FAST PRIVATE SET INTERSECTION
FROM
FULLY HOMOMORPHIC ENCRYPTION

CCS 2017
Private Set Intersection (PSI)
Private Set Intersection (PSI)

“Sender”

Ideal World

“Receiver”

$X \cap Y$
A Sampling of PSI Over the Decades

\[ x^a \beta = y^\beta \alpha \]
\[ \Rightarrow x = y \]
A Sampling of PSI Over the Decades

\[ Q(x) := (x - y) \]

\[ Q(x) = 0 \]

\[ \Rightarrow x = y \]
A Sampling of PSI Over the Decades
A Sampling of PSI Over the Decades


- [Meadows86]: Private equality test
- [NaorPinkas99]: Semi-honest PSI
- [DeCristofaroKimTsudik10]: Malicious secure
- [FreedmanNissimPinkas04]: Hash table base PSI
- [DachmanMalkinRaykovaYung09]: Malicious secure
- [HuangEvansKatz12]: Garbled Circuit base PSI
- [DongChenWen13]: Oblivious Transfer & Bloom filter base PSI
- [RR17a]: Malicious Oblivious Transfer & Bloom filter base PSI
- [HubermanFranklinHog99]: Private equality test to PSI
- [DachmanMalkinRaykovaYung09]: Malicious secure
A Sampling of PSI Over the Decades

- [Meadows86] First to define private equality test using Diffie-Hellman
- [HubermanFranklinHogg99] Extended Diffie-Hellman private equality test to PSI
- [NaorPinkas99] Oblivious Transfer base PSI using Polynomial Evaluation
- [FaginNaorWinkler96] Bitwise Oblivious Transfer encoding for private equality test
- [FreedmanNissimPinkas04] Homomorphic Enc base PSI using Polynomial Evaluation and hashing
- [DachmanMalkinRaykovaYung09] Homomorphic Enc base PSI using Polynomial Evaluation
- [DeCristofaroKimTsdik10] Diffie-Hellman base PSI
- [DongChenWen13] Oblivious Transfer + Bloom filter base PSI
- [KolesnikovKumaresanRosulekTrieu16] Element-wise Oblivious Transfer encoding PSI
- [PinkasSchneiderZohner14] Cuckoo hashing + Bitwise Oblivious Transfer encoding PSI
- [HuangEvansKatz12] Garbled Circuit base PSI
- [PinkasSchneiderZohner14] Cuckoo hashing + Bitwise Oblivious Transfer encoding PSI
- [KolesnikovKumaresanRosulekTrieu16] Element-wise Oblivious Transfer encoding PSI
- [FreedmanNissimPinkas04] Homomorphic Enc base PSI using Polynomial Evaluation and hashing
App: Contact discovery

Users → PSI → Contacts

WhatsApp Contacts → PSI → WhatsApp
App: Contact discovery

\[ |X| \gg |Y| \]

\[ X \cap Y \]

Notation:
- \[ N = |X| \]
- \[ n = |Y| \]
Shortcomings of Prior Work

- Communication linear in both sets \( O(|X| + |Y|) \)
  - What about \( |X| \gg |Y| \)?
  - Insecure solution:
    - Send small set to other party
    - Comm. = \( O(\min(|X|, |Y|)) \)

- Can we match this?
  - Almost…

*Some prior works achieve sublinear communication for related problems.*
Shortcomings of Prior Work

- Communication linear in both sets $O(|X| + |Y|)$
  - What about $|X| \gg |Y|$?
  - Insecure solution:
    - Send small set to other party
    - Comm. = $O(\min(|X|, |Y|))$

- Can we match this?
  - Almost…

- Computation = $O(|X|)$
- Communication = $O(|Y| \log |X|)$

*Some prior works achieve sublinear communication for related problems.*
Fully Homomorphic Encryption (FHE)

- Encryption technique that allows computation
  - \( \text{Enc}_k(f(x)) \equiv f(\text{Enc}_k(x)) \)
  - \( f \) can perform \(+, -, *\)

- Addition and subtraction are very cheap.
- Multiplication is very expensive.
  - Limited multiplication depth
  - E.g. \( f(x) = \prod_{i=1}^{8} x_i \)
  - Inefficient beyond depth \(~6\)
Equality Test from FHE

• Want to test if $y = x$

$$[y] := \text{Enc}_k(y)$$

$$[z] := [y] - x$$

• Issue: Receiver can recover $x = y - z$!
  • Need to randomize $z$

$$z = 0 \iff y = x$$
Equality Test from FHE

- Want to test if \( y = x \)

\[
\begin{align*}
\text{Sample } r & \leftarrow \mathbb{Z}_p^* \\
\llbracket y \rrbracket & := \text{Enc}_k(y) \\
\llbracket z \rrbracket & := (\llbracket y \rrbracket - x)r \\
\text{Given } x & \neq 0,\ z = 0 \iff y = x
\end{align*}
\]

- Issue: Receiver can recover \( x = y - z! \)
  - Need to randomize \( z \)
  - Elements are in the prime field \( \mathbb{Z}_p = \{0, 1, \ldots, p - 1\} \)
  - For a random \( r \in \mathbb{Z}_p^* = \{1, \ldots, p - 1\} \)
    - \( xr \) is a random elements in \( \mathbb{Z}_p^* \), given non-zero \( x \)

[ChenLaineRindal17]
Membership from FHE

• Want to test if \( y \in X \)

\[
\text{Sample } r \leftarrow \mathbb{Z}_p^* \\
[z] := r \prod_{x \in X} ([y] - x)
\]

\[
[y] := \text{Enc}_k(y)
\]

\[
z = 0 \iff y \in X
\]

• Issue: Depth of the computation is \( \log N = \log |X| \)
  • E.g. \( N = 2^{28} \Rightarrow \text{depth} = 28 > 6 \)

• Observe the polynomial
  • Symmetric poly. \(\Rightarrow\) efficiently computable

• Need to compute \( y^N \) in low degree

\[
[z] := f(y) = r \prod_{x \in X} (y - x) \\
= a_N y^N + \cdots + a_2 y^2 + a_1 y + a_0
\]
Windowing: computing $y^N$ in low depth

- Need to compute $[Z] := a_N y^N + \ldots + a_2 y^2 + a_1 y + a_0$

- Depth $\log N$ solution, send $[y]$ and compute:
  - $[y^2] = [y][y]$
  - $[y^4] = [y^2][y^2]$
  - ...

- Depth 0 solution, send all $[y], [y^2], \ldots, [y^N]$
  - $O(N)$ communication…

- Depth $\log \log N$ solution, send $[y], [y^2], [y^4], \ldots, [y^{2^i}], \ldots, [y^{2^\log N}]$
  - Compute all other powers in depth $\log \log N$
    - E.g. $[y^7] = [y^4][y^2][y]$
    - E.g. $N = 2^{2^8}$ $\Rightarrow$ depth $= 5$
  - $O(\log N)$ communication.
**Membership from FHE**

- Want to test if \( y \in X \):
  
  \[
  \{\left[ y^i \right] \mid i = 1,2,4,8, \ldots, \log N \}
  \]
  
  \[
  z \equiv \prod_{x \in X} (\left[ y \right] - x) \equiv a_N y^N + \cdots + a_1 y + a_0
  \]
  
  \[z = 0 \iff y \in X\]

- Performance,
  
  - Computation \( = O(N) \)
  - Depth \( = O(\log \log N) \)
  - Communication \( = O(\log N) \)

- Set intersection: For \( y \in Y \), run set membership protocol
  
  - Require \( O(nN) \) computation!!
  - Where \( n = |Y| \),
  - e.g. \( n = 1000 \)
Cuckoo Hashing

• Receiver performs Cuckoo hashing

\[ h(y_1), h(y_2), \ldots, h(y_n) \]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

\[ h(y_1) \]
\[ h(y_2) \]
\[ \vdots \]
\[ h(y_n) \]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

\[ h(y_1), h(y_2), \ldots, h(y_n) \]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

\[
\begin{align*}
    h(y_1) \\
    h(y_2) \\
    : \\
    h(y_n)
\end{align*}
\]
Cuckoo Hashing

• Receiver performs Cuckoo hashing

\[ x_1, x_2, \ldots, x_n \]
\[ y_1, y_2, \ldots, y_n \]

\[ h(y_1), h(y_2), \ldots, h(y_n) \]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

\[ \begin{align*}
  x_n & \quad x_4 \\
  x_3 & \quad x_1 \\
  y_4 & \\
  y_1 & \quad y_n \\
  y_3 & \\
  y_2 & \\
\end{align*} \]

\[ \quad \text{← Collision: } h(y_1) = h(y_n) \]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

- Use two hash functions $h, h'$

[PinkasScheiderZohner14, ChenLaineRindal17]
Cuckoo Hashing

• Receiver performs Cuckoo hashing

• Use two hash functions $h, h'$

[PinkasScheiderZohner14, ChenLaineRindal17]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

- Use two hash functions $h, h'$

[PinkasScheiderZohner14, ChenLaineRindal17]
Cuckoo Hashing

- Receiver performs Cuckoo hashing

- Use two hash functions $h, h'$

- For each bin, perform 1 membership test
  - When $N \gg n$, bin size $O(N/n)$
  - Overall complexity $O(N)$

[PinkasScheiderZohner14, ChenLaineRindal17]
Optimization: FHE Batching

- Fully homomorphic encryption naturally support “SIMD” type operations
  - A single FHE cipher-text/plaintext can be large…
  - Use Chinese Remainder Theorem (CRT) to pack several items into 1 cipher-text
    - E.g. 4096
Optimization: FHE Batching

- Fully homomorphic encryption naturally support “SIMD” type operations
  - A single FHE cipher-text/plaintext can be large…
  - Use Chinese Remainder Theorem (CRT) to pack several items into 1 cipher-text
    - E.g. 4096
Optimization: FHE Batching

- Fully homomorphic encryption naturally support “SIMD” type operations
  - A single FHE cipher-text/plaintext can be large…
  - Use Chinese Remainder Theorem (CRT) to pack several items into 1 cipher-text
    - E.g. 4096
Optimization: FHE Batching

- Fully homomorphic encryption naturally support “SIMD” type operations
  - A single FHE cipher-text/plaintext can be large…
  - Use Chinese Remainder Theorem (CRT) to pack several items into 1 cipher-text
    - E.g. 4096

\[
\vec{z} := \vec{r} \prod_{\vec{x} \in X} (\vec{y} - \vec{x})
\]

\[
\{ \left[ \vec{y}^i \right] \mid i = 1, 2, 4, \ldots \}
\]

\[
\vec{z}[i] = 0 \iff y[i] \in X
\]

\[
\vec{z} = \vec{y} \cdot \vec{x}
\]

4096 items per cipher-text
**Optimization: Splitting**

- Observe that the communication is unbalanced.
- Partition $\vec{X}$ into $s$ splits $\vec{X}_1, \ldots, \vec{X}_s$
  - Reduces depth to $\log \log N/n_s$
  - Large impact in practice, e.g. depth = 3

For $i = 1, \ldots, s$:

$$\vec{z}[i] = \bigwedge_{x \in \vec{X}_i} (\vec{y} - x)$$

\[ \{[\vec{y}^i] \mid i = 1, 2, 4, \ldots \} \]

\[ [\vec{z}_1] \ldots, [\vec{z}_s] \]

$z_i[i] = 0 \iff \vec{y}[i] \in X$
Final Protocol

\[
\begin{align*}
\text{For } i = 1, \ldots, s: \quad &\hat{z}_i^x := \hat{r} \prod_{x_i \in \tilde{X}_i} ([y_i] - x_i) \\
\{[\hat{y}_i^y] &| i = 1, 2, 4, \ldots \} &\quad [\hat{z}_1^z], \ldots, [\hat{z}_s^z] \\
\tilde{z}_j[i] = 0 &\iff \hat{y}[i] \in X
\end{align*}
\]

**Sender:**
- \( O(N) \) Computation w/ quasi-constant depth
- \( O(n \log N) \) communication
- Practical on server

**Receiver:**
- \( O(n \log N) \) Encryptions/Decryptions
- \( O(n \log N) \) communication
- Practical on cellphone

[ChenLaineRindal17]
Performance

- CLR17 (with unpublished updates to code)
  - Optimized for unequal set sizes

- Semi-honest hash table technique similar to [KKRT16]
- Uses Fully Homomorphic Encryption

- Very low communication when receiver’s set is much smaller than sender’s

- Communication = \(O(n \log N)\) bits
  - Previous approaches required \(O(N + n)\) bits

\[n = 5,000 \text{ vs } N = 16,777,216\]
The End