Practical Over-Threshold Multi-Party Private Set Intersection

Annual Computer Security Applications Conference (ACSAC 2020)

Rasoul Akhavan Mahdavi, Thomas Humphries, Bailey Kacsmar, Simeon Krastnikov, Nils Lukas, John A. Premkumar, Masoumeh Shafieinejad, Simon Oya, Florian Kerschbaum, Erik-Oliver Blass
Motivation

- Private Set Intersection (PSI)
  - Problem Description
  - Well studied problem
  - A variety of solutions
    - Multiple parties
    - Unbalanced set sizes

- Over-threshold Private Set Intersection
  - Problem Description
  - Parameterized by m, n and t
  - No Practical Deployments
Motivation

- **Private Set Intersection (PSI)**
  - Problem Description
  - Well studied problem
  - A variety of solutions
    - Multiple parties
    - Unbalanced set sizes

- **Over-threshold Private Set Intersection**
  - Problem Description
  - Parameterized by m, n and t
  - No Practical Deployments
Motivation

- Use cases for Over Threshold PSI
  - Network operation centers collaborate to identify threats
  - Compare indicators of compromise to find common threats
  - Can’t release the indicators (privacy)
  - Intersection isn’t enough (over-threshold)
**Related Work**

- $m$: number of parties
- $n$: maximum set size
- $t$: threshold

<table>
<thead>
<tr>
<th></th>
<th>Communication Rounds</th>
<th>Communication Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kissner &amp; Song</td>
<td>$O(m)$</td>
<td>$O(nm^3)$</td>
</tr>
<tr>
<td>Secret Share MPC</td>
<td>$O(\log^2(nm))$</td>
<td>$O(nm^3 \log^2(nm) + nm^3 t)$</td>
</tr>
<tr>
<td>t-PSI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Related Work

$m$: number of parties  
$n$: maximum set size  
$t$: threshold  
$k$: number of keyholders/collusion threshold

<table>
<thead>
<tr>
<th></th>
<th>Communication Rounds</th>
<th>Communication Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kissner &amp; Song</td>
<td>$O(m)$</td>
<td>$O(nm^3)$</td>
</tr>
<tr>
<td>Secret Share MPC</td>
<td>$O(\log^2(nm))$</td>
<td>$O(nm^3 \log^2(nm) + nm^3t)$</td>
</tr>
<tr>
<td>t-PSI</td>
<td>$O(1)$</td>
<td>$O(nmtk)$</td>
</tr>
</tbody>
</table>
Our Work

● A new primitive: Oblivious Pseudo-Random Secret Sharing

● Our Over-Threshold PSI protocol
  ○ First attempt: $t$-PSI$_0$
    ■ Less communication
    ■ Impractical Computation
  ○ More practical approach: $t$-PSI
    ■ More communication
    ■ Practical Computation

● Experiments and Evaluation
  ○ Scalability for number of parties and set size
Oblivious Pseudorandom Secret Sharing

Oblivious Pseudorandom Function

Keyholder

User
Oblivious Pseudorandom Secret Sharing

Oblivious Pseudorandom Function

Keyholder doesn’t learn X

User doesn’t learn k
Oblivious Pseudorandom Secret Sharing Protocol

Keyholder doesn’t learn X

User doesn’t learn k
Oblivious Pseudorandom Secret Sharing Protocol

Keyholder doesn’t learn X

User doesn’t learn k

Corresponding Reconstruction Procedure

$SS_{PRF_k}(X)$ (Secret)
Oblivious Pseudorandom Secret Sharing

\[ m = 6, \ t = 4 \]
Oblivious Pseudorandom Secret Sharing

$m = 6, t = 4$

t out of m threshold secret sharing scheme
Oblivious Pseudorandom Secret Sharing

$m = 6, t = 4$
Oblivious Pseudorandom Secret Sharing

\[ m = 6, t = 4 \]
Oblivious Pseudorandom Secret Sharing Protocol

Keyholder doesn’t learn X

User doesn’t learn k

Multiple Keyholders Allowed!
OPR-SS Variants

First Variant
- Simpler share generation process
- 1 round of communication
- $O(k)$ communication complexity
- A very costly reconstruction
- Used in $t$-PSI$_0$

Second Variant
- More complex share generation process
- 2 round of communication
- $O(kt)$ communication complexity
- Much more efficient reconstruction
  (up to 1000 fold speedup)
- Used in $t$-PSI

$t$: threshold
$k$: number of keyholders
OT-MP-PSI Setup

- Participants (m)
  - Owners of data
- Keyholders (k)
  - Owners of a joint key
- Reconstructors
  - With computation power

- Overlaps are allowed
- All entities are semi-honest
OT-MP-PSI Protocol

- Share Generation
- Hashing-to-bins
- Reconstruction
OT-MP-PSI Protocol

- Share Generation
- Hashing-to-bins
- Reconstruction

For each participant:
   For each element they own:
      Generate an OPR-SS for that element

\[ t\text{-PSI}_0 \] : use first variant
\[ t\text{-PSI} \] : use second variant
OT-MP-PSI Protocol

- Share Generation

- Hashing-to-bins
  - Common technique
  - Done by each user separately
  - Number of bins is a tradeoff

- Reconstruction

- Predefined # of bins (b)
- Assign each element to bin (using hash)
- Pad the bins to max_size
OT-MP-PSI Protocol

- Share Generation
- Hashing-to-bins
- Reconstruction
  - Can be parallelized

For each bin number:
  For each combination of $t$ participants:
    For each combination of shares:
      Run reconstruction procedure
      if (success): report to participants

$t$-PSI $^0$: reconstruction for first variant (slow)
$t$-PSI $^1$: reconstruction for second variant (fast)
Experiments

Scalability for set sizes

Scalability for number of participants
Experiments

Scalability for set sizes

Scalability for number of participants
Experiments

Scalability for set sizes

Scalability for number of participants

[Graphs showing scalability for set sizes and number of participants]
Conclusion

● A new primitive, OPR-SS, for generating pseudorandom shares of a secret

● $t$-PSI $0$
  ○ 1 round communication and $O(nmk)$ communication complexity
  ○ Expensive Reconstruction Stage

● $t$-PSI
  ○ 2 round communication and $O(nmk^t)$ communication complexity
  ○ Cheaper Reconstruction Stage

● Experiments and Evaluation
  ○ Scalability up to millions of elements and thousands of clients
  ○ Public code on github ([www.github.com/cryspuwaterloo/OT-MP-PSI](http://www.github.com/cryspuwaterloo/OT-MP-PSI))
Thank you!

Questions?