PIR-PSI: SCALING PRIVATE CONTACT DISCOVERY

PETS 2018
Motivation – Application: Contact Discovery

- Contact discovery tells social network users which of their contacts are in the social network
- An insecure naïve hashing-based protocol is used in practice
- Vulnerable to
  - Brute-force attacks (for small input domain, e.g. phone numbers)
  - Comparison with hashes from later sessions

Hashes of User Contacts → Matching WhatsApp Contacts
Motivation – Application: Private Contact Discovery

- Contact Discovery should be **efficient** and **scalable**, and **protect the privacy of user inputs**.
- It runs once when a user initially joins a social network
- ... and periodically to find contacts that join the social network later on.
Private Set Intersection (PSI)
Private Set Intersection (PSI)

Ideal World

“Receiver”

“Sender”

\[ X \cap Y \]
PSI for Contact Discovery

\[ |X| = n \ll |Y| = N \]

![Diagram showing PSI for Contact Discovery](image-url)
Status-Quo vs. PIR-PSI

- Communication linear in both sets $O(N + n)$
  - What about $N \gg n$?
  - Insecure solution
    - Send small set to other party
    - Communication = $O(\min(N, n))$

- PIR-PSI
  - Communication = $O \left(n \log \left(\frac{N \log n}{n}\right)\right)$
  - Client Computation = $O \left(n \log \left(\frac{N \log n}{n}\right)\right)$ AES operations
  - Server Computation = $O(N \log n)$ AES operations
Plaintext Database Query

\[ i \]

\[ y_i \]

\[ DB \]

\[ y_1 \]

\[ y_2 \]

\[ y_3 \]

\[ y_4 \]

\[ \cdots \]

\[ y_N \]
Private Information Retrieval (PIR)

Ideal World

DB

DB[i]

$\gamma_1$

$\gamma_2$

$\gamma_3$

$\gamma_4$

$\gamma_N$

$\gamma_1$

$\gamma_2$

$\gamma_3$

$\gamma_4$

$\gamma_N$
2-Server PIR [CGKS95]

Donald J. Trump @realDonaldTrump · Jul 7
Public opinion has turned strongly against the Rigged Witch Hunt and the "Special" Counsel because the public understands that there was no Collusion with Russia,
except by Crooked Hillary and the Dems!
34K t 17K 81K

Donald J. Trump @realDonaldTrump · Mar 25
...any law firm will take months to get up to speed (if for no other reason
than they can bill more), which is unfair to our great country - and I am very
happy with my existing team. Besides, there was NO COLLUSION with Russia,
except by Crooked Hillary and the Dems!

Donald J. Trump @realDonaldTrump · Mar 28
Why does the Mueller team have 13 hardened Democrat soma who Crooked
Hillary supporters, and Zero Republi
t anyone think this is fair? And yet th
85K t 32K 123K

Donald J. Trump @realDonaldTrump · Mar 26
The Mueller probe should never have
and there was no crime.
the Dossier paid for by Crooked
FISA COURT for surveillance of my
63K t 29K 102K

Donald J. Trump @realDonaldTrump · Jul 7
...Also, there is NO COLLUSION!
7:28 AM - 30 Oct 2017
21,611 Retweets 89,260 Likes

Donald J. Trump @realDonaldTrump · Mar 11
...have shown conclusively that there was no Collusion with Russia, just excuse
for losing. The only Collusion was that done by the DNC, the Democrats and
Crooked Hillary. The writer of the story, Maggie Haberman, a Hillary flunky, knows
nothing about me and is not given access.

Donald J. Trump @realDonaldTrump · Jun 5
Hoax continues, all because Jeff Sessions didn’t tell me
myself. I would have quickly picked someone else. So
casted, so many lives ruined...and Sessions knew better
No Collusion!
65K t 15K 82K

42K t 26K 96K

 workforce: NO COLLUSION, so why go crazy!

Donald J. Trump @realDonaldTrump · Apr 11
Collusion is caused by the Fake & Corrupt Russia
Democrat loyalists, or people that worked for
all (except Rosenstein who signed FISA &

 workforce: NO COLLUSION, so why go crazy!


#1

\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ y_4 \]
\[ \ldots \]
\[ y_N \]

#2

\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ y_4 \]
\[ \ldots \]
\[ y_N \]
2-Server PIR [CGKS95]

\[ r_1 \oplus r_2 = DB[i] \]

\[ q_1 \]
\[ q_2 \]

\[ r_1 \]
\[ r_2 \]

no collusion!

DB
\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ y_4 \]
\[ \ldots \]
\[ y_N \]
Example: 2-Server Linear Summation PIR [CGKS95]

\[ i = 2 \Rightarrow q = 001\ 000 \]
\[ q_1 \text{ chosen at random} \]
\[ q_2 = q \oplus q_1 \]

\[ r_1 = DB[1] \oplus DB[5] \]
\[ r_i = q_i \cdot DB \]

Query length \(|q| = N\)

\[ q_1 = 010\ 001 \]
\[ q_2 = 011\ 001 \]

\[ r_1 \oplus r_2 = DB[2] \]

\[ y_1 \]
\[ y_2 \]
\[ y_3 \]
\[ y_4 \]
\[ \ldots \]
\[ y_N \]
PIR from Distributed Point Functions (DPFs)

- **Point Functions:** $PF = \{ f_{i,v} : f_{i,v}(i) = v, f_{i,v}(x) = 0 \forall x \neq i \}$.

- **Distributed** PFs allow 2 parties the secret-shared PF evaluation, without revealing $i, v$.

- DPFs are described by short keys $k_1, k_2$ of length $O(\log N)$, where $N$ is the domain of $i$.

- By using $v = 1$, i.e., a DPF returning 1 only at index $i$, we can express the plain text query $q$ and thus build 2-server PIR with $O(\log N)$ communication complexity.

- Instantiated efficiently with AES.

Intuition:
DPF Key Expansion

$K_1$ $K_2$ $k_1$ $k_2$
Designated-Output PIR

\[ DB[i] \oplus m \]

1. \( q_1, m \)
2. \( q_2 \)
3. \( r_1 \oplus m \)
4. \( r_2 \oplus r_1 \oplus m \)
5. \( DB[i] \oplus m \)
PIR Private Equality Test

\[ x, i, m \]

\[ q_1, m \]

\[ q_2 \]

\[ r_1 \oplus m \]

\[ r_2 \oplus r_1 \oplus m = DB[i] \oplus m \]

\[ DB \]

\[ y_1 \]

\[ y_2 \]

\[ y_3 \]

\[ y_4 \]

\[ \ldots \]

\[ y_N \]
Cuckoo Hashing

- Server performs Cuckoo hashing.

\[ \begin{align*}
    h(y_1) & \rightarrow y_1 \\
    h(y_2) & \rightarrow y_2 \\
    \vdots & \quad \vdots \\
    h(y_N) & \rightarrow y_N
\end{align*} \]
Cuckoo Hashing

- Server performs Cuckoo hashing.
Server performs Cuckoo hashing.

Cuckoo Hashing

\[
\begin{align*}
h(y_1) & : y_1 \\
h(y_2) & : y_2 \\
\vdots & : \ldots \\
h(y_N) & : y_N
\end{align*}
\]
Server performs Cuckoo hashing.
Cuckoo Hashing

- Server performs Cuckoo hashing.

\[
\begin{align*}
&\text{Collision: } h(y_1) = h(y_N) \\
&y_1, y_N, y_3, y_2
\end{align*}
\]
Cuckoo Hashing

- Server performs Cuckoo hashing.

- To avoid collisions: use multiple hash functions - in this example: $h, h'$.

- In our implementation we used 3 hash functions and a cuckoo expansion factor of $e \approx 1.4$ for a cuckoo failure probability of $2^{-20}$ during one-time initialization.
Cuckoo Hashing

• Every element can be located in two possible bins.

• The client computes all hash functions for every element.
Cuckoo Hashing

- Every element can be located in two possible bins.

- To check if the server holds $x_1$, the client runs a PIR-PEQ with the 2nd and 4th bin.

- In the full protocol: instead of single PIR-PEQ, we run all of them together in a PSI protocol.
1. **Cuckoo Hashing**
   - Both servers compute the same cuckoo hash table for their $N$ elements.

2. **DPF-PIR Query**
   - The client delegated extraction of $n$ elements from the cuckoo table.

3. **Oblivious Shuffle**
   - One server receives the other server’s masked output and obliviously shuffles the PIR results to hide which Cuckoo hash function was used.

4. **Small PSI**
   - A standard PSI protocol is used to determine intersection.
Optimizations

• **Binning**
  • Instead of running full domain DPFs, we partition the server cuckoo table into bins and a smaller DPFs per bin.
  • Parallelization!

• **Batching**
  • Instead of running DPF queries separately, run all queries in each bin in parallel.
  • Only a single pass over the cuckoo table for multiple queries.

• **Larger PIR Blocks**
  • PIR queries can return multiple cuckoo table entries.
  • less communication, more computation in PSI.
PIR-PSI with 3 PIR Servers

\[ DB = DB_2 \oplus DB_3 \]

\[ DB \]

\[ (K_1 \cdot DB) \oplus m \]

\[ DB_2 \]

\[ K_2 \cdot DB_2 \]

\[ DB_3 \]

\[ K_3 \cdot DB_3 \]

\[ (K_2 \cdot DB_2) \oplus (K_2 \cdot DB_3) \oplus (K_1 \cdot DB) \oplus m \]

\[ = \]

\[ K_2 \cdot (DB_2 \oplus DB_3) \oplus (K_1 \cdot DB) \oplus m \]

\[ = \]

\[ K_2 \cdot DB \oplus K_1 \cdot DB \oplus m \]

\[ = \]

\[ DB[i] \oplus m \]
PIR-PSI Performance

• Communication and running time for $n = 1024$ client elements and server set sizes $N \in \{2^{20}, 2^{24}, 2^{26}, 2^{28}\}$.

• Benchmarked in Gigabit LAN, on 1 machine with 36 x 2.3 GHz. Implementation set to use 4 threads.

• Client computation is $\approx 10\%$ of total.

• Parameters for communication / computation trade-off
Conclusion

• Combination of DPF-based PIR with state-of-the-art PSI to achieve scalable contact discovery.

• Efficient open-source C++ implementation on Github: [github.com/osu-crypto/libPSI](https://github.com/osu-crypto/libPSI)

• Many more details in the paper!
  • Security Analysis
  • Cuckoo Hashing Parameters
  • Detailed performance analysis and comparison with related work
  • Extensions
Thank you!
References

- Some icons are made by Freepik from flaticon.com
• Extra / Backup slides coming up next...
A Sampling of PSI Over the Decades

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

\[ Q(x) := (x - y) \]
\[ Q(x) = 0 \]
\[ \Rightarrow x = y \]
A Sampling of PSI Over the Decades

- **[Meadows86]**: Private equality test
- **[HubermanFranklinHogg99]**: Private equality test to PSI
- **[DeCristofaroKimTsudik10]**: Malicious secure
- **[NaorPinkas99]**: Semi-honest PSI
- **[FreedmanNissimPinkas04]**: Hash table base PSI
- **[DachmanMalkinRaykovaYung09]**: Malicious secure
- **[GhoshJasper17]**: Malicious secure
- **[HuangEvansKatz12]**: Garbled Circuit based PSI

Timeline:
- 1985
- 1990
- 1995
- 2000
- 2005
- 2010
- 2015
- 2020
A Sampling of PSI Over the Decades

  - Private equality test

- 1990: [HubermanFranklinHogg99] Oblivious Polynomial Evaluation
  - Private equality test to PSI

- 1995: [NaorPinkas99] Oblivious Transfer & Bloom filter
  - Semi-honest PSI

- 2000: [FreedmanNissimPinkas04] Garbled Circuit based PSI
  - Hash table based PSI

- 2005: [DachmanMalkinRaykovaYung09] Oblivious Transfer + Bloom filter
  - Malicious secure

- 2010: [DeCristofaroKimTsudik10] Oblivious Transfer + Bloom filter
  - Malicious secure

  - Malicious secure

- 2020: [RindalRosulek17a] Oblivious Transfer + Bloom filter
  - Malicious secure

- 2020: [DongChenWen13] Oblivious Transfer + Bloom filter
  - Malicious secure
A Sampling of PSI Over the Decades

- **Diffie-Hellman**
  - Private equality test

- **Oblivious Polynomial Evaluation**
  - Semi-honest PSI

- **Generic MPC**

- **Oblivious Transfer & Bloom filter**

- **Oblivious Transfer Encoding**

- **FaginNaorWinkler96**
  - Private equality test

- **NaorPinkas99**
  - Semi-honest PSI

- **FreedmanNissimPinkas04**
  - Hash table based PSI

- **DachmanMalkinRaykovaYung09**
  - Malicious secure

- **DeCristofaroKimTsudik10**
  - Malicious secure

- **[HuangEvansKatz12]**
  - Garbled Circuit based PSI

- **[KKRT16]**
  - Malicious Oblivious Transfer + Bloom filter base PSI

- **[PinkasSchneiderZohner14, ...]**
  - Cuckoo hashing PSI

- **[KKRT16]**
  - Element-wise OT encoding

- **[RindalRosulek17b]**
  - Hash Table based PSI from OT

- **[RindalRosulek17a]**
  - Malicious Oblivious Transfer + Bloom filter base PSI

- **[DeCristofaroKimTsudik10]**
  - Malicious secure
A Sampling of PSI Over the Decades


Diffie-Hellman
Private equality test
Private equality test to PSI

Oblivious Polynomial Evaluation
Semi-honest PSI
Hash table base PSI
Malicious secure

Generic MPC

1985
Private equality test

Oblivious Transfer & Bloom filter
Oblivious Transfer + Bloom filter base PSI
Malicious secure

Fully Homomorphic Encryption
Element-wise OT encoding
Hash Table based PSI from OT
Hash Table based PSI from HFE

[NaorPinkas99] Semi-honest PSI
[FreedmanNissimPinkas04] Hash table base PSI
[DachmanMalkinRaykovaYung09] Malicious secure
[DeCristofaroKimTsudik10] Malicious secure

[NaorPinkas99]
[FreedmanNissimPinkas04]
[DachmanMalkinRaykovaYung09]
[DeCristofaroKimTsudik10]
[DeCristofaroKimTsudik10]

[This Work]
PIR+PSI

[ChenLaineRindal17]

[This Work]
PIR+PSI

[ChenLaineRindal17]

[PIR+PSI]

[PIR+PSI]

[PIR+PSI]

[PIR+PSI]
A Sampling of PSI Over the Decades