Over-Threshold Multi-Party Private Set Intersection

1 INTRODUCTION

Use case: collaborating network operations centers under different legislations

Brief problem intro

Comparison to Kissner, Song protocol [3]:
- $O(1)$ rounds vs. $O(m)$ - problem-adaptive communication complexity $O(nmk)$ vs. $O(nm^3)$

Comparison to generic secure computation:
- lower communication complexity $O(nmk)$ vs. $O(nm^3 \log^2 nm)$ – assuming optimal circuit is $O(nm \log^2 nm)$
- constant round multi-party protocols not yet practical

SS-OPRF

Two variants: SS in the exponent, communication O(nmk); SS in the base, communication O(nmkt)

Contributions
- new construction of over-threshold multi-party private set intersection protocol with communication ... and security ...
- two secret-shared oblivious pseudo-random function (SS-OPRF)
- practical evaluation?

2 PROBLEM DESCRIPTION

m parties, each with a set of size n, would like to compute the number of occurences of each element that occurs at least t, but nothing else. The computation should be secure against a coalition of k semi-honest parties.

3 BACKGROUND

3.1 OPRF

$A- > S : x^a$
$S- > A : x^{as}$
$A : x^s$

3.2 Secret Shares

Shamir secret shares

3.3 Definition SS-OPRF

OPRF that outputs secret share of PRF

Note that the ss polynomial needs to be different for every message

3.4 Paillier [4] and Damgard Jurik [1] Crytposystems

4 PROTOCOL OVERVIEW

4.1 Setup

In over-threshold multi-party private set intersection there are three entity types: i) a key holder, ii) m parties each holding at most n elements, and iii) a reconstructor $\mathcal{R}$, who is also one of the m parties. The parties want to learn what element appears at least t times among the group and they should not learn anything else. 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 [5] 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.

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 $b_1, \ldots, b_b$
$H(\cdot)$	Hash function used in the SS-OPRF schemes
$H_B(\cdot)$	Hash function used in hashing-to-bins
$\mathbb{F}_p$	The base prime field in SS-OPRF schemes
$\mathbb{F}_q$	The field for 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
$Share_i(l)$	Share of the element l for party i
$MAC_i(l)$	Share of the MAC of the element l for party i

4.2.2 Assumptions and Trust Model.

- Each bin is padded to max
- Key holder is semi-honest
- Key holder is not a party, and does not collude with any party
- The reconstructor sees at most t-1 shares

4.3 Scheme 1

4.3.1 Scheme Setup. field $\mathbb{F}$ with prime p, generator g, hash functions $H(\cdot)$ is used in share generation and $H_B(\cdot)$ is used for hashing-to-bins, global randomnesses: i) key holder: K_1 and K_2; K_1 is just known by key holder, but K_2 is known by all, ii) the random numbers $r_1, \ldots, r_{t-1} \leftarrow \mathbb{F}_p$ generated by the key holder, that are fixed for all participants and their all elements. Participants use a homomorphic encryption scheme, Enc, with public and private keys pk, sk.

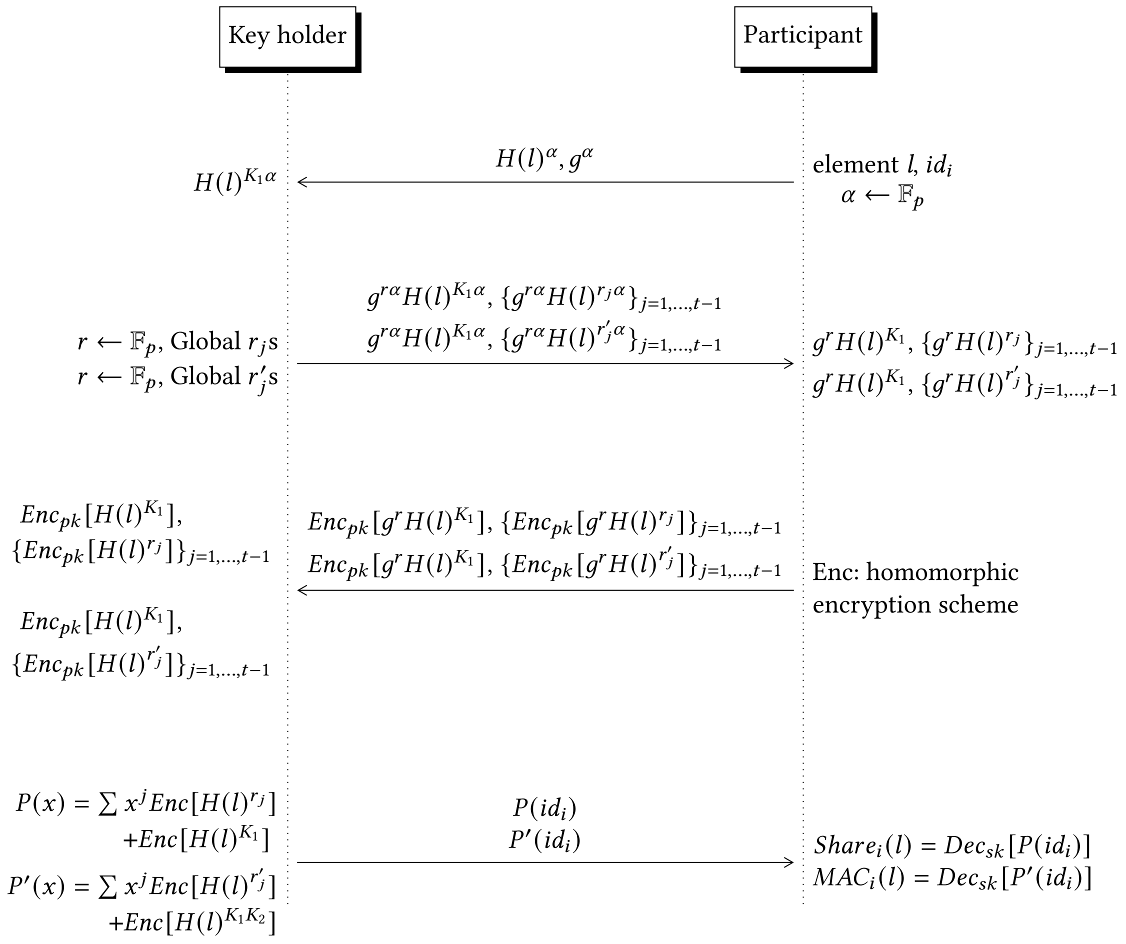
Key holder
Participant
$H(l)^\alpha, g^\alpha$
$H(l)^{K_1\alpha}$
element l, id_i
$\alpha \leftarrow \mathbb{F}_p$
$g^{r\alpha}H(l)^{K_1\alpha}, \{g^{r\alpha}H(l)^{r_j\alpha}\}_{j=1,\ldots,t-1}$
$g^{r\alpha}H(l)^{K_1\alpha}, \{g^{r\alpha}H(l)^{r'_j\alpha}\}_{j=1,\ldots,t-1}$
$r \leftarrow \mathbb{F}_p$, Global r_js
$r \leftarrow \mathbb{F}_p$, Global r'_js
$g^rH(l)^{K_1}, \{g^rH(l)^{r_j}\}_{j=1,\ldots,t-1}$
$g^rH(l)^{K_1}, \{g^rH(l)^{r'_j}\}_{j=1,\ldots,t-1}$
$Enc_{pk}[H(l)^{K_1}]$, $\{Enc_{pk}[H(l)^{r_j}]\}_{j=1,\ldots,t-1}$
$Enc_{pk}[H(l)^{K_1}]$, $\{Enc_{pk}[H(l)^{r'_j}]\}_{j=1,\ldots,t-1}$
$Enc_{pk}[g^rH(l)^{K_1}], \{Enc_{pk}[g^rH(l)^{r_j}]\}_{j=1,\ldots,t-1}$
$Enc_{pk}[g^rH(l)^{K_1}], \{Enc_{pk}[g^rH(l)^{r'_j}]\}_{j=1,\ldots,t-1}$
Enc: homomorphic encryption scheme
$P(x) = \sum x^j Enc[H(l)^{r_j}] + Enc[H(l)^{K_1}]$
$P'(x) = \sum x^j Enc[H(l)^{r'_j}] + Enc[H(l)^{K_1K_2}]$
$P(id_i)$
$P'(id_i)$
$Share_i(l) = Dec_{sk}[P(id_i)]$
$MAC_i(l) = Dec_{sk}[P'(id_i)]$

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

4.3.2 *Share Generation.*

(a) Share generation for each element: Party id_i generates a random number α to obliviously send its element's hash value, $H(l)$, to the key holder. The key holder generates a random number r for this session[1] to mask the secret value $H(l)^{K_1\alpha}$ and send it to the party. The key holder also uses t global random numbers $r_1, \ldots, r_{t-1} \leftarrow \mathbb{F}_p$ 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 $P(x)$. 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 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 $l \in \mathbb{L}$ to generate the corresponding share for each element owned by id_i.

(b) Share generation for the MAC of each element: 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

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

generating shares for the MAC of the distributed secret in (a). The purpose of these MACs is to verify the secrets that are reconsrtucted later in Section 1 by the reconstructor $\mathcal{R}$. The key holder uses a publicly known key K_2 to generate MACs for the elements' secrets.

4.3.3 Hashing-to-bins. Participants hash their shares to bins to reduce the computation cost for the reconstructor. Corresponding shares to an element l are stored in bin number $H_B(l)$. Each stored value is a quadruple consisting of i) the participant's identifier, id_i, ii) id_i's share for its element l, $Share_i(l)$, iii) the corresponding MAC, $MAC_i(l)$, and iv) the number of the bin, $H_B(l)$, that stores the information.

4.3.4 Reconstruction. After all participants stored 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 from distinct parties 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. 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 $H(l)^{k_1}$. Similarly, $\mathcal{R}$ calculates the corresponding MAC for the selected set of t shares mentioned earlier, which equals $H(l)^{k_1 K_2}$ if the set of shares correspond to the element l. As the key K_2 is publicly known, $\mathcal{R}$ can verify 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 1.

Algorithm 1 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 4.3.3)*/
4: for each Participant $id_i$; $i = 1, \ldots, m$ do
5:     for each Element $l$ owned by $id_i$; $l \in \mathbb{L}_i$ do
6:         $id_i$: store $(id_i, Share_i(l), MAC_i(l), H_B(l))$ in the bin number $H_B(l)$
7:
8: for each Bin $b_z$; $z = 1, \ldots, b$ do
9:     for each t-subset of quadruples in the bin $b_z$; $\{(id_{i_1}, Share_{i_1}(l), MAC_{i_1}(l), b_z), \ldots, (id_{i_t}, Share_{i_t}(l), MAC_{i_t}(l), b_z)\}$, where $id_{i_i}$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 $Secret_{MAC} == (Secret_{share})^{K_2}$ then Reveal that $(id_{i_1}, \ldots, idi_t)$ can reconstruct the $Secret_{share}$ which is $H(l)^{K_1}$
```

4.3.5 Complexity.

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 two rounds, regardless of the number of parties, the number of elements or the threshold. Hence Scheme 1 is $O(1)$. □

THEOREM 4.2. *The communication complexity of Scheme 1 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 1, 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 1 is $O(nmt)$. □

THEOREM 4.3. *The computation complexity of Scheme 1 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 1, 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. □

4.4 Scheme 2

4.4.1 Scheme Setup. In this scheme, the polynomial in Shamir secret sharing scheme is calculated in the exponent, the result is eliminating the need to use the homomorphic encryption scheme, Enc. This change reduces the communication cost of the scheme as described in Section 4.4.2. Similar to Scheme 1, we have the following set up parameters: field $\mathbb{F}$ with prime p, generator g, hash functions $H(\cdot)$ is used in share generation and $H_B(\cdot)$ is used for hashing-to-bins, global randomnesses: i) key holder: K_1 and K_2; K_1 is just known by key holder, but k_2 is known by all, ii) the random numbers $r_1, \ldots, r_{t-1} \leftarrow \mathbb{F}_p$ generated by the key holder, that are fixed for all participants and their all elements.

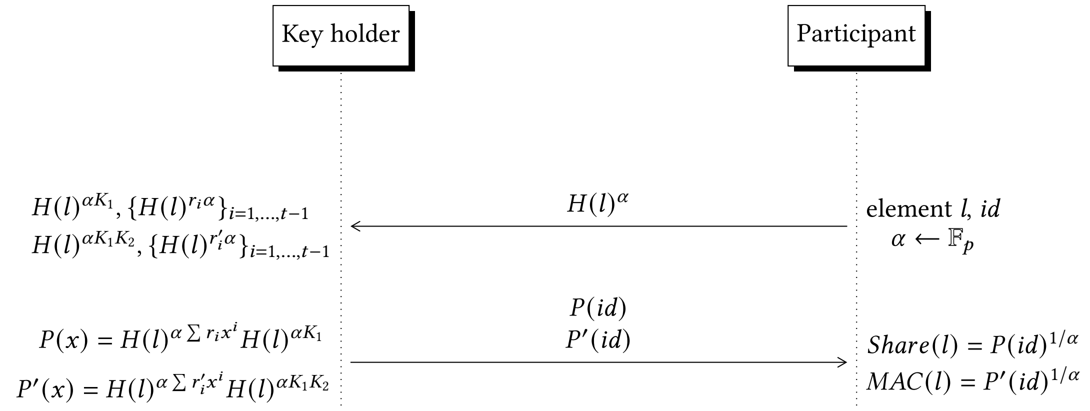
Key holder
Participant
$H(l)^{\alpha K_1}, \{H(l)^{r_i \alpha}\}_{i=1,\ldots,t-1}$
$H(l)^{\alpha K_1 K_2}, \{H(l)^{r'_i \alpha}\}_{i=1,\ldots,t-1}$
$H(l)^{\alpha}$
element l, id
$\alpha \leftarrow \mathbb{F}_p$
$P(x) = H(l)^{\alpha \sum r_i x^i} H(l)^{\alpha K_1}$
$P'(x) = H(l)^{\alpha \sum r'_i x^i} H(l)^{\alpha K_1 K_2}$
$P(id)$
$P'(id)$
$Share(l) = P(id)^{1/\alpha}$
$MAC(l) = P'(id)^{1/\alpha}$

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

4.4.2 Share Generation.

(a) Share generation for the element: Party id_i generates a random number α to obliviously send an element's hash, $H(l)$, to the key holder. The key holder generates global random numbers $r_1, \ldots, r_{t-1} \leftarrow \mathbb{F}_p$. Then it communicates with each participant to generate shares for each participant's element using Shamir secret sharing scheme, by forming a polynomial in the exponent. 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 $l \in \mathbb{L}$ 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. Similar to the procedure in Section 4.3.2(b), the key holder in scheme 2 uses a publicly known key K_2 to generate MACs for the elements' 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 sceheme 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 participant– reconstructs the secrets in each bin for every $\binom{m}{t}$ subset of shares for distinct parties, $id_{i_1}, \ldots, id_{i_t}$. 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: $H(l)^{k_1} = \prod_{w=0}^{t} Share_{i_w}(\cdot)^{(\prod_{w' \neq w} \frac{-id_{w'}}{id_w - id_{w'}})}$. $\mathcal{R}$ 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 2.

4.4.5 Complexity.

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 one rounds, regardless of the number of parties, the number of elements or the threshold. Hence Scheme 1 is $O(1)$. □

Algorithm 2 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 4.3.3)*/
 4: for each Participant $id_i$; $i = 1, \ldots, m$ do
 5:     for each Element $l$ owned by $id_i$; $l \in \mathbb{L}_i$ do
 6:         $id_i$: store $(id_i, Share_i(l), MAC_i(l), H_B(l))$ in the bin number $H_B(l)$
 7:
 8: for each Bin $b_z$; $z = 1, \ldots, b$ do
 9:     for each t-subset of quadruples in the bin $b_z$; $\{(id_{i_1}, Share_{i_1}(l), MAC_{i_1}(l), b_z), \ldots, (id_{i_t}, Share_{i_t}(l), MAC_{i_t}(l), b_z)\}$, 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 $Secret_{MAC} == (Secret_{share})^{K_2}$ then Reveal that $(id_{i_1}, \ldots, id_{i_t})$ can reconstruct the $Secret_{share}$ which is $H(l)^{K_1}$
```

THEOREM 4.5. *The communication complexity of Scheme 2 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 2, 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 1 is $O(nm)$. □

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. The proof follows the same reasoning as the proof of Theorem 4.3. As described in Algorithm 2, the reconstructor $\mathcal{R}$ forms all possible combinations of t shares from distinct parties in each bin. There are $\binom{m}{t}$ 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. □

4.5 Additional Notes

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

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. They key holders run a multi-party protocol among themselves to form the followings from their keys and random numbers: i) a group key, ii) group random numbers. The group key serves as the key holder key K_1 in the Schemes 1 and 2 (Figures 1 and 2), and the group random numbers serve as r, r' and $r_j, r'_j; j = 1, \ldots, t-1$ in the Scheme 1 and as $r_j, r'_j; j = 1, \ldots, t-1$ in Scheme 2. The multi key holder scheme follows the exact same procedure as in Schemes 1 and 2 otherwise.

5 SS-OPRF

5.1 Security Definition

Obliviousness
Random PRF Output
No reconstruction even given PRF

5.2 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

5.3 Security Proof

5.4 SS-OPRF 2

My construction, 2 round protocol
$A- > S : g^r, x^r$
$S- > A : (g^r)^{\log s} x^{rc}$
$A- > S : E(((g^r)^{\log s} x^{rc})^{1/r}) = E(sx^c)$
$S- > A : E(SS(x^c))$

5.5 Security Proof

6 COMPLETING THE PROTOCOL

6.1 Verifying the Reconstruction of Shares

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

6.2 Distributing S

Share $c = c_1 c_2 c_3 ...$
Also distribute decryption

6.3 Reducing the number of possible share combinations

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

7 EVALUATION

7.1 SS-OPRF

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

7.2 Reconstruction

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

8 RELATED WORK

Kissner, Song
Generic Secure Computation, PSI via Secure computation
PSI protocols based on OPRF
PSI protocols based on hashing

9 CONCLUSION

REFERENCES

[1] Ivan Damgård and Mads Jurik. A generalisation, a simpli.cation and some applications of paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography*, pages 119–136, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

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

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

[4] 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.

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