A Survey on Private Set Intersection

Presented by Hongrui Cui

RickFreeman@sjtu.edu.cn

October 17, 2019

Overview

Content

Private Set Intersection

Research Background

- Multiparty computation of set intersection

Functionality Classification

- Security: Semi-Honest/Malicious
- Players: Two Party/Multi Party
- Output: Plain Intersection/Post-Processing

Literature of Private Set Intersection

Paper	Parties	Security	Building Blocks
[PSZ14]	2	Semi-Honest	OT(OPRF)
[HEK12]	2	Semi-Honest	GC,GMW
[CHLR18]	2	Hybrid	(leveled-)FHE
[RR17]	2	Malicious	OT(OPRF)
[KMP$^+$17]	n	Semi-Honest	OT(OPPRF)

Table: Comparison of Different Private Set Intersection Protocols

Notations

PSI Notations:

- $X, Y \subset \{0,1\}^{\sigma}$: Input sets
- $X^*, Y^* \subset \{0,1\}^{\lambda+\log(|X|)+\log(|Y|)}$: Processed input sets
- $\binom{m}{1} - OT_v^k$: k instances of m-choose-1 oblivious transfer on v-bit strings
- $\mathcal{F}_{\text{PSM}}$: Private set membership protocol (i.e. $y \in X$)

Notations

Cuckoo Hashing Notations:

- $\mathfrak{B}$: Hash table "bins"
- $m \in \mathbb{N}$: Hash table size
- $h_1, h_2, h_3 : \{0,1\}^* \to [m]$: Hash function

A Naïve PSI Protocol

Compute Intersection on Hashed Values

$$
\begin{array}{ccc}
\textit{Sender} & & \textit{Receiver} \\
& \xrightarrow{X^* := \{H(x) | x \in X\}} & \text{Output } X \cap Y := \\
& & \{y \in Y | H(y) \in X^*\} \\
& \xleftarrow{X \cap Y \text{ (optionally)}} & \\
\text{Output } X \cap Y & &
\end{array}
$$

A Naïve PSI Protocol

Why Naïve

- Hashed set X^* has the same entropy as X
- This entropy is usually low
- Feasible brute-force attack

A Naïve PSI Protocol

Why Naïve

- Hashed set X^* has the same entropy as X
- This entropy is usually low
- Feasible brute-force attack

When the entropy is acceptable (e.g. 80 bits), this is secure.

Content

Semi-Honest PSI

- 2-Party Semi-Honest PSI receives most attention
- State-of-the-art only incurs $1 - 10$ times overhead

Cuckoo Hashing

- A special hashing function
- Using *eviction* to resolve collision

Cuckoo Hashing

Insertion

- Let $i = 1$, compute index $l = h_i(x)$
- If $\mathfrak{B}[l] = \perp$, then insert $\langle x, i \rangle$
- If not, insert anyway
- Let $\langle y, j \rangle$ be the original content, let $x := y$ $i \xleftarrow{\$} [3] \setminus \{j\}$, goto step 1

If the process iterates more than t times, put the item in a *stash* s.

Cuckoo Hashing

Insertion

- Let $i = 1$, compute index $l = h_i(x)$
- If $\mathfrak{B}[l] = \perp$, then insert $\langle x, i \rangle$
- If not, insert anyway
- Let $\langle y, j \rangle$ be the original content, let $x := y$ $i \xleftarrow{\$} [3] \setminus \{j\}$, goto step 1

If the process iterates more than t times, put the item in a *stash* s.

Lookup

- For inserted item x, there are only $3 + |s|$ possible locations

Cuckoo Hashing

Receiver: "Thin" Table
Cuckoo Hashing with h_1, h_2

Sender: "Thick" Table
Regular Hashing with h_1, h_2

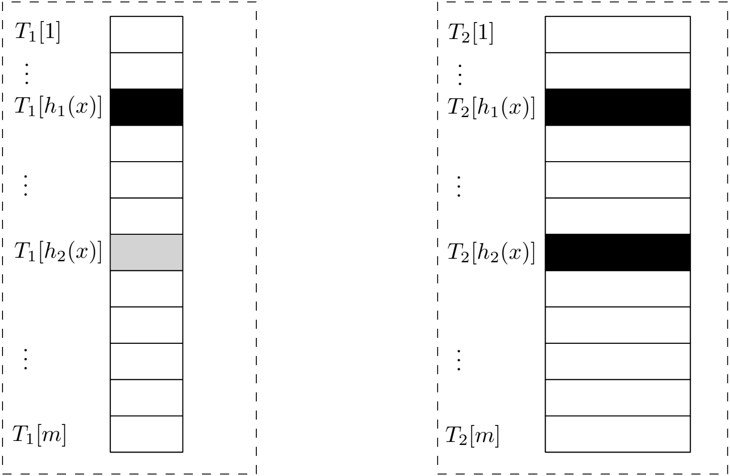

Figure: Cuckoo Hash Table

The Paradigm of [PSZ14]

$\mathcal{F}_{\textbf{PSI}} \leq \mathcal{F}_{\textbf{PSM}}$

- Receiver does cuckoo hashing, while the sender does regular hashing
- They then perform m instances of $\mathcal{F}_{\text{PSM}}$ $(m = |\mathfrak{B}|)$

The Paradigm of [PSZ14]

$\mathcal{F}_{\textbf{PSI}} \leq \mathcal{F}_{\textbf{PSM}}$

- Receiver does cuckoo hashing, while the sender does regular hashing
- They then perform m instances of $\mathcal{F}_{\text{PSM}}$ $(m = |\mathfrak{B}|)$

Discussion

- Most works in the semi-honest model follow this paradigm
- Various means to implement $\mathcal{F}_{\text{PSM}}$, e.g. OT, FHE, GC/GMW
- Cuckoo Hashing may be inherently unsuitable for malicious world

OT as OPRF

- $\mathcal{F}_{\mathsf{PSM}}$ from Oblivious PRF is quite easy
- (One-Time) Oblivious PRF can be considered some $\binom{2^\sigma}{1} - ROT$
- OT-Extension can efficiently implement this primitive

A Brief Review on OT-Extension

The idea is to “bootstrap” a large number of OT instances from a small number of base OT’s.

Sender		*Receiver*
$b \xleftarrow{\$} \{0,1\}^v$		$T_0, T_1 \xleftarrow{\$} \{0,1\}^{m\times v}$
$\xrightarrow{\quad b_j \quad}$	$\boxed{\binom{2}{1} - OT_m^v}$	$\xleftarrow{\quad (T_{0,j}, T_{1,j}) \quad}$
$\xleftarrow{\quad T_{b,j} \quad}$		
	$\xleftarrow{C^i = T_o^i \oplus T_1^i \oplus \mathrm{ECC}(w_i)}$	
$Q^i = T_b^i \oplus s \cdot C^i$		
Output (s, Q^i)		Output $H(i \| T_0^i)$

Set Membership from Homomorphic Encryption

Naive Approach

Sender		*Receiver*
	$\xleftarrow{\mathsf{Enc}(pk,y)}$	
$r \xleftarrow{\$} R_q$	$\xrightarrow{c=\mathsf{Eval}(r\cdot\prod_{x\in X}(y-x))}$	
		Output 1 if $\mathsf{Dec}(sk, c) = 0$ Output 0 otherwise

Set Membership from Homomorphic Encryption

Naive Approach

Sender		*Receiver*
	$\xleftarrow{\mathsf{Enc}(pk,y)}$	
$r \xleftarrow{\$} R_q$	$\xrightarrow{c=\mathsf{Eval}(r\cdot\prod_{x\in X}(y-x))}$	
		Output 1 if $\mathsf{Dec}(sk, c) = 0$ Output 0 otherwise

Several Optimizations

- Batching: reduce communication by n/d
- Partitioning: reduce polynomial degree by α
- Windowing: reduce circuit depth logarithmically
- Pre-Processing: reduce circuit depth by 1

Set Membership from General Framework

The main advantage is arbitrary post-processing can be applied (by concatenation of circuits), but shuffling the output may be needed.

Content

Malicious PSI via Dual Execution

Ideas of [RR17]:

Sender		*Receiver*
Randomly Permute X		Randomly Permute Y
$\xrightarrow{x}$		
$\xleftarrow{[x]_i}$	$\mathcal{F}_{\text{OPRF}}$	$\xrightarrow{k_i}$
		$\xleftarrow{y}$
$\xleftarrow{k'_i}$	$\mathcal{F}_{\text{OPRF}}$	$\xrightarrow{[y]'_i}$
	$\xrightarrow{Q := \{[x]_{i,j} = [x]_i \oplus [x]'_j\}}$	
		Output $X \cap Y = \{y \mid \exists i, [y]_i \oplus [y]'_j \in Q\}$

Optimizations

It is possible to use regular hashing to reduce the quadratic complexity:

- Assuming n bins, $\log(n)$ items per bin, the complexity is $n \log(n)^2$
- Cuckoo hashing cannot be used here

Content

Multiparty PSI

The authors of [KMP$^+$17] proposed a simple protocol for semi-honest, multiparty PSI:

- Zero-Sharing
- Reconstruction

Multiparty PSI

The authors of [KMP$^+$17] proposed a simple protocol for semi-honest, multiparty PSI:

- Zero-Sharing
- Reconstruction

The protocol heavily uses the *Oblivious Programmable PRF* functionality, which can be implemented from $\mathcal{F}_{\mathsf{OPRF}}$ and polynomial interpolation.

Multiparty PSI

For every pair of parties P_i, P_j:

P_i

chooses $s_k^{i,1}, \ldots, s_k^{i,n}$

such that $\bigoplus_l s_k^{i,l} = 0$

$$\xrightarrow{\{(x_k^i, s_k^{i,j})\}_k}$$

$$\xleftarrow{k_{i,j}}$$

$$\boxed{\mathcal{F}_{\mathrm{OPPRF}}}$$

P_j

chooses $s_k^{j,1}, \ldots, s_k^{j,n}$

such that $\bigoplus_l s_k^{j,l} = 0$

$$\xleftarrow{\{x_k^j\}}$$

$$\xrightarrow{\{s_k^{i,j}\}}$$

$$s_k^j = \bigoplus_i s_k^{i,j}$$

Multiparty PSI

Note that

- if $x \in \bigcap_i X^i$
- then $\bigoplus_j s_k^j = 0$

Multiparty PSI

Note that

- if $x \in \bigcap_i X^i$
- then $\bigoplus_j s_k^j = 0$

Reconsturction

- The n parties agree on a dealer, e.g. P_1
- The party P_i uses (x_k^i, s_k^i) to program a PRF
- P_1 interacts with these parties and gets the sharings
- If $x \in X^1$ is in the intersection, then the $n-1$ results from $\mathcal{F}_{\text{OPPRF}}$ with s_k^1 (assuming $x = s_k^1$) should form an additive sharing of 0

Reference I

Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal.
Labeled PSI from fully homomorphic encryption with malicious security.
In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1223–1237, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

Yan Huang, David Evans, and Jonathan Katz.
Private set intersection: Are garbled circuits better than custom protocols?
In *ISOC Network and Distributed System Security Symposium – NDSS 2012*, San Diego, CA, USA, February 5–8, 2012. The Internet Society.

Reference II

Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu.
Practical multi-party private set intersection from symmetric-key techniques.
In Thuraisingham et al. [TEMX17], pages 1257–1272.

Benny Pinkas, Thomas Schneider, and Michael Zohner.
Faster private set intersection based on OT extension.
In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 797–812, San Diego, CA, USA, August 20–22, 2014. USENIX Association.

Peter Rindal and Mike Rosulek.
Malicious-secure private set intersection via dual execution.
In Thuraisingham et al. [TEMX17], pages 1229–1242.

Reference III

Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors.
ACM CCS 2017: 24th Conference on Computer and Communications Security, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

Thank You