Mobile Private Contact Discovery at Scale

Daniel Kales (TU Graz), Christian Rechberger (TU Graz), Thomas Schneider (TU Darmstadt), Matthias Senker (TU Darmstadt), Christian Weinert (TU Darmstadt)

https://contact-discovery.github.io/
Agenda

1. Mobile contact discovery in practice

2. Improved *unbalanced* Private Set Intersection (PSI) protocols

3. Native MPC protocol implementations on *mobile* platforms

https://contact-discovery.github.io/
Mobile Contact Discovery – Privacy Concerns?

https://contact-discovery.github.io/
Mobile Contact Discovery – Privacy Concerns!

https://contact-discovery.github.io/
Our Contributions

Survey of secure mobile messaging applications

Optimized OPRF-based unbalanced PSI protocols

Native implementations in C/C++ utilizing ARMv8-A instruction sets

No proper privacy protection during contact discovery

Cuckoo filter compression, new OT precomputation, LowMC for OPRF

1000x faster GC evaluation, Signal integration, ~5s online phase
Contact Discovery in “Secure” Mobile Messaging Applications

SURVEY
## Contact Discovery in “Secure” Mobile Messaging Applications

<table>
<thead>
<tr>
<th>Messenger</th>
<th>Naïve Hashing</th>
<th>Analysis Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confide*</td>
<td>✔️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>Dust*</td>
<td>✘️</td>
<td>Network Traffic</td>
</tr>
<tr>
<td>Eleet*</td>
<td>✘️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>G DATA Secure Chat</td>
<td>✔️</td>
<td>Network Traffic</td>
</tr>
<tr>
<td>Signal (legacy / non-SGX)</td>
<td>✔️</td>
<td>Source Code</td>
</tr>
<tr>
<td>SIMSme</td>
<td>✔️</td>
<td>Network Traffic</td>
</tr>
<tr>
<td>Telegram</td>
<td>✘️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>Threema</td>
<td>✔️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>Viber</td>
<td>✘️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>WhatsApp</td>
<td>✘️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>Wickr Me</td>
<td>✔️</td>
<td>Privacy Policy</td>
</tr>
<tr>
<td>Wire</td>
<td>✔️</td>
<td>Privacy Policy</td>
</tr>
</tbody>
</table>

* contact discovery is optional
Unbalanced PSI Protocols – Related Work – Precomputation Form

PSI FOR MOBILE CONTACT DISCOVERY
Private Set Intersection (PSI) Protocols

Communication complexity:
\[ O(|\text{Client}| + |\text{Server}|) \text{ in online phase!} \]
Unbalanced PSI Protocols

Communication complexity:
\( O(|\text{Client}|) \) in \textbf{online} phase
\( O(|\text{Server}|) \) in \textbf{setup} phase
Related Work

Unbalanced PSI Protocols

- FHE [CLR17, CHLR18]

- Precomputation Form [KLS+17]

- PIR-PSI [DRRT18]

Oblivious PRF (OPRF)

- RSA Blind Signature
- Naor-Reingold PRF (NR-PSI)

Public-Key Crypto Diffie-Hellman Style

- AES Garbled Circuit (AES-GC-PSI)
OPRF-Based Unbalanced PSI Protocols in Precomputation Form

1. **Base Phase** $O(|\text{Client}|)$
   - OT Precomputation
   - Generate secret key $k$
   - (Build Garbled Circuits $GC_i$)

2. **Setup Phase** $O(|\text{Server}|)$
   - Encrypt all contacts with key $k$ and insert them into Cuckoo filter $CF$

3. **Online Phase** $O(|\text{Client}|)$
   - Run OPRF for all contacts $c_i$
   - Check if $e_i$ is in $CF$

Contacts $c_i \in \{1, \ldots, n\}$
Cuckoo Filter Compression

- Realistic parameters: tag size 32 / 42 bit and bucket size 3 for FPP $2^{-30} / 2^{-40}$
- More efficient updates: 4.3x less communication

Cuckoo Filter Bitmap Tag List

<table>
<thead>
<tr>
<th>Cuckoo Filter</th>
<th>Bitmap</th>
<th>Tag List</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 a</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>1 b</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>3 c d</td>
<td>1 1 0</td>
<td></td>
</tr>
<tr>
<td>4 e f</td>
<td>0 1 1</td>
<td></td>
</tr>
<tr>
<td>5 g</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>6 h i</td>
<td>1 1 0</td>
<td></td>
</tr>
<tr>
<td>7 j</td>
<td>1 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Compression ratio $\approx$ Load factor
More Efficient PRF for GC-PSI

- Free XOR [KS08] optimization for Yao’s GC protocol allows “free” evaluation of XOR gates
  - Use LowMC [ARS+15] instead of AES
    - Highly parametrizable block cipher for MPC and FHE applications

<table>
<thead>
<tr>
<th>PRF</th>
<th>Block Size</th>
<th>Key Size</th>
<th>#S-Boxes</th>
<th>Data Complexity</th>
<th>#Rounds</th>
<th>#ANDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowMC</td>
<td>128</td>
<td>128</td>
<td>42</td>
<td>$2^{64}$</td>
<td>13</td>
<td>1638</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>128</td>
<td>31</td>
<td>$2^{64}$</td>
<td>13</td>
<td>1209</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>128</td>
<td>1</td>
<td>$2^{64}$</td>
<td>208</td>
<td>624</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>128</td>
<td>1</td>
<td>$2^{32}$</td>
<td>192</td>
<td>576</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>128</td>
<td>1</td>
<td>$2^{128}$</td>
<td>287</td>
<td>861</td>
</tr>
<tr>
<td>AES-128</td>
<td>128</td>
<td>128</td>
<td>16</td>
<td>$2^{128}$</td>
<td>10</td>
<td>5120</td>
</tr>
</tbody>
</table>

8.2x Comm. Improvement
ARMv8-A Instruction Sets

IMPLEMENTATION

Open Source!
https://contact-discovery.github.io/
ARMv8-A Instruction Sets for Native Implementations

- Malicious Secure OT Extension Protocols: libOTe [Rin]
  - Heavily optimized for x86 architecture
  - Ported to ARMv8-A while maintaining compatibility with x86 counterpart

- Yao’s GC protocol [Yao86] with fixed-key AES garbling [BHKR13]
  - ARMv8 Cryptography Extensions (CE) provide hardware instructions for AES, SHA-1, and SHA-2
    - 35x faster AES evaluations compared to standard software implementation
  - ARMv8 NEON instruction set for vector operations on 128-bit registers
    - Efficiently work with 128-bit wire labels

  ➢ 1000x faster GC evaluation than Java implementation of [KLS+17] based on ObliVM [LWN+15]
Setup – Benchmarks – Protocol Extensions

EVALUATION
Setup – WiFi

Google Pixel 2 XL
Snapdragon 835 CPU @ 2.45GHz
4GiB of RAM

IEEE 802.11ac WiFi
230Mbit/s down-/upload
70ms RTT

Commodity Laptop
Intel Core i7-4600U CPU @ 2.6GHz
16GiB of RAM
Setup – LTE

Google Pixel 2 XL
Snapdragon 835 CPU @ 2.45GHz
4GiB of RAM

42Mbit/s download
4Mbit/s upload
80ms RTT

Commodity Laptop
Intel Core i7-4600U CPU @ 2.6GHz
16GiB of RAM
Benchmarks – Base + Online Phase (Checking 1k Client Contacts)

Run-Time (in seconds)

- AES-GC [KLS+17]
- LowMC-GC (This work)
- NR [KLS+17]
- ECC-NR (This work)

Communication (in MiB)

- AES-GC [KLS+17]
- LowMC-GC (This work)
- NR [KLS+17]
- ECC-NR (This work)
Benchmarks – Setup Phase (for $2^{28} = 268$M Server Contacts)

Run-Time (in seconds)

<table>
<thead>
<tr>
<th></th>
<th>[KLS+17]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTE</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>WiFi</td>
<td></td>
<td>267</td>
</tr>
</tbody>
</table>

Communication (in MiB)

<table>
<thead>
<tr>
<th></th>
<th>[KLS+17]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTE</td>
<td></td>
<td>1500</td>
</tr>
<tr>
<td>WiFi</td>
<td></td>
<td>1100</td>
</tr>
</tbody>
</table>

1.3x improvement in both Run-Time and Communication for This work compared to [KLS+17]
Protocol Extensions – Combination with Multi-Server PIR

1. Cuckoo Filter Transfer (one-time cost)

2. OPRF Evaluations
\[ e = \text{PRF}(\text{Contact } x) \]

3. Multi-Server PIR Lookup for \( e \)

Total Client - Server Communication \( O(\log |\text{Server}|) \) [BGI16]
CONCLUSION
Conclusion

- Most practical protocols and implementations for mobile **private** contact discovery at scale
- **General purpose** unbalanced PSI protocols
  - Mobile malware detection service, discovery of leaked passwords, etc.
- **Native Yao’s GC implementation** on ARMv8-A

<table>
<thead>
<tr>
<th>Requirement</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td># Registered Users</td>
<td>&gt; 1B</td>
</tr>
<tr>
<td># Entries per Address Book</td>
<td>10k</td>
</tr>
<tr>
<td>Latency</td>
<td>&lt; 2s</td>
</tr>
<tr>
<td>Communication</td>
<td>&lt; 10MiB</td>
</tr>
<tr>
<td></td>
<td>~ 250M</td>
</tr>
<tr>
<td></td>
<td>1k</td>
</tr>
<tr>
<td></td>
<td>&gt; 30s (~ 5s online)</td>
</tr>
<tr>
<td></td>
<td>&gt; 1GiB (~ 6MiB online)</td>
</tr>
</tbody>
</table>

https://contact-discovery.github.io/
Thank You!

https://contact-discovery.github.io/
Bibliography (1/3)


Bibliography (2/3)


Bibliography (3/3)


