Practical Over-Threshold Multi-Party Private Set Intersection

1 INTRODUCTION

Two-party private set intersections are practically deployed for ad conversions [22], but multi-party set intersections have many more applications. However, in many of these applications it is required to learn whether a subset of the parties has an element in common. This is known as the over-threshold multi-party private set intersection (OT-MP-PSI) problem [11]. A typical use case is that of collaborating network operation centers. These centers may want to determine whether they are subject to a common attack by comparing log files of indicators of compromise [2]. An attack is already worth investigating, if it affects a subset of the centers, but not all. However, exchanging indicators of compromise without probable cause is not an option, since they contain privacy-protected information, such as IP addresses [13]. The situation is often complicated by different legislations governing those data centers. An OT-MP-PSI protocol can identify common indicators of compromise without revealing any information about those that do not pass the threshold. It is hence privacy-preserving, yet enables the investigation of cyberattacks.

The problem of OT-MP-PSI has been defined and its first protocol has been presented by Kissner and Song [11]. Let there be m parties each holding a database of n elements. Determine the elements (and their owners) that occur in at least t databases, but reveal nothing else as long as not more than k parties collude. Kissner and Song's protocol is not yet practical and no implementation exist. It is a polynomial-based PSI protocol and requires to factor a polynomial of degree nm over a field. It requires $O(m)$ rounds and has a communication complexity of $O(nm^3)$.

The problem could also be solved using generic multi-party computation techniques. We assume that constant round multi-party computation protocols [14, 15] are not yet practical, since no implementation exists, and resort to secret shared multi-party protocols, such as SCALE-MAMBA [1]. Using a circuit similar to the one used in PSI based on two-party computation [9], the optimal circuit size is $O(nm \log^2 nm)$ for $O(n + \log^2 nm)$ rounds or $O(nm \log^2 nm + nmt)$ for $O(\log^2 nm)$ rounds. The communication complexity is larger by at least a factor of $O(m^2)$ [3]. Although, multi-party computation protocols do not require a reconstructions phase, the communication complexity is prohibitive in most cases.

In this paper we present a different construction of OT-MP-PSI. PSI protocols generally exploit locally computation for efficiency. Our protocol is not different, but uses oblivious pseudo-random functions (OPRF) and hashing to achieve efficiency. These techniques are commonly used in efficient two-party PSI protocols [6, 12, 19], but have not yet been applied to OT-MP-PSI. While time complexity $O(nm^t \log^t n)$ of our reconstruction phase is exponential in the threshold t, we take care to keep the constants low in order to scale to practical sizes for the envisioned use case of indicator of compromise exchange. Our protocol is very flexible with tunable parameters for the threshold t and the collusion threshold k. We present a variant with communication complexity $O(nmk)$ and round complexity $O(1)$. However, the constants in reconstruction prevent scaling to practically relevant problem sizes. Our improved scheme has communication complexity $O(nmkt)$ and round complexity $O(1)$. We practically implement and evaluate our protocol showing that the reconstruction phase is feasible for up $m = 10$ parties and a threshold of $t = 5$. Our construction features a new primitive, we call secret-shared oblivious pseudo-random function (SS-OPRF) which returns a secret share of a PRF in an oblivious protocol. This primitve may of independent interest in other applications than OT-MP-PSI.

In summary, this paper contributes

(1) a new OT-MP-PSI protocol with communication complexity $O(nmk)$ and round complexity $O(1)$.
(2) a new OT-MP-PSI protocol with communication complexity $O(nmkt)$, round complexity $O(1)$ and low constants during the reconstruction phase.
(3) a new primitive SS-OPRF which is an OPRF that returns the secret share of a PRF in an oblivious protocol.
(4) a practical implementation and evaluation of our new OT-MP-PSI protocol.

The remainder of the paper is structured as follows.

2 PROBLEM DESCRIPTION

m parties, each with a set of size at most n, would like to compute the set of elements that occur in at least t participants' sets and who the corresponding owners are without revealing elements outside of the threshold intersecting set

3 BACKGROUND

3.1 Secret Sharing

*Bailey: usually written with the threshold first ime.

In secret sharing, the goal is to distribute among m parties *shares* $s_1, \ldots, s_m$ of a secret s. An (t, m)-*threshold scheme** meets the conditions that any size t subset of the shares is sufficient to *reconstruct* s but no group of parties possessing fewer than t shares can infer any information about s. In Shamir's secret sharing scheme [21], the distributing party forms the polynomial

$$f(x) = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \ldots c_1 x + s$$

over a finite field $\mathbb{F}_p$ of prime order p, with $c_1, \ldots, c_{t-1}$ randomly generated. It then produces the shares by evaluating f at m publicly-known distinct values: for instance, by setting $s_{id_i} = (id_i, f(id_i))$. Since t points uniquely determine a polynomial of degree t, given all the s_{id_i}, anyone possessing t shares can use Lagrange interpolation to recover f and thus obtain s. However, fewer than t shares together reveal no information about s.

3.2 Paillier Cryptosystems

*Bailey: should define lcm although I'm assuming it is just least common multiple. Perhaps uppercase.

The Paillier cryptosystem [17] is an additively homomorphic scheme for public key encryption based on the intractability of the Composite Residuosity Class Problem. Ciphertexts are regarded as elements over the multiplicative field $\mathbb{F}_{N^2} = \mathbb{Z}^*_{N^2}$, where $N = p'q'$ for primes p' and q'. Letting $\mu = \text{lcm}(p-1, q-1)$*, and choosing g to be a random base such that its order is divisible by N, the public key is (N, g) and the private key is (p, q). Encrypting a plaintext message $0 \le x < N$ is done by choosing a random $r < N$ and computing:

$$Enc(x) = g^x r^N \mod N^2.$$

A given ciphertext $C < N^2$ is decrypted using μ:

$$Dec(C) = \frac{L(C^\mu \bmod N^2)}{L(g^\mu \bmod N^2)} \mod N,$$

where

$$L(x) = \frac{x-1}{N}.$$

Such a scheme satisfies the following homomorphic properties (the moduli are implicit):

$$Dec(Enc(x_1)Enc(x_2)) = x_1 + x_2$$
$$Dec(Enc(x_1)^{x_1}) = x_1 x_2.$$

3.3 Oblivious Pseudo-Random Function

Let $F_k(x)$ be a pseudo-random function (PRF) with key k on input x. We say a pseudo-random function is secure if $\mathsf{Adv}^{\mathsf{game}^{\mathsf{prf}}}_{\mathcal{A}}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$

```
Game^PRF
$(k, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathsf{st}, x) \leftarrow \mathcal{A}(\mathsf{pk})$
$y_0 \leftarrow F_k(x)$
$y_1 \leftarrow\$ \mathbb{G}$
$b \leftarrow\$ 0, 1$
$b' \leftarrow \mathcal{A}(y_b, \mathsf{st})$
return $b = b'$
```

An oblivious pseudo-random function (OPRF) [7, 19] is a two-party protocol where the client can query a pseudo-random function F_k on input values x, in such a way that the key k, generated by the server, remains hidden from the client, while at the same time the client's input x is not revealed to the server. OPRFs can be established through generic methods for secure multiparty computation (on top of circuits that implement ordinary pseudo-random functions), or by means of the Diffie-Hellman assumption over a group $\mathbb{G}_q$.

An oblivious pseudo-random function (OPRF) is defined as a protocol between a keyholder and client, such that

- the client inputs x and outputs $F_k(x)$ and nothing else.

- the keyholder inputs k and outputs nothing.

A secret-shared oblivious pseudo-random function (SS-OPRF) is defined as a pairwise protocol between a keyholder and one of m clients, such that

- the client inputs x.
- any subset of clients of size $t-1$ or smaller outputs nothing.
- any subset of clients of size t or larger outputs $F_k(x)$ and nothing else.
- the keyholder inputs a key k and outputs nothing.

4 PROTOCOL OVERVIEW

4.1 Setup

In over-threshold multi-party private set intersection there are three entity types: i) m parties each holding at most n elements, ii) k key holders, who are among the m parties, and iii) r reconstructors, who are also among the m parties. First, we consider the case of $k = r = 1$ where also the keyholder and reconstructor $\mathcal{R}$ need to be distinct and extend it to the multi-party case in Section 4.5. The parties want to learn what element appears at least t times among the group, and who the owners are. We introduce two schemes of secret-shared oblivious pseudo-random functions (SS-OPRF) that achieve the goal through the following general steps:

(a) The key holder communicates with each party through an SS-OPRF scheme. During this communication, the key holder uses Shamir secret sharing scheme [21] to generate secret shares and MAC shares and send them to the party for each element owned by the party, while remaining oblivious to the elements.
(b) The parties store their shares and the corresponding MACs for each element into bins, in order to facilitate the reconstruction process.
(c) The reconstructor reconstructs all the possible secrets in each bin. Validating the results with the corresponding MACs reveals which elements were owned by at least t parties.

4.2 Notations, Assumptions, Trust Model

4.2.1 Notations. Table 1 defines the notations used in our schemes. *

*Bailey: for k_2 in the table, publicly known or just public

4.2.2 Assumptions and Trust Model.

- Each bin is padded to max
- Key holder is semi-honest
- Key holder is a party, and does not collude with the reconstructor
- The reconstructor does not collude with more than t-2 (non-key-holder) parties

4.3 Scheme 1

*

*Bailey: Replacing l with ℓ. I will try not to miss any; for readability l can look like 1 to people, especially with subscripts involved at points

4.3.1 Scheme Setup. We have the following set up parameters: field $\mathbb{F}_p$ with prime $p = 2q + 1$, generator g of subgroup $\mathbb{G}_q$, hash function $H(\cdot)$ is used in share generation and $H_B(\cdot)$ is used for hashing-to-bins, global randomnesses: i) the secret key K_1, just known by key holder, ii) the random numbers $c_1, \ldots, c_{t-1} \leftarrow\$ \mathbb{Z}_q$ generated by the key holder, that are fixed for all participants and their all elements. These random numbers are used to calculate the coefficients of the polynomial in Shamir secret sharing scheme. This polynomial is calculated in the exponent of $H(\ell)$ in $P(x)$ as shown in Figure 1.

4.3.2 Share Generation.

(a) Share generation for the element: Party id_i generates a random number α to obliviously send an element's hash, $H(\ell)$, to the key holder. The key holder generates global random numbers $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$. Then it communicates with each participant to generate shares for each participant's element using Shamir secret sharing by forming a polynomial in the exponent. Figure 1 shows the protocol taking place between the key holder and a participant with identifier identifier id_i, $i = 1, \ldots, m$, to generate shares for an element ℓ. This protocol iterates over all elements $\ell \in \mathbb{L}_i$ to generate the corresponding share for each element owned by id_i.
(b) Our schemes consist of two parallel procedures as shown in Figures 1 and 2. The first one is the generating the secret shares for each element as described in (a). The second procedure is generating shares for the MAC of the distributed secret

Table 1: Notations

m	The number of parties
id_i	Identifier for party i; $i = 1, \ldots, m$
n	Maximum size of the set of elements owned by a party ($n = \max_{i=1,\ldots,m} \lvert\mathbb{L}_i\rvert$)
$\mathbb{L}_i$	The set of elements owned by id_i; $i = 1, \ldots, m$
$\mathbb{L}$	The set of all elements owned by parties ($\mathbb{L} = \cup_{i=1,\ldots,m}\mathbb{L}_i$)
t	threshold (intersection and the secret sharing threshold)
h	number of key holders ($h = 1$ in the default schemes)
b	number of bins ($b = \frac{n}{\log n}$); bins are shown as $b_1, \ldots, b_b$
$\lvert b_i\rvert$	The size of the bin b_i, which is $c \times \log n$; for $i = 1, \cdots, b$
$H(\cdot)$	Hash function used in the SS-OPRF schemes
$H_B(\cdot)$	Hash function used in hashing-to-bins
$\mathbb{G}_q$	The group or the PRF
$\mathbb{F}_p$	The base prime field in SS-OPRF schemes
$\mathbb{F}_N$	The field of the plaintext of the homomorphic encryption scheme
$\mathbb{F}_{N^2}$	The field of the ciphertext of the homomorphic encryption scheme
Enc_{pk}	Encryption with public key pk in the homomorphic encryption scheme
Dec_{sk}	Decryption with private key sk in the homomorphic encryption scheme
K_1	The key holder's secret key used to generate secret shares for each element
K_2	The key holder's publicly known key used to generate the MAC for the secrets
$Share_i(\ell)$	Share of the element ℓ for party i
$MAC_i(\ell)$	Share of the MAC of the element ℓ for party i

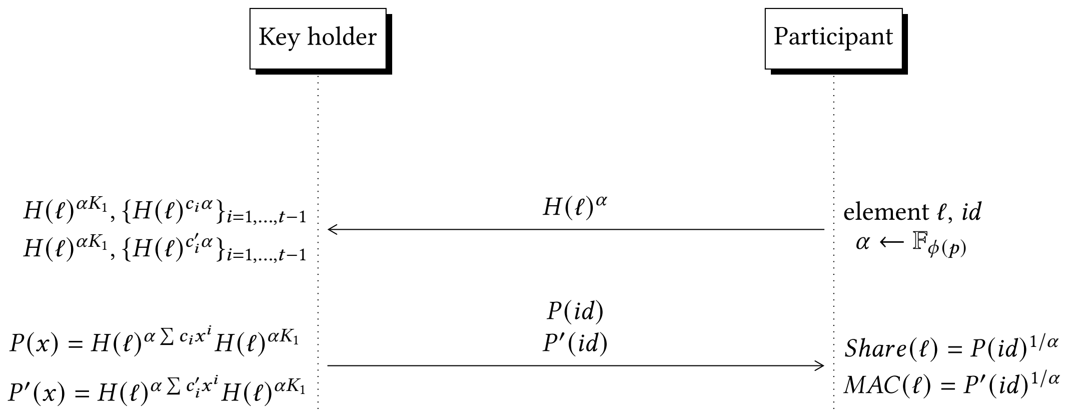
Key holder
Participant
$H(\ell)^{\alpha K_1}, \{H(\ell)^{c_i\alpha}\}_{i=1,\ldots,t-1}$
$H(\ell)^{\alpha K_1}, \{H(\ell)^{c'_i\alpha}\}_{i=1,\ldots,t-1}$
$H(\ell)^{\alpha}$
element ℓ, id
$\alpha \leftarrow \mathbb{F}_{\phi(p)}$
$P(x) = H(\ell)^{\alpha \sum c_i x^i} H(\ell)^{\alpha K_1}$
$P'(x) = H(\ell)^{\alpha \sum c'_i x^i} H(\ell)^{\alpha K_1}$
$P(id)$
$P'(id)$
$Share(\ell) = P(id)^{1/\alpha}$
$MAC(\ell) = P'(id)^{1/\alpha}$

Figure 1: Communication between the key holder and a participant id_i in Scheme 1 - Share generation for an element $\ell \in \mathbb{L}_i$, owned by id_i

in (a). The purpose of these MACs is to verify the secrets that are reconstructed later in Section 2 by the reconstructor $\mathcal{R}$. The key holder uses a second polynomial $P'(x)$ to generate MACs for the elements' secrets.

4.3.3 Hashing-to-bins. Participant' hash their shares to b ($=\frac{n}{\log n}$) bins, each of size $c \times \log n$* to reduce the computation cost for the reconstructor. Corresponding shares to an element ℓ are stored in bin number $H_B(\ell)$. Each stored value is a quadruple consisting of i) the participant's identifier, id_i, ii) id_i's share for its element ℓ, $Share_i(\ell)$, iii) the corresponding MAC, $MAC_i(\ell)$, and iv) the number of the bin, $H_B(\ell)$, that stores the information.

*Florian: Maybe this should $O(\log n)$

4.3.4 Reconstruction. After all participants store their shares in the corresponding bins, the reconstructor $\mathcal{R}$ – who is also a participant – reconstructs secrets in each bin. Recall that each bin might contain shares of several elements from each party. The reconstructor is not able to recognize which shares correspond to the same element. Hence, $\mathcal{R}$, forms all possible combinations of t shares – $\binom{m}{t}$ – from distinct parties, $id_{i_1}, \ldots, id_{i_t}$, with the hope of finding the shares of the same elements. In order to find those shares, $\mathcal{R}$ uses the corresponding MACs to each set of shares that verifies if the shares in the set correspond to the same element. As the polynomial is in the exponent of the generator in this scheme, the reconstructor applies the Lagrange interpolation in the exponent to calculate the secret as follows: $H(\ell)^{K_1} = \prod_{w=0}^{t} Share_{i_w}(\cdot)^{(\prod_{w' \neq w} \frac{-id_{w'}}{id_w - id_{w'}})}$. $\mathcal{R}$ verifies if the selected set of shares correspond to the same element by calculating the MAC of the set and evaluating the results of the two calculations. These reconstruction steps are summarized in the Algorithm 1.

Algorithm 1 RECONSTRUCT$_{\text{SCHEME2}}$

```
1: /*$K_1$ is key holder's secret key and $K_2$ is a publicly known key used for MAC generation*/
2: /*$H(\cdot)$ is a hash function in share generation*/
3: /*$H_B(\cdot)$ is a hash functions used for hashing-to-bin (Section 4.4.3)*/
4: for each Participant $id_i$; $i = 1, \ldots, m$ do
5:     for each Element $\ell$ owned by $id_i$; $\ell \in \mathbb{L}_i$ do
6:         $id_i$: store $(id_i, Share_i(\ell), MAC_i(\ell), H_B(\ell))$ in the bin number $H_B(\ell)$
7:
8: for each Bin $b_z$; $z = 1, \ldots, b$ do
9:     for each t-subset of quadruples in the bin $b_z$; $\{(id_{i_1}, Share_{i_1}(\ell), MAC_{i_1}(\ell), b_z), \ldots, (id_{i_t}, Share_{i_t}(\ell), MAC_{i_t}(\ell), b_z)\}$, where $id_{i_i}$s are distinct, do
10:        $\mathcal{R}$: Reconstruct the corresponding secret, $Secret_{share}$, to the points $\{(i_1, Share_{i_1}), \ldots, (i_t, Share_{i_t})\}$
11:        $\mathcal{R}$: Reconstruct the corresponding secret, $Secret_{MAC}$, to the points $\{(i_1, MAC_{i_1}), \ldots, (i_t, MAC_{i_t})\}$
12:        if $Secret_{MAC} == Secret_{share}$ then Reveal that $(id_{i_1}, \ldots, idi_t)$ can reconstruct the $Secret_{share}$ which is $H(\ell)^{K_1}$
```

4.3.5 Complexity. We analyse the complexity of Scheme 1 in its base version with a single key holder. We discuss the multi key holder case in Section 4.5.3.

THEOREM 4.1. *The communication complexity of Scheme 1 in the number of rounds is $O(1)$.*

PROOF. As shown in Figure 1, the communication between the key holder and each party takes places in one rounds, regardless of the number of parties, the number of elements or the threshold. Hence Scheme 2 is $O(1)$. □

THEOREM 4.2. *The communication complexity of Scheme 1 in each rounds is $O(nm)$, where m is the number of parties and n is the maximum size of the set of the elements owned by a party.*

PROOF. As shown in Figure 1, the key holder and the party communicate 2 messages in the round. This communication is iterated n times for each of m parties to generate share for all of their elements. Hence, Scheme 2 is $O(nm)$. □

THEOREM 4.3. *The computation complexity of Scheme 1 is $O(bm^t(\log n)^t)$[1]; where $b = O(\frac{n}{\log n})$ is the number of bins, m is the number of parties, and t is the threshold value in the over-threshold set intersection scheme.*

PROOF. The proof follows the same reasoning as the proof of Theorem 4.6. As described in Algorithm 1, the reconstructor $\mathcal{R}$ forms all possible combinations of t shares from distinct parties in each bin. There are $\binom{m}{t}$ options for choosing a set of t distinct parties out of m of them. Moreover, each of these parties have $\log n$ shares in each bin. Hence, forming a set of t shares from distinct parties in a bin takes $m^t(\log n)^t$ combinations. This procedure iterates b times to cover all the bins. □

4.4 Scheme 2

The reconstruction in Scheme 1 has very high constants, since Lagrange interpolation is executed in the exponent and modular exponentiations are slow. To improve the reconstruction time, we construct Scheme 2 with Lagrange interpolation in the base. This is challenging, since the PRF requires exponentiation, but secret sharing requires addition. No cryptographic scheme supports both operations on its plaintext securely. Hence, we use a conversion between two schemes – discrete logs and additively homomorphic encryption.

[1]The computation complexity is $O(bm^t(\log n)^t)$ for small values of t. For large values of t and m, the complexity is $O(2^m(\log n)^t)$.

4.4.1 Scheme Setup. Similar to Scheme 1, this scheme uses field $\mathbb{F}_p$ with prime $p = 2q + 1$, generator g of subgroup $\mathbb{G}_q$, hash function $H(\cdot)$ is used in share generation and $H_B(\cdot)$ is used for hashing-to-bins, global randomnesses: i) K_1 known and used by key holder, ii) the random numbers $c_1, \ldots, c_{t-1}, c'_1, \ldots, c_{t-1} \leftarrow\!\$\, \mathbb{Z}_q$ generated by the key holder, that are fixed for all participants and their all elements. These random numbers are used to calculate the coefficients of the polynomial in Shamir secret sharing scheme. Participants use a homomorphic encryption scheme, Enc, over plaintext field $\mathbb{F}_N$ where $N > 2^\lambda p^2$.

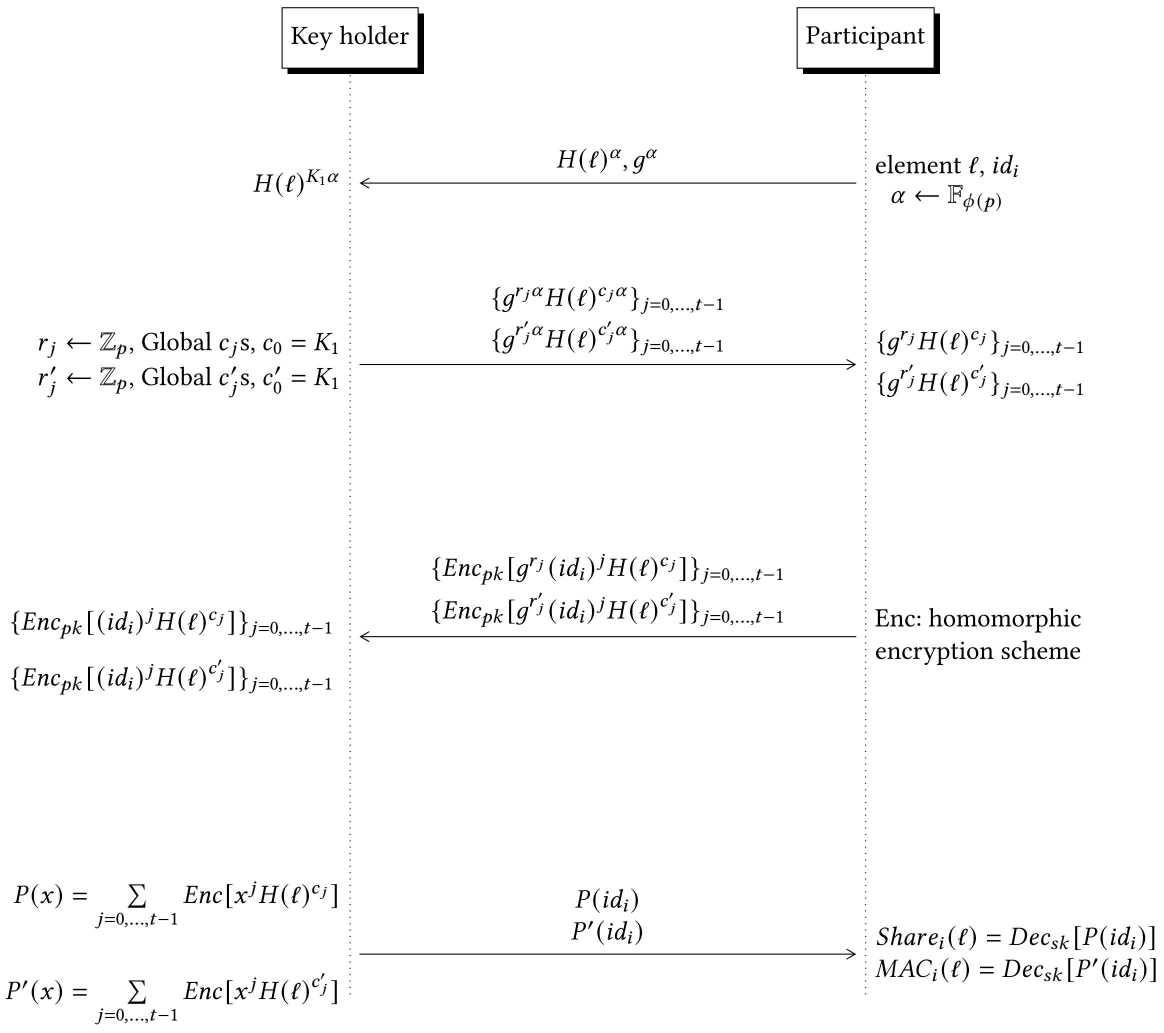
Key holder
Participant
$H(\ell)^{K_1\alpha}$
$H(\ell)^\alpha, g^\alpha$
element ℓ, id_i
$\alpha \leftarrow \mathbb{F}_{\phi(p)}$
$\{g^{r_j\alpha}H(\ell)^{c_j\alpha}\}_{j=0,\ldots,t-1}$
$\{g^{r'_j\alpha}H(\ell)^{c'_j\alpha}\}_{j=0,\ldots,t-1}$
$r_j \leftarrow \mathbb{Z}_p$, Global c_js, $c_0 = K_1$
$r'_j \leftarrow \mathbb{Z}_p$, Global c'_js, $c'_0 = K_1$
$\{g^{r_j}H(\ell)^{c_j}\}_{j=0,\ldots,t-1}$
$\{g^{r'_j}H(\ell)^{c'_j}\}_{j=0,\ldots,t-1}$
$\{Enc_{pk}[g^{r_j}(id_i)^jH(\ell)^{c_j}]\}_{j=0,\ldots,t-1}$
$\{Enc_{pk}[g^{r'_j}(id_i)^jH(\ell)^{c'_j}]\}_{j=0,\ldots,t-1}$
$\{Enc_{pk}[(id_i)^jH(\ell)^{c_j}]\}_{j=0,\ldots,t-1}$
$\{Enc_{pk}[(id_i)^jH(\ell)^{c'_j}]\}_{j=0,\ldots,t-1}$
Enc: homomorphic encryption scheme
$P(x) = \sum_{j=0,\ldots,t-1} Enc[x^jH(\ell)^{c_j}]$
$P'(x) = \sum_{j=0,\ldots,t-1} Enc[x^jH(\ell)^{c'_j}]$
$P(id_i)$
$P'(id_i)$
$Share_i(\ell) = Dec_{sk}[P(id_i)]$
$MAC_i(\ell) = Dec_{sk}[P'(id_i)]$

Figure 2: Communication between the key holder and a participant id_i in Scheme 2 - Share generation for an element $\ell \in \mathbb{L}_i$, owned by id_i

4.4.2 Share Generation.

(a) Share generation for each element: Party id_i generates a random number α to obliviously send its element's hash value, $H(\ell)$, to the key holder. The key holder generates random numbers r_1 and r_2* for this session[2] to mask the secret value $H(\ell)^{K_1\alpha}$ and send it to the party. The key holder also uses t global random numbers $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$ to generate the coefficients of the polynomial $P(x)$ in Shamir secret sharing scheme. We use the term global to indicate that the same random numbers are used in all mn sessions that generate the secret shares for n elements for each of m parties. The key holder communicates with each participant to generate shares for each participant's element using the polynomial $P(x)$.

*Florian: I think this needs fixing.

[2]In each session, the key holder generates the share and its MAC for an element of a party

Since the exponentiation is not required for the rest of the protocol, the party removes the α exponent and encrypts the messages using a homomorphic encryption scheme. Figure 2 shows the protocol taking place between the key holder and a participant with identifier identifier id_i, $i = 1, \ldots, m$, to generate shares for an element l. This protocol iterates over all elements $\ell \in \mathbb{L}$ to generate the corresponding share for each element owned by id_i.

(b) Share generation for the MAC of each element: As described in MAC generation in Section 4.3.2(b), our schemes consist of a MAC generation procedure in parallel to the share generation for secrets. The MAC's are used in the reconstruction phase in Section 4.4.4 to verify the reconstructed secrets.

4.4.3 Hashing-to-bins. This step is identical to the hashing-to-bins described in Section 4.3.3.

4.4.4 Reconstruction. The reconstruction procedure of Scheme 2 is similar to the reconstruction in Section 4.3.4 for Scheme 1. Each party's shares are hashed into the corresponding bins, with each bin containing multiple shares of each party. The reconstructor $\mathcal{R}$ –who is also a participant– reconstructs the secrets in each bin for every $\binom{m}{t}$ subset of shares for distinct parties, $id_{i_1}, \ldots, id_{i_t}$. In other words, for each $\binom{m}{t}$ subset of the shares in the bin from distinct parties, $id_{i_1}, \ldots, id_{i_t}$, $\mathcal{R}$ reconstructs the corresponding secret by applying Lagrange interpolation: $\sum_{w=0}^{t} Share_{i_w}(\cdot)(\prod_{w' \neq w} \frac{-id_{i_{w'}}}{id_{i_w} - id_{i_{w'}}})$. For a set of shares that correspond to element ℓ, the Lagrange interpolation results in $H(\ell)^{K_1}$. Similarly, $\mathcal{R}$ calculates the corresponding MAC for the selected set of t shares mentioned earlier, which equals $H(\ell)^{K_1}$ as well if the set of shares correspond to the element ℓ. $\mathcal{R}$ can verify the equality in the result of the Lagrange interpolation on the secret shares with the corresponding results of the MAC shares and validate the secret. These reconstruction steps are summarized in the Algorithm 2.

Algorithm 2 RECONSTRUCT$_{\text{SCHEME1}}$

```
1: /*$K_1$ is key holder's secret key and $K_2$ is a publicly known key used for MAC generation*/
2: /*$H(\cdot)$ is a hash function in share generation*/
3: /*$H_B(\cdot)$ is a hash functions used for hashing-to-bin (Section 4.4.3)*/
4: for each Participant $id_i$; $i = 1, \ldots, m$ do
5:     for each Element $\ell$ owned by $id_i$; $\ell \in \mathbb{L}_i$ do
6:         $id_i$: store $(id_i, Share_i(\ell), MAC_i(\ell), H_B(\ell))$ in the bin number $H_B(\ell)$
7:
8: for each Bin $b_z$; $z = 1, \ldots, b$ do
9:     for each t-subset of quadruples in the bin $b_z$; $\{(id_{i_1}, Share_{i_1}(\ell), MAC_{i_1}(\ell), b_z), \ldots, (id_{i_t}, Share_{i_t}(\ell), MAC_{i_t}(\ell), b_z)\}$, where $id_{i_j}$s are distinct, do
10:        $\mathcal{R}$: Apply Lagrange interpolation to find the corresponding intercept, $Secret_{share}$, for the polynomial that covers the points $\{(i_1, Share_{i_1}), \ldots, (i_t, Share_{i_t})\}$
11:        $\mathcal{R}$: Apply Lagrange interpolation to find the corresponding intercept, $Secret_{MAC}$, for the polynomial that covers the points $\{(i_1, MAC_{i_1}), \ldots, (i_t, MAC_{i_t})\}$
12:        if $Secret_{MAC} == Secret_{share}$ then Reveal that $(id_{i_1}, \ldots, idi_t)$ can reconstruct the $Secret_{share}$ which is $H(\ell)^{K_1}$
```

4.4.5 Complexity. We analyse the complexity of Scheme 2 in its base version with a single key holder. The multi key holder scheme is discussed in Section 4.5.3.

THEOREM 4.4. *The communication complexity of Scheme 2 in the number of rounds is $O(1)$.*

PROOF. As shown in Figure 2, the communication between the key holder and each party takes places in two rounds, regardless of the number of parties, the number of elements or the threshold. Hence Scheme 2 is $O(1)$. □

THEOREM 4.5. *The communication complexity of Scheme 2 in each rounds is $O(nmt)$, where m is the number of parties, n is the maximum size of the set of the elements owned by a party, and t is the threshold in the over-threshold set intersection scheme.*

PROOF. As shown in Figure 2, the key holder and the party communicate $t + 3$ messages in the first round and $t + 2$ messages in the second round. This communication is iterated n times for each of m parties to generate share for all of their elements. Hence, Scheme 2 is $O(nmt)$. □

THEOREM 4.6. *The computation complexity of Scheme 2 is $O(bm^t(\log n)^t)$; where b is the number of bins, m is the number of parties, and t is the threshold value in the over-threshold set intersection scheme.*

PROOF. As described in Algorithm 2, the reconstructor $\mathcal{R}$ forms all possible combinations of t shares from distinct parties in each bin. There are $\binom{m}{t}$ ways for choosing a set of t distinct parties out of m of them. Moreover, each of these parties have $\log n$ shares in each bin. Hence, forming a set of t shares from distinct parties in a bin takes $m^t(\log n)^t$ combinations. This procedure iterates b times to cover all the bins. □

4.5 Additional Notes

4.5.1 Statistical Secret Sharing Scheme. Our protocol requires switching between two fields to perform the share generation, described in Figures 2 and 1: i) the base prime field $\mathbb{F}_p$ and ii) the field $\mathbb{F}_N$ in which the homomorphic encryption is performed. We use the statistical secret sharing scheme introduced in [4] to covert a secret-shared message over a prime field to another field. Hence, we require $N > 2^\lambda p^2$.

4.5.2 Multi-Reconstructor Scheme. We described in Algorithms 1 and 2 how the reconstructor $\mathcal{R}$ reconstructs the secrets in each bin. We emphasize the reconstruction is not restricted to a single party and can be performed by multiple parties in parallel.

4.5.3 Multi key holder Scheme. We can extend our schemes from a single key holder scenario to a multi key holder one. In this extension, all or a subset of all parties form the key holders set. We describe this extension only briefly due to space restrictions. The key holders received the initial message from the participant and run a secure computation protocol among themselves using a group key and group random numbers (in Scheme 2). The group keys $c_j = c_{j,1} + c_{j,2} + \ldots c_{j,k}$ and the group random numbers $R_j = g^{\alpha r_j} = R_{j,1} + R_{j,2} + \ldots R_{j,k}$ (in Scheme 2) are shared additively. Keyholder id_i holds shares $c_{j,i}$ and $R_{j,i}$. The k keyholders need to multiply the exponentiation of the participant's message and their key share and finally the sum of their random group shares using a secure computation. Instead of performing this secure computation using secret-share based secure computation with $O(k^2)$ communication in $O(\log k)$, the parties can use precomputed shares $R_j = R'_{j,1} \cdot R'_{j,2} \cdot \ldots R'_{j,k}$, since all inputs are random and independent of the elements in $\mathbb{L}$. Each keyholder sends the product $R'_{j,i} H(x)^{\alpha c_{j,i}}$ to the first keyholder who multiplies them. In the semi-honest model each keyholder follows the protocol, but must not learn additional information. The first keyholder learns no information, since all products are indistinguishable from random numbers. In round 2 of Scheme 2 the keyholders use precomuted additive shares of the inverse $R_j^{-1/\alpha}$ of the group random number. Then, the product (of the ciphertexts / sum of the plaintexts) can again be performed by sending it to the first keyholder, since they are semantically secure ciphertexts. The entire protocol has $O(k)$ communication and runs in $O(1)$ rounds with an offline pre-computation phase. The multi key holder scheme follows the exact same procedure as in Schemes 1 and 2 otherwise.

5 SECURITY

Definition 5.1. We say the Decisional Diffie-Hellman (DDH) Assumptions holds if for any PPT adversary $\mathcal{A}$

$$a, b, c \leftarrow_{\$} \mathbb{Z}_q$$
$$\Pr\left[\mathcal{A}(g^a, g^b, g^c) = 1\right] - \Pr\left[\mathcal{A}(g^a, g^b, g^{ab}) = 1\right] < \mathsf{negl}(\lambda)$$

*Bailey: which H. The hash function $H(\cdot)$ or need a different notation here

+Florian: Fixed.

THEOREM 5.2. *If the DDH Assumption holds and $H(\cdot)$*+ is a programmable random oracle, each coefficient of the share share's polynomial in Scheme 2 is a pseudo-random function.*

PROOF. We prove by reducing an adversary in Game$^{\mathsf{PRF}}$ to an adversary against the DDH Assumption. Let g^a, g^b, g^c be a DDH instance. We program $H(\cdot)$ to output g^a on input x. We set the public pk to g^b. We can still answer queries for $F_k(x')$ where $x' \neq x$ by programming $H(\cdot)$ to choose and store $r \leftarrow_{\$} \mathbb{Z}_q$ and output $R = g^r$. An answer to an OPRF query for x' is then $g^{br} = H(x')^b$. We set y_b to g^c. We set the output of adversary $\mathcal{A}^{\mathsf{DDH}}$ to the output of adversary $\mathcal{A}^{\mathsf{PRF}}$. All outputs are indistinguishable from the real scheme and any advantage between the two adversaries translates directly. □

THEOREM 5.3. *Scheme 2 is a secret-shared oblivious pseudo-random function.*

COROLLARY 5.4. *There exist simulators $\mathsf{Sim}_{Participant}(x, Share_i(x)$ and $\mathsf{Sim}_{Keyholder}(k)$ that are computationally indistinguishable from the messages received during the protocol.*

*Bailey: I'm confused by the leq reconstruction here

+Florian: Fixed.

PROOF. Correctness is given by the protocol construction, such that $Share_i(x)$ is returned to the participant and $t' \geq t$ participants*+ can reconstruct 0.

Since each coefficient of the Shamir secret shares is the output of a PRF (Theorem 5.2), the output of Scheme 2 is computationally indistinguishable from a secret share of 0. Furthermore, $t-1$ secret shares are perfectly indistinguishable from a set of $t-1$ uniformly chosen random numbers, since they leave one degree of freedom for choosing x, as long as it is not a priori known that they reconstruct to 0. To explain further, consider an adversary that controls $t-2$ parties. If this adversary obtains a share $Share_i(x')$ where party i is not controlled by the adversary, e.g. during reconstruction. Then this adversary cannot determine whether $x' = x$ for any x chosen by the adversary, since any $t-1$ shares may reconstruct to 0. Now, consider an adversary that controls $t-1$ parties. This adversary can choose x, obtain secret shares and reconstruct the coefficients of the secret share polynomial using $t-1$, since it knows the "secret" 0. Hence, it can test whether for another share $Share_i(x')$ it holds $x' = x$. However, the output of this test is included in the output of the protocol and the attack would be feasible for any adversary admissible to the protocol. We repeat, $t-1$ secret shares are perfectly indistinguishable from a set of $t-1$ uniformly chosen random numbers, as long as it is not a priori known that they reconstruct to 0.

It remains to show that the protocols is oblivious and the simulators exist.

We construct the simulator $\mathsf{Sim}_{Participant}(x, Share_i(x)$ as follows. The simulator outputs $2t$ random elements $r \leftarrow\$ \mathbb{F}_p$. This is perfectly indistinguishable, since the keyholder chooses a uniform random blinding element per message. Let $r, r' \leftarrow\$ [0, 2^\lambda p]$. The simulator outputs $\mathsf{Enc}(rp + Share_i(x))$ and $\mathsf{Enc}(r'p + Share'_i(x))$. This is statistically indistinguishable, since the keyholder uses share conversion to hide the multiplications in $\mathbb{F}_N$.

We construct the simulator $\mathsf{Sim}_{Keyholder}(k)$ as follows. The simulator outputs $a, c \leftarrow\$ \mathbb{Z}_q$ and programs the random oracle $H(\cdot)$ for $x \leftarrow mathbbL$ to $g^{c/a}$. This is perfectly indistinguishable, since all values are uniform and the random oracle is consistent. The simulator outputs $2t$ random elements $r \leftarrow\$ \mathbb{F}_{N^2}$. This is computationally indistinguishable, since Paillier ciphertexts are semantically secure [17].

□

6 SS-OPRF

6.1 Security Definition

Requirements: -keyholder oblivious -PRF output should be random (obviously) - Given (t-1) elements all of them appear random (b/c our ss-oprf has an IT argument and a computatioal argument) - with $H(x)^k$ and (t-1) shares still appears random -with t, cannot use it to solve prf property...

Obliviousness

Random PRF Output

No reconstruction even given PRF

6.2 SS-OPRF 1

Rasoul's construction, secret sharing in the exponent

$A- > S : x^r$

$S- > A : x^{rSS(c)}$ $A : x^{SS(c)}$

Not sure I got this right.

Problem reconstruction involves modular exponentiation

6.3 Security Proof

6.4 SS-OPRF 2

My construction, 2 round protocol

$A- > S : g^r, x^r$

$S- > A : (g^r)^{\log s} x^{rc}$

$A- > S : E(((g^r)^{\log s} x^{rc})^{1/r}) = E(sx^c)$

$S- > A : E(SS(x^c))$

6.5 Security Proof

7 COMPLETING THE PROTOCOL

7.1 Verifying the Reconstruction of Shares

Give x^c and $x^{cc'}$, reveal c'

7.2 Distributing S

Share $c = c_1 c_2 c_3 ...$

Also distribute decryption

7.3 Reducing the number of possible share combinations

Use hashing to bin the elements. Only reconstruct within one bin

8 EVALUATION

8.1 SS-OPRF

Time and communication to compute one SS-OPRF1 Time and communication to compute one SS-OPRF2

8.2 Reconstruction

Time to reconstruct OT-SI in the base Time to reconstruct OT-SI in the exponent

9 RELATED WORK

Private set intersection is a well-studied problem, although most prior work deals with its standard, non-threshold formulation: the specific case when $t = m$. We first discuss approaches to the threshold PSI problem discussed in this paper and then give an overview of the techniques used for standard PSI. A more comprehensive overview of the latter is included in [19].

9.1 Over-Threshold PSI

Kissner and Song [11] proposed protocols for private set intersection and various related problems such as cardinality set intersection, where only the size of the set is revealed; over-threshold intersection, which roughly corresponds to our problem; and threshold set intersection, where it is not revealed how many parties own a given element in the intersection. Their approach (to threshold and over-threshold intersection) does not reveal the identities of the parties which own a given element in the intersecting set and involves a small amount of local computation (on the order of the communication cost). However, their protocol's total communication complexity is $O(nm^3)$ and takes $O(m)$, which makes it less suitable than our $O(nmt)$ scheme requiring $O(1)$ rounds.

Threshold PSI can also be achieved using generic secure computation protocols since it corresponds to the functionality

$$f(\mathbb{L}_1, \ldots, \mathbb{L}_m) = \{x \mid \exists i_1, \ldots i_t,\ x \in \cap_{r=1}^{t} \mathbb{L}_{i_r}\}.$$

f can be implemented using the following circuit: the union $\mathbb{L}$ of all input elements is sorted (using a sorting network), and the resulting output is scanned for contiguous groups of at least t identical elements. Note that since the required output is an (unordered) set, we need to randomly shuffle all such found elements to avoid leaking additional information. The communication complexity of this approach depends on the resulting circuit size. The circuit size for sorting (and random shuffling) is $O(nm \log^2 nm)$. The circuit size for scanning depends whether it is done sequentially with multiplicative depth $O(nm)$ in size $O(nm)$ or $O(\log t)$ multiplicative depth in size $O(nmt)$.

9.2 Standard PSI

9.2.1 Public-Key Protocols. Many approaches to PSI rely heavily on public-key cryptography, as in our paper. Meadows et al. [16] and Huberman et al. [10] present protocols based on Diffie-Hellman key exchange, while several based on RSA feature in the more recent work by De Cristofaro et al [5]. The protocols in [6, 8] are similar to our work in that they make use polynomial interpolation and the Paillier cryptosystem (or alternatively, ElGamal), ultimately relying on the decisional Diffie-Hellman assumption. Public-key approaches to PSI tend to have better communication complexity than other methods, at the expense of having higher overall computational costs [19].

9.2.2 Protocols Based on Oblivious Transfer. Oblivious transfer (OT) [20] is a two-party protocol where one of the parties, the sender, holds several pieces of data, one of which will be sent to the receiver. The receiver can freely choose which of the messages to learn, however, the sender does not obtain any information about the receiver's choice. A common way to perform a large number of OT executions with low amortized cost is through OT extension [19], which is often used in the construction of OPRFs. Recent work in this direction is that of Kolesnikov et al. [12], who greatly improved the communication cost of protocols based on OT extension. Further improvements were introduced by Pinkas et al. in [19] and then subsequently in [18].

10 CONCLUSION

REFERENCES

[1] Abdelrahaman Aly, Marcel Keller, Dragos Rotaru, Peter Scholl, Nigel P.Smart, and Tim Wood. SCALE–MAMBA software. https://homes.esat.kuleuven.be/~nsmart/SCALE/, 2020.

[2] Eric W. Burger, Michael D. Goodman, Panos Kampanakis, and Kevin A. Zhu. Taxonomy model for cyber threat intelligence information exchange technologies. In *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*, pages 51–60, 2014.

[3] Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure mpc, with or without preprocessing. In *Advances in Cryptology – CRYPTO*, pages 61–84, 2019.

[4] Ivan Damgård and Rune Thorbek. Efficient conversion of secret-shared values between different fields. *IACR Cryptology ePrint Archive*, 2008:221, 01 2008.

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

[6] Michael J Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

[7] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference*, pages 303–324. Springer, 2005.

[8] 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*, pages 1–19. Springer, 2004.

[9] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012.

[10] Bernardo A Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 78–86, 1999.

[11] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO*, pages 241–257, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] 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 SIGSAC Conference on Computer and Communications Security*, pages 818–829, 2016.

[13] Frederick Lah. Are ip addresses "personally identifiable information"? *I/S: A Journal of Law and Policy for the Information Society*, 4:681–707, 2008.

[14] Yehuda Lindell, Benny Pinkas, Nigal P. Smart, and Avishay Yanai. Efficient Constant-Round Multi-party Computation Combining BMR and SPDZ. *Journal of Cryptology*, 32(3):1026–1069, 2019.

[15] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More Efficient Constant-Round Multi-party Computation from BMR and SHE. In *Proceedings of the 14th International Conference on Theory of Cryptography*, pages 554–581, 2016.

[16] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE, 1986.

[17] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[18] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In *Annual International Cryptology Conference*, pages 401–431. Springer, 2019.

[19] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[20] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[21] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[22] Moti Yung. From mental poker to core business: Why and how to deploy secure computation protocols? In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 1–2, 2015.