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:

- $O(1)$ 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 and Damgard Jurik Crytposystems

4 PROTOCOL OVERVIEW

4.1 Notations and Assumptions

- m = the number of parties each identified by id_i; $i = 1, \cdots, m$
- n = max size of the set of elements, $\mathbb{L}_i$, owned by id_i; $i = 1, \cdots, m$
- t = threshold (intersection and the secret sharing threshold)
- b = number of bins; $b_1, \cdots, b_b$
- Each bin is padded to max
- Key holder is semi-honest
- Key holder is not a party
- Key holder sees at most t-1 shares

4.2 Scheme 1

Set up: 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 $r_1, \cdots, r_{t-1} \leftarrow \mathbb{F}_p$ generated by the key holder, that are fixed for all participants and their all elements. Participants use a homomorphic encryption scheme, Enc, with pubic and private keys pk, sk.

4.2.1 Share Generation. The key holder generates random numbers $r_1, \cdots, r_{t-1} \leftarrow \mathbb{F}_p$. Then it communicates with each participant to generate shares for each participant's element using Shamir secret sharing scheme. Figure 1 shows the protocol taking place between the key holder and a participant with identifier identifier id_i, $i = 1, \cdots, m$, 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.

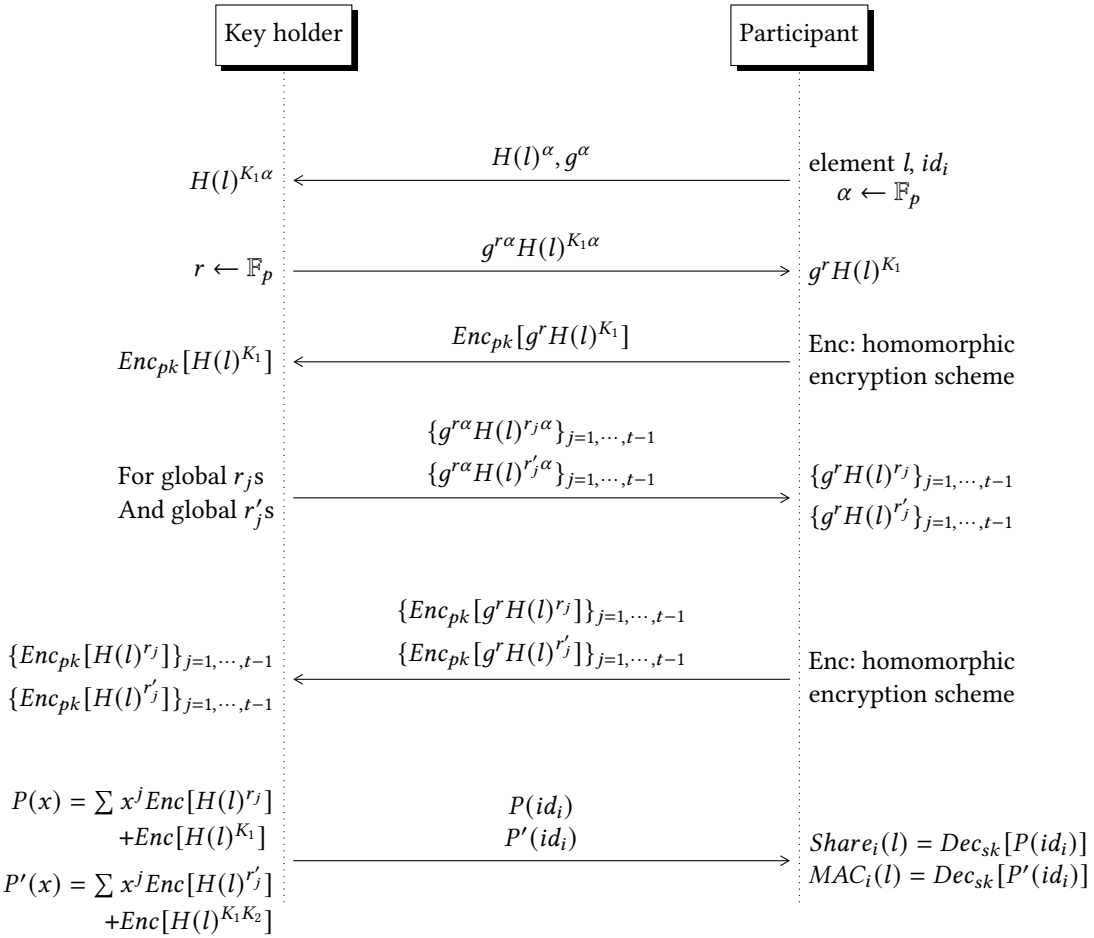

Figure 1: Communication between the key holder and a participant id_i in scheme 1 - Share generation for an element $l \in \mathbb{L}_i$, owned by id_i

4.2.2 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.

4.2.3 Reconstruction. After each participant stored their shares in the corresponding bins, the reconstructor $\mathcal{R}$ – who is also a participant – reconstructs secrets in each bin. For each $\binom{m}{t}$ subset of the participants, $id_{i_1}, \cdots, id_{i_t}$, $\mathcal{R}$ collects the shares

and reconstructs the corresponding secret by applying Lagrange interpolation: $H(l)^{k_1} = \sum_{w=0}^{t} Share_{i_w}(\cdot)(\prod_{w' \neq w} \frac{-id_{i_{w'}}}{id_{i_w} - id_{i_{w'}}})$. Reconstruction's steps are summarized in the Algorithm 1.

Algorithm 1 RECONSTRUCT$_{\text{SCHEME1}}$

/* K_1 is key holder's secret key and K_2 is a publicly known key used for MAC generation */
/* $H(\cdot)$ is a hash function in share generation */
/* $H_B(\cdot)$ is a hash functions used for hashing-to-bin (Section 4.2.2) */
for each Participant id_i; $i = 1, \cdots, m$ **do**
  for each Element l owned by id_i; $l \in \mathbb{L}_i$ **do**
    id_i: store $(id_i, Share_i(l), MAC_i(l), H_B(l))$ in the bin number $H_B(l)$

for each Bin b_z; $z = 1, \cdots, b$ **do**
  for each t-subset of quadruples in the bin b_z; $\{(id_{i_1}, Share_{i_1}(l), MAC_{i_1}(l), b_z), \cdots, (id_{i_t}, Share_{i_t}(l), MAC_{i_t}(l), b_z)\}$ **do**
    $\mathcal{R}$: Apply Lagrange interpolation to find the corresponding intercept, $Secret_{share}$, for the polynomial that covers the points $\{(i_1, Share_{i_1}), \cdots, (i_t, Share_{i_t})\}$
    $\mathcal{R}$: Apply Lagrange interpolation to find the corresponding intercept, $Secret_{MAC}$, for the polynomial that covers the points $\{(i_1, MAC_{i_1}), \cdots, (i_t, MAC_{i_t})\}$
    if $Secret_{MAC} == (Secret_{share})^{K_2}$ **then** Reveal that $(id_{i_1}, \cdots, idi_t)$ can reconstruct the $Secret_{share}$ which is $H(l)^{K_1}$

4.2.4 *Complexity.*

THEOREM 4.1. *The communication complexity of Scheme 1 is* $\cdots$

PROOF. □

THEOREM 4.2. *The computation complexity of Scheme 1 is* $\cdots$

PROOF. □

4.3 Scheme 2

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 4.3.1. 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 $r_1, \cdots, r_{t-1} \leftarrow \mathbb{F}_p$ generated by the key holder, that are fixed for all participants and their all elements.

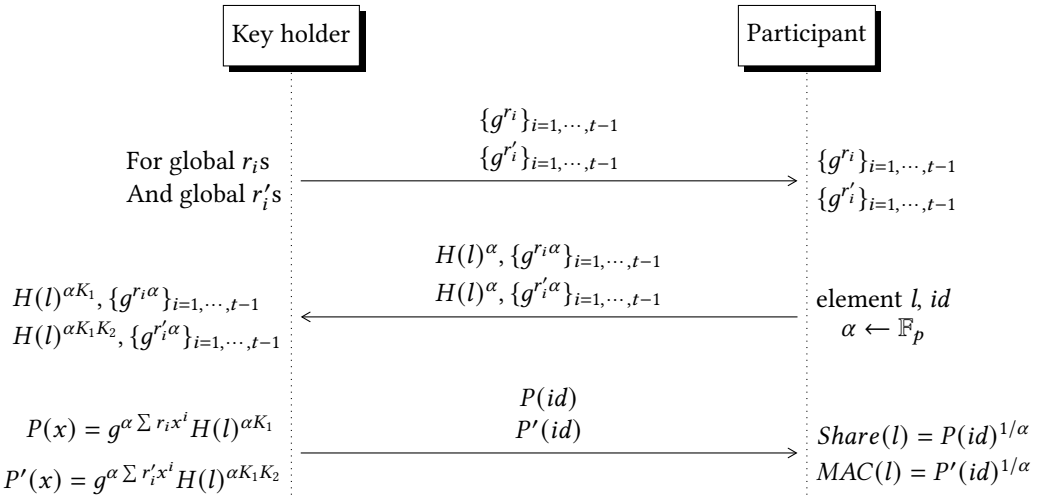

Figure 2: Communication between the key holder and a participant id_i in scheme 2 - Share generation for an element $l \in \mathbb{L}_i$, owned by id_i

4.3.1 Share Generation. The key holder generates random numbers $r_1, \cdots, r_{t-1} \leftarrow \mathbb{F}_p$. 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, $i = 1, \cdots, m$, 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.

4.3.2 Hashing-to-bins. This step is identical to the hashing-to-bins described in Section 4.2.2.

4.3.3 Reconstruction. Similar to the reconstruction in Section 4.2.3 for Scheme 1, each participant stores their shares in the corresponding bins. Then the reconstructor $\mathcal{R}$ –who is also participant– reconstructs the secrets in each bin for every $\binom{m}{t}$ subset of the participants, $id_{i_1}, \cdots, id_{i_t}, \mathcal{R}$. 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: $H(l)^{k_1} = \prod_{w=0}^{t} Share_{i_w}(\cdot)^{(\prod_{w' \neq w} \frac{-id_{w'}}{id_w - id_{w'}})}$.

Algorithm 2 RECONSTRUCT$_{\text{SCHEME2}}$

/* K_1 is key holder's secret key and K_2 is a publicly known key used for MAC generation */
/* $H(\cdot)$ is a hash function in share generation */
/* $H_B(\cdot)$ is a hash functions used for hashing-to-bin (Section 4.2.2) */
for each Participant id_i; $i = 1, \cdots, m$ **do**
  for each Element l owned by id_i; $l \in \mathbb{L}_i$ **do**
    id_i: store $(id_i, Share_i(l), MAC_i(l), H_B(l))$ in the bin number $H_B(l)$

for each Bin b_z; $z = 1, \cdots, b$ **do**
  for each t-subset of quadruples in the bin b_z; $\{(id_{i_1}, Share_{i_1}(l), MAC_{i_1}(l), b_z), \cdots, (id_{i_t}, Share_{i_t}(l), MAC_{i_t}(l), b_z)\}$ **do**
    $\mathcal{R}$: Reconstruct the corresponding secret, $Secret_{share}$, to the points $\{(i_1, Share_{i_1}), \cdots, (i_t, Share_{i_t})\}$
    $\mathcal{R}$: Reconstruct the corresponding secret, $Secret_{MAC}$, to the points $\{(i_1, MAC_{i_1}), \cdots, (i_t, MAC_{i_t})\}$
    if $Secret_{MAC} == (Secret_{share})^{K_2}$ **then** Reveal that $(id_{i_1}, \cdots, idi_t)$ can reconstruct the $Secret_{share}$ which is $H(l)^{K_1}$

4.3.4 Complexity.

THEOREM 4.3. *The communication complexity of Scheme 2 is* $\cdots$.

PROOF. □

THEOREM 4.4. *The computation complexity of Scheme 2 is* $\cdots$.

PROOF. □

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 $c = c_1c_2c_3...$
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