$

Our protocol is secure against semi-honest adversaries who follow the protocol as specified but try to obtain more information than what is allowed. Note that we assume the leader does not collude with any client. This assumption is widely used in the literature [Aba+17; Zha+19]. Having this in mind, our protocol can tolerate up to $t - 1$ corruptions.

The security of an MPC protocol is typically proven respecting real/ideal simulation paradigm. That is, a protocol is considered to be secure if the real execution of the protocol Π computationally looks like the ideal execution of the protocol $\mathcal{F}$. To put in another way, imagine an ideal world where there exists a fully trusted entity that parties can privately send their inputs to and then it computes the result and returns it back to the parties. Surely, this ideal world execution captures all the required security we want. So, if we somehow show that for any real world adversary, the real and ideal execution of the protocol are computationally indistinguishable, then we can deduce the protocol Π is secure. Here, we give the formal definition of real/ideal simulation paradigm for two-party protocol introduced in [Gol04].

Definition 2. (Semi-Honest Security) *Let denote P_i's view in the real execution of the protocol Π as $view_i^{\Pi}(X_1, X_2)$ for $i \in [2]$. $\mathcal{F}_i(X_1, X_2)$ denotes the output of ideal functionality for P_i. The protocol Π securely realizes the ideal functionality*

$\mathcal{F}$ in the presence of static semi-honest adversaries if there exist PPT simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ for all inputs such that

$$\{\mathcal{S}_1(\lambda, X_1, \mathcal{F}_1(X_1, X_2))\} \approx \{view_1^{\Pi}(X_1, X_2))\},$$

$$\{\mathcal{S}_2(\lambda, X_2, \mathcal{F}_2(X_1, X_2))\} \approx \{view_2^{\Pi}(X_1, X_2))\}.$$

2.6 Hamming Correlation Robustness

The security of some protocols can be proven using a weaker assumption than random oracle model which is called *correlation robustness* [KK13; Pin+19a; CM20]. In this paper we use the definition presented in [Pin+19a; CM20] to prove the security of our protocol.

Definition 3. (Hamming Correlation Robustness) *Let H be a hash function with the input length n. Then H is d-Hamming correlation robust if, for any $a_1, \ldots, a_m, b_1, \ldots, b_m \in \{0,1\}^n$ with $\|b_i\| \geq d = \lambda$ for each $i \in [m]$, the following distribution, induced by random sampling of $s \overset{R}{\leftarrow} \{0,1\}^n$, is pseudorandom.*

$$H(a_1 \oplus [b_1 \cdot s]), \ldots, H(a_m \oplus [b_m \cdot s]),$$

where $\cdot$ denotes bitwise-AND.

2.7 PSI From OPRF

An oblivious pseudorandom function (OPRF) is a secure two-party computation protocol which was introduced in [Fre+05]. In an OPRF protocol, the sender inputs a random PRF key K and the receiver inputs a single input x. By the end of the protocol, the sender learns nothing and the receiver learns the evaluation of the OPRF functionality on his input.

There are several works on two-party PSI protocol which use single-point OPRF construction [Pin+15; Kol+16]. At a high level, the general structure of these protocols is as follows. Firstly, the sender (P_1) and the receiver (P_2) run the OPRF protocol that at the end P_1 obtains a random key K and P_2 obtains the outcome of the functionality on his input $\text{OPRF}_K(x_1^2)$. They run the protocol for all items in the receiver's set $x_1^2, \ldots, x_{n_2}^2 \in X_2$. As a result, the sender learns a set of random keys and the receiver learns a set of OPRF values. Then, P_1 evaluates OPRF functionality on his set of inputs $x_1^1, \ldots, x_{n_1}^1 \in X_1$ and sends the resulting values to P_2. We should note that in these protocols parties often use a Cuckoo hashing construction [PR04] to map every single of their elements to a separate bin and it is also assumed parties' sets have the same size $n_1 = n_2 = n$. Finally and by comparing the received values and his OPRFs, P_2 can determine the intersection.

From Single-Point to Multi-Point OPRF. In [Kol+16], a hash function (which is modeled as a random oracle) is considered an OPRF whose keys are in fact parts of its input argument. Pinkas et al. in [Pin+19a] proposed a PSI protocol based on multi-point OPRF construction in which it enables parties to instead of having to evaluate n instances of OPRF protocol, do it in a way that computing OPRF values requires far less communication cost, i.e., they no longer need to use Cuckoo hashing and perform several OPRFs for every single hash bin, so each element is only evaluated once. But, multi-point OPRF of [Pin+19a] incurs a high computational overhead compared to the single-point version of [Kol+16], since it needs to evaluate and interpolate a high-degree polynomial over a large field which obviously causes much more cost than just using symmetric primitives and bitwise operations as in [Kol+16].

Efficient Multi-Point OPRF. To provide a more reasonable balance between communication and computation costs, Chase and Miao [CM20] introduced a two-party PSI protocol using a lightweight multi-point OPRF where oblivious transfer protocol is the only heavy cryptographic operation needed that also itself can be performed efficiently using OT extension. To perform the multi-point OPRF, a random seed of length w is picked by the sender, $s \overset{R}{\leftarrow} \{0,1\}^w$, and the receiver constructs two $m \times w$ matrices of A and B where the entries of the former are selected randomly from $\{0,1\}$ and those of the latter are determined by evaluating a pseudorandom function with the output length of $w \cdot \log m$ on each element of the receiver's set, $v = F_K(x_i^2)$. The matrix B is formed such that for every $x_i^2 \in X_2$, the corresponding bits in two matrices are the same while other bits differ. After running w OTs between parties, the sender who acts as a receiver obtains an $m \times w$ matrix C that each of its columns is either A_i or B_i for all $i \in [w]$ depending on the chosen seed s. Then, the sender evaluates the PRF on each of his input elements $x_i^1 \in X_1$ as $v = F_K(x_i^1)$ and computes the OPRF value $\psi = H(C_1[v[1]] \parallel \ldots \parallel C_w[v[w]])$ and sends all the resulting OPRFs to the receiver. Ultimately, the receiver computes the OPRF values of his input elements and finds the intersection of the two sets. Notice that if a sender's element be in the intersection, $x_i^1 \in X_2$, its corresponding input to the OPRF is equal to one of the receiver's element input to the OPRF, otherwise the inputs to OPRF are different with overwhelming probability.

3 Our Multi-Party PSI Protocol

3.1 An Overview

In this section we introduce our proposed multi-party PSI protocol. As mentioned earlier, there is a group of parties $P_1, \ldots, P_t$ with their private input sets $X_1, \ldots, X_t$, respectively, who want to jointly compute their set intersection $X_1 \cap \ldots \cap X_t$ without leaking any other private information relating to either individual or a proper subset of parties. As in many other multi-party protocols, we consider P_t as the party who learns the intersection at the end of the protocol. The functionality of multi-party PSI is defined in Figure 2. We use the lightweight multi-point OPRF construction introduced in [CM20] to build an efficient and scalable multi-party PSI. The full description of our protocol which constitutes several steps is presented in Figure 3.

There are t parties $P_1, \ldots, P_t$.
Input: Each party P_j has an input set $X_j = \{x_1^j, \ldots, x_{n_j}^j\}$ where every $x^j \in \{0,1\}^*$.
Output: Party P_t receives the intersection $I = X_1 \cap \ldots \cap X_t$ and other parties receive nothing.

Fig. 2: The functionality of multi-party private set intersection ($\mathcal{F}_{\text{MPSI}}$)

Generally speaking, the protocol works as follows. At first, P_t constructs a random $m \times w$ matrix A. In fact, to generate the i-th column of matrix A, P_t chooses $t-1$ strings of length m uniformly at random and sets $A_i = A_i^1 \oplus \ldots \oplus A_i^{t-1}$. In addition, for each $j \in [t-1]$, party P_t generates the matrix B^j from the matrix A^j by computing a pseudorandom function $F_K(\cdot)$ on all of his input elements and sets $B = B^1 \oplus \ldots \oplus B^{t-1}$. After running w OTs between P_t as the sender and each $\{P_j\}_{j\in[t-1]}$ as the receiver, every P_j ends up with a matrix C^j which its column vectors are just m-bit random strings. Then, each party locally constructs a garbled Bloom filter of his input set GBF_j using the entries of the received matrix. Afterwards, P_1 sends GBF_1 to P_2 that upon getting it, he XORs GBF_1 with GBF_2 and sends the resulting GBF to the next party. This process continues until P_{t-1} computes the cumulative GBF and also OPRF values and then sends the OPRFs to the P_t to allow him to find the intersection.

Remark 1. We can consider an upper bound N on each party's input set size. Meaning, parties $P_1, \ldots, P_t$ can have different input set sizes up to N. In this way, parties' exact set sizes would not be revealed during the execution of the protocol.

Remark 2. We assume that clients are connected by secure channels, i.e., party P_t is not able to learn useful information by observing communication between $P_1, \ldots, P_{t-1}$. We stress that deploying such point-to-point channels is cheap and does not impose that much cost.

3.2 Protocol Correctness

Regarding the particular form of matrices A^j and B^j constructed by P_t, for each $x^t \in X_t$, let $v = F_K(H_1(x^t))$, it holds that $A_i^j[v[i]] = B_i^j[v[i]]$ for all $i \in [w]$. Let x be an element which is in the intersection, i.e., it exists in all the parties' input sets. Since P_t inputs uniformly random shares of each column of matrix A (using XOR secret sharing scheme) while performing OTs with clients, for each $x \in I$ it holds that $A_i[v[i]] = \bigoplus_{j=1}^{t-1} C_i^j[v[i]]$, for all $i \in [w]$. Therefore, regardless of what random string s_j is chosen by the client P_j, XORing the strings at all coordinates corresponded to x in GBF^* by P_{t-1} (Step 8) results in a string which is the same as one of the P_t's elements input to the hash function H_2. The correctness of the protocol is satisfied with all but negligible probability of a false-positive occurring.

Parameters: Parties $P_1, \ldots, P_t$ agree on security parameters λ and σ, two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $H_2 : \{0,1\}^w \to \{0,1\}^{l_2}$, and a pseudorandom function $F : \{0,1\}^\lambda \times \{0,1\}^{l_1} \to \{m\}^w$. They also agree on a garbled Bloom filter specification which includes a set of independent random hash functions $H = \{h_1, \ldots, h_k\}$ that $h_i : \{0,1\}^* \to [M]$ for $i \in [k]$, and entries string length of w.

Initial Computation:

1. Each party P_j picks a random string $s_j \overset{R}{\leftarrow} \{0,1\}^w$ for $j \in [t-1]$.
2. P_t generates an $m \times w$ matrix A in which its entries are selected randomly from $\{0,1\}$. For all $i \in [w]$, party P_t chooses $t-1$ shares uniformly at random under the constraint that $A_i = A_i^1 \oplus \ldots \oplus A_i^{t-1}$. He also samples a uniform PRF key $K \overset{R}{\leftarrow} \{0,1\}^\lambda$.

Performing Oblivious Transfer:

3. To construct an $m \times w$ matrix B^j, party P_t computes $v = F_K(H_1(x^t))$ for all $x^t \in X_t$, and copies those bits from the corresponding positions in matrix A^j to the matrix B^j. He also flips the bits from A^j to B^j for the remaining empty positions.
4. At this stage, P_t as the sender with inputs $\{A_i^j, B_i^j\}$ independently runs w OTs with each party P_j as the receiver with inputs $s_j[i]$ for all $i \in [w]$ and $j \in [t-1]$. Eventually, for all $j \in [t-1]$ each party P_j forms an $m \times w$ matrix C^j which its columns are those strings he receives after doing OTs.
5. P_t sends the PRF key K to P_j for all $j \in [t-1]$.

Concluding the Intersection:

6. Each party $\{P_j\}_{j \in [t-1]}$ constructs a garbled Bloom filter of his input set GBF_j such that for every $x^j \in X_j$ it holds that $\bigoplus_{i=1}^k GBF_j[h_i(x^j)]$ equals concatenation of all the bits in positions $C_i^j[v[i]]$ where $v = F_K(H_1(x^j))$.
7. P_1 sends GBF_1 to P_2 that upon receiving it, he XORs GBF_1 and GBF_2 and sends the resulting GBF to the next party. This process continues until P_{t-1} computes the cumulative garbled Bloom filter $GBF^\star = GBF_1 \oplus \ldots \oplus GBF_{t-1}$.
8. For each $x^{t-1} \in X_{t-1}$, P_{t-1} computes $u = \bigoplus_{i=1}^k GBF^*[h_i(x^{t-1})]$ and its OPRF value $\psi = H_2(u)$ and sends it to P_t. Let Ψ denote the set of all OPRFs.
9. After receiving the OPRFs, P_t computes his corresponding OPRF values as $\psi = H_2(A_1[v[1]] \parallel \ldots \parallel A_w[v[w]])$ for all $x^t \in X_t$, where $v = F_K(H_1(x^t))$. P_t considers x^t in the intersection iff $\psi \in \Psi$.

Fig. 3: Our multi-party private set intersection protocol ($\mathit{\Pi}_{\text{MPSI}}$)

3.3 Protocol Security

Security Analysis. In the protocol, P_t runs OTs independently with each client using randomly chosen shares of columns of the matrix A that itself is a random matrix sampled by P_t. So, each matrix C^j formed by the P_j contains independent uniform strings as its columns. As a result, concerning the way each party computes his garbled Bloom filter (Step 7), receiving the GBF of P_i by P_j for any $i, j \in [t-1]$ leaks no useful information about P_i's input set. Also, by suitable choice of the parameters m, w (as will be discussed later) and indeed security properties of GBF, for any item in the P_{t-1}'s input set which is not in the intersection I, the corresponding OPRF value is pseudorandom to P_t. Thus, P_t is only able to obtain intersection over all parties' input sets and learns nothing about partial set intersection (i.e., elements which exist in some but not all parties' input sets).

The parameters m, w. Choosing m and w plays an important role in providing the security of the protocol. The mentioned parameters should be selected in a way that for any common element in clients' sets which is not in the intersection (i.e., $x \in I \backslash X_t$), its OPRF value must be pseudorandom to P_t. In view of this, we need to make sure that if F is a random function and H_1 is a collision resistant hash function then for all $i \in [w]$, there exist at least λ flipped bits in the positions $B_i[v[i]]$, where $v = F_K(H_1(x))$. This is essentially because of fulfilling the correlation robustness property of H_2, and consequently preventing brute force searches by P_t. It should also be noted that for any P_{t-1}'s element which is not in client's intersection, its OPRF value is pseudorandom to P_t due to the obliviousness property of garbled Bloom filter.

Since the input to $F_K(\cdot)$ is different for every $x^t \in X_t$, the probability that any bit in each column of matrix B is flipped equals $p = (1 - \frac{1}{m})^{n_t}$. Thus, for any $x \in I \backslash X_t$, the number of flipped bits in $B_1[v[1]], \ldots, B_w[v[w]]$ has a binomial distribution, which the probability of having d flips is equal to

$$\binom{w}{d} p^d (1-p)^{w-d}.$$

So, by fixing m we can determine the proper value for w using the union bound as follows

$$N \cdot \sum_{d=0}^{\lambda-1} \binom{w}{d} p^d (1-p)^{w-d} \leq negl(\sigma).$$

It is also worth mentioning that the parameter l_2 which is the output length of H_2 needs to be chosen such that the probability of having collision in PSI protocol (Step 9) be negligible. In a similar way to [Pin+19a; CM20], it can be calculated as $l_2 = \sigma + 2\log(N)$ for the semi-honest model.

Security Proof. In this part, we formally prove the security of our proposed multi-party protocol based on the notion of real/ideal simulation paradigm in the semi-honest model. Note that we consider two cases for corruption, in one case adversary corrupts a subset of clients and in the other case only leader is corrupted.

Theorem 1. *Assume that F is a pseudorandom function, H_1 is a collision resistant hash function, and H_2 is a d-Hamming robust hash function, then protocol Π_{MPSI} (Figure 3) securely realizes the functionality $\mathcal{F}_{\mathrm{MPSI}}$ (Figure 2) in the presence of semi-honest adversaries for proper choice of parameters as discussed.*

Proof. (P_t is not corrupted) We show that there exists a PPT simulator $\mathcal{S}_Z$ that given corrupted parties' inputs can generate simulated views which are computationally indistinguishable from joint distribution of corrupted parties' views in the real execution of the protocol. Let us consider a subset Z of parties $P_1, \ldots, P_{t-1}$ is corrupted by the adversary. Given $\{X_j\}_{j \in Z}$, the simulator $\mathcal{S}_Z$ honestly chooses random strings $\{s_j\}_{j \in Z}$ and random matrices $\{C^j\}_{j \in Z} \in \{0,1\}^{m \times w}$. Then, $\mathcal{S}_Z$ runs OT simulator in order to simulate the view of each corrupted party $P_j \in Z$ as the receiver with respect to the inputs $s_j[1], \ldots, s_j[w]$ and outputs $C_1^j, \ldots, C_w^j$. Moreover, $\mathcal{S}_Z$ sends a randomly picked PRF key to the corrupted parties. Knowing the description of garbled Bloom filter, the simulator also constructs random garbled Bloom filters on behalf of the honest parties from its randomness.

We now argue that $\mathcal{S}_Z(\lambda, \{X_j\}_{j \in Z}, \perp) \approx view_Z^{\Pi}(\lambda, X_1, \ldots, X_t)$. To do so, we use a sequence of hybrid distributions in which each two adjacent distributions are computationally indistinguishable and thanks to the transitive property, it can be concluded that the two desired distributions are also computationally indistinguishable.

Hybrid$_0$: The view of corrupted parties $\{P_j\}_{j \in Z}$ in the real execution of the protocol.
Hybrid$_1$: The same as **Hybrid**$_0$, except, $\mathcal{S}_Z$ instead of P_t does the following for every corrupt P_j. That is, if $s_j[i] = 0$, it randomly chooses an m-bit string A_i^j and does the same as in Step 3 to construct each corresponding column of matrix B^j; on the other hand, if $s_j[i] = 1$, it randomly picks an m-bit string B_i^j and computes A_i^j by flipping corresponding bits as mentioned in Step 3. So, this argument is essentially identical to **Hybrid**$_0$.
Hybrid$_2$: The same as previous hybrid, except, $\mathcal{S}_Z$ computes a garbled Bloom filter on behalf of each honest client (i.e., party $P_j \notin Z$) using its own randomness. Note that the indisguishability of this hybrid and **Hybrid**$_1$ stems from using XOR secret sharing scheme by P_t for his inputs to the OTs and also the special way the garbled Bloom filters are constructed.
Hybrid$_3$: The simulated view of $\mathcal{S}_Z$. Due to the security properties of OT protocol and garbled Bloom filter, this hybrid is computationally indistinguishable from **Hybrid**$_2$.

Proof. (P_t is corrupted) We show that there exists a PPT simulator $\mathcal{S}_t$ that given P_t's input and output can generate a simulated view which is computationally indistinguishable from P_t's view in the real execution of the protocol. The simulator can be considered as follows. $\mathcal{S}_t$ first receives P_t's input set X_t, P_{t-1}'s set size n_{t-1}, and the intersection I. Running the OT simulator, $\mathcal{S}_t$ simulates P_t's view as the sender by honestly constructing matrices A^j and B^j for all $i \in [t-1]$. Moreover, for any $x \in I$, it computes $\psi = H_2(A_1[v[1]] \parallel \ldots \parallel A_w[v[w]])$, where $v = F_K(H_1(x))$. Let $\Psi_{\mathcal{I}}$ denote this set of OPRF values. It also picks a set of size n_{t-1} - $|I|$ containing l_2-bit random strings. Let us denote this set by $\Psi_{\mathcal{R}}$. Then, simulator sends $\Psi_{\mathcal{I}} \cup \Psi_{\mathcal{R}}$ to P_t. Eventually, $\mathcal{S}_t$ outputs P_t' simulated view as $\mathcal{S}_t(\lambda, X_t, n_{t-1}, I)$.

We now argue that $\mathcal{S}_t(\lambda, X_t, n_{t-1}, I) \approx view_t^{\Pi}(\lambda, X_1, \ldots, X_t)$ by using a multi-step hybrid argument.

Hybrid$_0$: The view of P_t in the real execution of the protocol.
Hybrid$_1$: The same as **Hybrid**$_0$, except, the protocol terminates if there is any $x^i, x^j \in X_1 \cup X_2 \cup \ldots \cup X_t$, $x^i \neq x^j$ that $H_1(x^i) = H_1(x^j)$. The probability of termination is negligible by collision resistance property of H_1.
Hybrid$_2$: The same as **Hybrid**$_1$, except, the protocol also terminates if there is any $x \in I \backslash X_t$ that for all $i \in [w]$ the number of flipped bits in $B_i[v[i]]$, where $v = F_K(H_1(x))$, be fewer than the security parameter λ. As discussed earlier, the parameters m and w must be chosen such that the probability of termination be negligible.
Hybrid$_3$: The same as **Hybrid**$_2$, except, $\mathcal{S}_t$ runs the OT simulator with honestly selected inputs $\{A_i^j, B_i^j\}$ to simulate the view of P_t as the sender. On account of the security properties of OT protocol, this hybrid is computationally indistinguishable from **Hybrid**$_2$.
Hybrid$_4$: The same as **Hybrid**$_3$, except, the OPRF values sent from P_{t-1} are replaced with l_2-bit random strings for all $x \in I \backslash X_t$. Regarding correlation robustness property of the hash function H_2, it can be shown that this hybrid is computationally indistinguishable from the previous one. More specifically, for all $i \in [w]$, let a_i equals the concatenation of bits $A_i[v[i]]$ and also b_i equals the concatenation of bits $B_i[v[i]]$, where $v = F_K(H_1(x))$. In Step 8, the hash function H_2 takes the concatenation of bits $\bigoplus_{j=1}^{t-1} C_i^j[v[i]]$ as its input which is equivalent to $a_i \oplus [(a_i \oplus b_i) \cdot s]$. Since we know that $\|a_i \oplus b_i\| \geq \lambda$ and also s is a random string unknown to P_t, thanks to the correlation robustness property of H_2, the OPRF value sent to P_2 is pseudorandom.
Hybrid$_5$: The same as **Hybrid**$_4$, except, the protocol does not terminate. This hybrid is P_t's view simulated by $\mathcal{S}_t$. Indeed, what described above simply implies indisguishability of this hybrid and **Hybrid**$_4$.

Remark 1. It is important to mention that the two-party PSI protocol of [CM20] guarantees one-sided malicious security (i.e., against a malicious sender) in the random oracle model. We believe using the same assumption our protocol can also provide security against malicious clients.

4 Complexity Analysis

4.1 Asymptotic Complexity

Now, we analyze the asymptotic complexity of our multi-party PSI protocol. We should highlight the fact that the protocol is concretely efficient since it only relies on cheap tools including oblivious transfer extension, hashing, and bitwise operations. Without loss of generality, we consider n as the set size for all parties.[2] Also, as in [CM20], we set $m = n$. So, by fixing m and n in our complexity analysis w can be regarded as a value depending on λ (Section 3.3).

[2] One can think of n as the upper bound on set sizes.

Communication Pattern	$P_t \to P_j$	$P_j \to P_t$	$P_j \to P_{j+1}$	$P_{t-1} \to P_t$
Star Topology	nw	$w(\lambda - 1)$	–	–
Path Topology	–	–	$1.44nwk$	nl_2

Table 1: Bits sent for leader and clients. Note that we do not consider the initial base-OTs which can be done ahead of time. Also, optimal parameters are considered for garbled Bloom filter.

Recall that we denote party P_t as the leader who takes the main overhead of the protocol and other parties as the clients. In terms of the asymptotic complexity of our protocol, P_t first constructs specially formed matrices A^j and B^j which preparing them takes him linear complexity in n. He then as the sender independently runs w OTs with each client which leads to linear communication and computation complexities in the number of OTs. Apart from running OTs, parties just do hashing and bitwise-XOR operation which regarding the optimal parameters for garbled Bloom filter (as discussed in Section 2.3), they incur linear complexity in both communication and computation. As shown in [CM20] for the case of two-party, it is possible to use random OT in our multi-party PSI protocol which causes the communication overhead from the leader to the clients to be dramatically decreased. We refer the reader to [CM20] to see how random OT can be used in the protocol. Thus, taking this into account and also concerning the optimized semi-honest OT extension of [Ash+13], the total amount of bits exchanged between parties are summarized in Table 1.

Remark 1. Our protocol can be separated into two phases of offline and online which the former can be done before even parties' inputs are available and the latter is executed after learning the inputs. Therefore, a considerable part of the communication and computation costs of the protocol (which includes performing base-OTs, together with the messages sent from receiver to sender in the random OTs) can be done in the offline phase and only lightweight operations take place in the online phase.

Remark 2. Although the overall communication cost is not evenly distributed over all clients and P_{t-1} has less overhead compared to others, he needs to do more evaluation of hash functions in order to compute the OPRF values. So, we can think of it as a trade-off between the P_{t-1}'s communication and computation costs. In addition, the costs in our protocol is rather balanced which makes the protocol preferable in terms of not having a single designated party who has significantly higher overhead compared to others that may cause problem in practice.

Remark 3. An interesting feature of our protocol is that as the number of parties involved in the protocol increases, the communication and computation complexities of each client remain the same. This is a crucial point especially when it

	Communication		Computation		Security	Concretely
Protocol	Leader	Client	Leader	Client	Model	Efficient
[HV17]	$O(tn\lambda)$	$O(n\lambda)$	$O(tn\log(n))$	$O(n)$	Semi-Honest	No
[IOP18]	$O(tn\lambda k)$	$O(tn\lambda k)$	$O(tn\lambda k)$	$O(tn\lambda k)$	Semi-Honest	Yes
[IOP18]	$O(\log(t)n\lambda k)$	$O(\log(t)n\lambda k)$	$O(tn\lambda k)$	$O(tn\lambda k)$	Aug Semi-Honest	Yes
[GN19]	$O((t^2+tn)\lambda)$	$O(n\lambda)$	$O(tn\log(n))$	$O(n\log^2(n))$	Malicious	No
Ours	$O(tn\lambda)$	$O(n\lambda k)$	$O(tn\lambda)$	$O(n\lambda k)$	Semi-Honest	Yes

Table 2: Comparison of communication and computation complexities of multi-party PSI protocols in different security models, where t is the number of parties, n is the size of input sets, k is the number of hash functions, and λ is the security parameter.

comes to having a large number of participants and indeed makes our protocol scale well with the number of parties.

4.2 Comparison

In Table 2, we compare the communication and computation complexities of our multi-party PSI protocol with those of [HV17; IOP18; GN19]. We should mention that having various structures and security levels makes it hard to provide a fair comparison, though, we have tried to pick some recent works with different security models. As in [HV17; GN19], the client's complexities do not depend on the number of parties. However, the two mentioned protocols are not concretely efficient. We observe that the distribution of costs is asymptotically rather fair in our protocol concerning the number of parties t and the number of hash functions k. The workload of parties in [IOP18] is also balanced. We should note that the reported complexities of the augmented semi-honest secure version of [IOP18] are with regard to some optimizations and security relaxations.[3]

5 Conclusion

In this work we proposed a multi-party PSI protocol utilizing a lightweight multi-point OPRF construction. Our protocol is concretely efficient because of involving oblivious transfer extension and garbled Bloom filter as its two core building blocks and achieves linear complexity in both computation and communication concerning each party's input set size. In our protocol, interactions among parties are performed over a combination of star and path network topologies and as a consequence of this design, the asymptotic communication and computation

[3] Augmented semi-honest security is a weaker notion than semi-honest security. We consider the optimized version of the protocol which tries to load balance the interactions between pairs of parties at the cost of some security relaxations.

complexities of each client only rely on his input set size and not on the number of parties, namely $O(n\lambda k)$. In general, this study has gone some way towards presenting an efficient scalable multi-party PSI protocol that can be deployed in practice. This inevitably comes at a cost of relaxation on the security model, but we do believe that future works can focus on enhancing the security of multi-party PSI based on OPRF to obtain more robust security guarantees without that much compromising efficiency.

References

[Aba+17] Aydin Abadi, Sotirios Terzis, Roberto Metere, and Changyu Dong. "Efficient delegated private set intersection on outsourced private datasets". In: *IEEE Transactions on Dependable and Secure Computing* 16.4 (2017), pp. 608–624.

[Ash+13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. "More efficient oblivious transfer and extensions for faster secure computation". In: *Proceedings of the 2013 ACM Conference on Computer and Communications Security*. 2013, pp. 535–548.

[ATD20] Aydin Abadi, Sotirios Terzis, and Changyu Dong. *Feather: Lightweight Multi-party Updatable Delegated Private Set Intersection*. Cryptology ePrint Archive, 2020/407. https://eprint.iacr.org/2020/407. 2020.

[Bad+20] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. *Multi-Party Threshold Private Set Intersection with Sublinear Communication*. Cryptology ePrint Archive, 2020/600. https://eprint.iacr.org/2020/600. 2020.

[BDP20] Pedro Branco, Nico Döttling, and Sihang Pu. *Multiparty Cardinality Testing for Threshold Private Set Intersection*. Cryptology ePrint Archive, 2020/1307. https://eprint.iacr.org/2020/1307. 2020.

[Blo70] Burton H Bloom. "Space/time trade-offs in hash coding with allowable errors". In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[Bud+20] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. *Private Matching for Compute*. Cryptology ePrint Archive, 2020/599. https://eprint.iacr.org/2020/599. 2020.

[CM20] Melissa Chase and Peihan Miao. "Private set intersection in the internet setting from lightweight oblivious PRF". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 34–63.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol". In: *Proceedings of the 2013 ACM Conference on Computer and Communications Security*. 2013, pp. 789–800.

[Dem+18] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. "PIR-PSI: scaling private contact discovery". In: *Proceedings on Privacy Enhancing Technologies* 2018.4 (2018), pp. 159–178.

[Dit+20] Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. *Function Secret Sharing for PSI-CA: With Applications to Private Contact Tracing*. Cryptology ePrint Archive, 2020/1599. https://eprint.iacr.org/2020/1599. 2020.

[DPT20] Thai Duong, Duong Hieu Phan, and Ni 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.

[DT10] Emiliano De Cristofaro and Gene Tsudik. "Practical private set intersection protocols with linear complexity". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2010, pp. 143–159.

[DT12] Emiliano De Cristofaro and Gene Tsudik. "Experimenting with fast private set intersection". In: *International Conference on Trust and Trustworthy Computing*. Springer. 2012, pp. 55–73.

[Efr+21] Aner Ben Efraim, Olga Nissenbaum, Eran Omri, and Anat Paskin-Cherniavsky. *PSImple: Practical Multiparty Maliciously-Secure Private Set Intersection*. Cryptology ePrint Archive, 2021/122. https://eprint.iacr.org/2021/122. 2021.

[FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient private matching and set intersection". In: *International conference on the theory and applications of cryptographic techniques*. Springer. 2004, pp. 1–19.

[Fre+05] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. "Keyword search and oblivious pseudorandom functions". In: *Theory of Cryptography Conference*. Springer. 2005, pp. 303–324.

[GN19] Satrajit Ghosh and Tobias Nilges. "An algebraic approach to maliciously secure private set intersection". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 154–185.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. 2004.

[GS19] Satrajit Ghosh and Mark Simkin. "The communication complexity of threshold private set intersection". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 3–29.

[Hal18] Shai Halevi. "Advanced Cryptography: Promise and Challenges." In: *ACM Conference on Computer and Communications Security*. 2018, p. 647.

[Haz18] Carmit Hazay. "Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs". In: *Journal of Cryptology* 31.2 (2018), pp. 537–586.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. "Private set intersection: Are garbled circuits better than custom protocols?" In: *NDSS*. 2012.

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. "Scalable multi-party private set-intersection". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2017, pp. 175–203.

[IOP18] Roi Inbar, Eran Omri, and Benny Pinkas. "Efficient scalable multiparty private set-intersection via garbled bloom filters". In: *International Conference on Security and Cryptography for Networks*. Springer. 2018, pp. 235–252.

[IR89] Russell Impagliazzo and Steven Rudich. "Limits on the provable consequences of one-way permutations". In: *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. 1989, pp. 44–61.

[Ish+03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. "Extending oblivious transfers efficiently". In: *Annual International Cryptology Conference*. Springer. 2003, pp. 145–161.

[Kal+19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. "Mobile private contact discovery at scale". In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 2019, pp. 1447–1464.

[KK13] Vladimir Kolesnikov and Ranjit Kumaresan. "Improved OT extension for transferring short secrets". In: *Annual Cryptology Conference*. Springer. 2013, pp. 54–70.

[KMS20] Alireza Kavousi, Javad Mohajeri, and Mahmoud Salmasizadeh. "Improved Secure Efficient Delegated Private Set Intersection". In: *2020 28th Iranian Conference on Electrical Engineering (ICEE)*. IEEE. 2020, pp. 1–6.

[Kol+16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. "Efficient batched oblivious PRF with applications to private set intersection". In: *Proceedings of the 2016 ACM Conference on Computer and Communications Security*. 2016, pp. 818–829.

[Kol+17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. "Practical multi-party private set intersection from symmetric-key techniques". In: *Proceedings of the 2017 ACM Conference on Computer and Communications Security*. 2017, pp. 1257–1272.

[Mia+20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. "Two-sided malicious security for private intersection-sum with cardinality". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 3–33.

[Pin+15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. "Phasing: Private set intersection using permutation-based hashing". In: *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 2015, pp. 515–530.

[Pin+18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. "Efficient circuit-based PSI via cuckoo hashing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 125–157.

[Pin+19a] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. "Spotlight: Lightweight private set intersection from sparse ot extension". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 401–431.

[Pin+19b] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. "Efficient circuit-based psi with linear communication". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 122–153.

[Pin+20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay 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.

[PR04] Rasmus Pagh and Flemming Friche Rodler. "Cuckoo hashing". In: *Journal of Algorithms* 51.2 (2004), pp. 122–144.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Scalable private set intersection based on OT extension". In: *ACM Transactions on Privacy and Security (TOPS)* 21.2 (2018), pp. 1–35.

[Rab05] Michael O. Rabin. *How To Exchange Secrets with Oblivious Transfer*. Cryptology ePrint Archive, 2005/187. https://eprint.iacr.org/2005/187. 2005.

[RR16] Peter Rindal and Mike Rosulek. "Faster malicious 2-party secure computation with online/offline dual execution". In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 2016, pp. 297–314.

[Sha79] Adi Shamir. "How to share a secret". In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[Yin+20] Jason H. M. Ying, Shuwei Cao, Geong Sen Poh, Jia Xu, and Hoon Wei Lim. *PSI-Stats: Private Set Intersection Protocols Supporting Secure Statistical Functions*. Cryptology ePrint Archive, 2020/623. https://eprint.iacr.org/2020/623. 2020.

[ZC18] Yongjun Zhao and Sherman SM Chow. "Can you find the one for me?" In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 2018, pp. 54–65.

[Zha+19] En Zhang, Feng-Hao Liu, Qiqi Lai, Ganggang Jin, and Yu Li. "Efficient Multi-Party Private Set Intersection Against Malicious Adversaries". In: *Proceedings of the 2019 ACM Conference on Cloud Computing Security Workshop*. 2019, pp. 93–104.