The Circle Game: Scalable Private Membership Test Using Trusted Hardware

Sandeep Tamrakar 1
Jan-Erik Ekberg 2
Jian Liu 1
Benny Pinkas 3
Andrew Paverd 1
N. Asokan 1

1. Aalto University, Finland
2. Darkmatter (work done at Trustonic)
3. Bar-Ilan University, Israel
Malware Checking

On-device checking
- High communication and computation costs
- Database changes frequently
- Database is revealed to everyone
Malware Checking

On-device checking
• High *communication* and *computation* costs
• Database *changes* frequently
• Database is *revealed* to everyone

Cloud-based checking
• Minimal *communication* and *computation* costs
• Database *can change* frequently
• Database is *not revealed* to everyone
• User *privacy at risk!"
**The problem:** How to preserve end user privacy when querying cloud-hosted databases?

Server must not learn contents of client query (q).

**Current solutions** (e.g. private set intersection, private information retrieval):
- Single server: expensive in both computation and/or communication
- Multiple independent servers: unrealistic in commercial setting
Trusted Execution Environments (TEEs) are ubiquitous

- ARM TrustZone, Intel SGX, …

Can TEEs provide a practical solution for Private Membership Test?

Background: Kinibi on ARM TrustZone

Kinibi
• Trusted OS from Trustonic

Remote attestation
• Establish a trusted channel

Private memory
• Confidentiality
• Integrity
• Obliviousness

https://www.trustonic.com/solutions/trustonic-secured-platforms-tsp/
Background: Intel SGX

CPU enforced TEE (enclave)

Remote attestation

Secure memory
- Confidentiality
- Integrity

Obliviousness only within 4 KB page granularity

https://software.intel.com/sgx
System Model

Lookup Server

**REE**

Untrusted application

**TEE**

Trusted application

Dictionary provider

Dictionary: $X = \{x_1, x_2, \ldots, x_n\}$

User

Mobile device A

APK conversion
System Model

Untrusted application

Dictionary: $X$

Dictionary provider

User

Lookup Server

REE

Untrusted application

$x_1$

$x_2$

$\ldots$

$x_n$

Dictionary: $X$

TEE

Trusted application
System Model

Lookup Server

User

Dictionary provider

Dictionary: $X$

Untrusted application

Trusted application

Query: $q$

Response: $r$

$h(\text{APK})$

Secure channel with remote attestation

$\text{TEE}$

$\text{REE}$

$q \leftarrow (q == x_i)$
System Model

Lookup Server

REE
- Untrusted application
- Dictionary: \( X \)
- Information leak: Memory access patterns

TEE
- Trusted application
- Query buffer
- Response buffer
- \( r \leftarrow (q == x_i) \)

Query: \( q \)
Response: \( r \)

Secure channel with remote attestation

Dictionary provider
User

Mobile device A

Information leak: Memory access patterns
Requirements

Query Privacy: Adversary cannot learn/infer query or response content
  • User can always choose to reveal query content

Accuracy: No false negatives
  • However, some false positives are tolerable (i.e. non-zero false positive rate)

Response Latency: Respond quickly to each query

Server Scalability: Maximize overall throughput (queries per second)
Path ORAM

Stefanov et al. ACM CCS 2013, https://dl.acm.org/citation.cfm?id=2516660
Path ORAM

\[ f_{\text{locate\_block}}(q) = b_4 \]

Stefanov et al. ACM CCS 2013, https://dl.acm.org/citation.cfm?id=2516660
Path ORAM

Stefanov et al. ACM CCS 2013, https://dl.acm.org/citation.cfm?id=2516660
Stefanov et al. ACM CCS 2013, https://dl.acm.org/citation.cfm?id=2516660
Path ORAM

O(log(n)) computation and constant communication overhead per query

Not amenable for simultaneous queries O(mlog(n))

Stefanov et al. ACM CCS 2013, https://dl.acm.org/citation.cfm?id=2516660
Android App Landscape

On average a user installs 95 apps (Yahoo Aviate)
Yahoo Aviate study
Source: https://yahooaviate.tumblr.com/image/9579583893

Unique Android malware samples
Source: G Data https://secure.gd/dl-en-mmwr201504

Current dictionary size < $2^{22}$ entries

Even comparatively “high” FPR (e.g., ~$2^{-10}$) may have negligible impact on privacy
Cloud Scale PMT

Verify Apps: cloud-based service to check for harmful Android apps prior to installation

“… over 1 billion devices protected by Google’s security services, and over 400 million device security scans were conducted per day”

Android Security 2015 Year in Review

(c.f. ~2 million malware samples)
Requirements Revisited

Query Privacy: Adversary cannot learn/infer query or response content
  • User can always choose to reveal query content

Accuracy: No false negatives
  • However, some false positives are tolerable (i.e. non-zero false positive rate)

Response Latency: Respond quickly to each query

Server Scalability: Maximize overall throughput (queries per second)

Dictionary size* = $2^{26}$ entries (~ 67 million entries)

* parameters suggested by a major anti-malware vendor
Carousel Approach

Dictionary provider

User

Lookup Server

REE

Untrusted application

Dictionary: \( X \)

\[
\begin{align*}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{align*}
\]

TEE

Trusted application

\[
r = (q \in X)
\]
Carousel Caveats

1. Adversary can measure dictionary processing time
   • Spend equal time processing each dictionary entry

2. Adversary can measure query-response time
   • Only respond after one full carousel cycle

Both impact response latency (recall Requirements)
Therefore, aim to minimize carousel cycle time
How to Minimize Cycle Time?

Represent dictionary using efficient data structure

Various existing data structures support membership test:
  • Bloom Filter
  • Cuckoo hash

Experimental evaluation required for carousel approach
Carousel Approach

Lookup Server

REE

Untrusted application

Dictionary representation: $Y$

$y_1$

$y_2$

$\ldots$

$y_m$

Encode

TEE

Trusted application

$r = (q \in Y)$

Query representation

Query: $q$

$h(APK)$

Response: $r$

Response buffer

Query buffer

Dictionary: $X$

$x_1$

$x_2$

$\ldots$

$x_n$

Dictionary provider

User

Secure channel with *remote attestation*
Sequence of differences

**Dictionary:**

\[
\begin{array}{cccccc}
0 & x_1 & x_2 & x_3 & \cdots & x_n \\
x_1 - 0 & x_2 - x_1 & x_3 - x_1 & \cdots & x_n - x_{n-1}
\end{array}
\]

**Sorted hashes**

**Dictionary Representation:**

\[
\begin{array}{cccccc}
d_1 & d_2 & d_3 & \cdots & d_n
\end{array}
\]

**Differences**

\[
x_1 = 0 + d_1 \quad x_2 = x_1 + d_2
\]

**Queries:**

\[q\]

**Very efficient sequential access**
Bloom Filter

Dictionary:

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>...</th>
<th>xₙ</th>
</tr>
</thead>
</table>

k hash functions

Insert

Dictionary Representation:

0 1 1 0 1 0 1 0 1 0

Check

Queries:

q
Cuckoo Hash

Dictionary:

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>...</th>
<th>x_n</th>
</tr>
</thead>
</table>

k hash functions

Dictionary Representation:

<table>
<thead>
<tr>
<th>x_5</th>
<th>x_1</th>
<th>x_j</th>
<th>x_j</th>
<th>x_2</th>
<th>x_20</th>
</tr>
</thead>
</table>

Queries:

Insert

Check

h_1  h_2  h_3  h_4

q
Experimental Evaluation

Kinibi on ARM TrustZone
• Samsung Exynos 5250 (Arndale)
• 1.7 GHz dual-core ARM Cortex-A17
• Android 4.2.1
• ARM GCC compiler and Kinibi libraries
• Maximum TA private memory: 1 MB
• Maximum shared memory: 1 MB

Intel SGX
• HP EliteDesk 800 G2 desktop
• 3.2 GHz Intel Core i5 6500 CPU
• 8 GB RAM
• Windows 7 (64 bit), 4 KB page size
• Microsoft C/C++ compiler
• Intel SGX SDK for Windows
Experimental Evaluation

**Kinibi on ARM TrustZone**
- Samsung Exynos 5250 (Arndale)
- 1.7 GHz dual-core ARM Cortex-A17
- Android 4.2.1
- ARM GCC compiler and Kinibi libraries
- Maximum TA private memory: 1 MB
- Maximum shared memory: 1 MB

**Intel SGX**
- HP EliteDesk 800 G2 desktop
- 3.2 GHz Intel Core i5 6500 CPU
- 8 GB RAM
- Windows 7 (64 bit), 4 KB page size
- Microsoft C/C++ compiler
- Intel SGX SDK for Windows

Note: Different CPU speeds and architectures
Performance: Batch Queries

Kinibi on ARM TrustZone

Intel SGX
Performance: Steady State Query Arrival

Kinibi on ARM TrustZone

Intel SGX
Performance: Steady State Query Arrival

Kinibi on ARM TrustZone

Intel SGX

Beyond breakdown point query response latency increases over time
## Evaluation Summary

Cuckoo hash provides best performance

### Average response latency

<table>
<thead>
<tr>
<th></th>
<th>Kinibi on ARM TrustZone</th>
<th>Intel SGX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cuckoo on (Path) ORAM</td>
<td>0.009 s</td>
<td>0.001 s</td>
</tr>
<tr>
<td>Cuckoo on a Carousel</td>
<td>1.240 s</td>
<td>0.360 s</td>
</tr>
</tbody>
</table>

### Sustainable query throughput

<table>
<thead>
<tr>
<th></th>
<th>Kinibi on ARM TrustZone</th>
<th>Intel SGX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cuckoo on (Path) ORAM</td>
<td>111 q/s</td>
<td>1354 q/s</td>
</tr>
<tr>
<td>Cuckoo on a Carousel</td>
<td>1025 q/s</td>
<td>3720 q/s</td>
</tr>
</tbody>
</table>
Private Contact Discovery in Signal

Technology preview: Private contact discovery for Signal

maxie0 on 26 Sep 2017

At Signal, we've been thinking about the difficulty of private contact discovery for a long time. We've been working on strategies to improve our current design, and today we've published a new private contact discovery service.

Using this service, Signal clients will be able to efficiently and scalably determine whether the contacts in their address book are Signal users without revealing the contacts in their address book to the Signal service.

Background

Signal is social software. We don't believe that privacy is about austerity, or that a culture of sharing and communication should mean that privacy is a thing of the past. We want to enable online social interactions that are rich and expressive in all the ways that people desire, while simultaneously ensuring that communication is only visible to its intended recipients.

https://signal.org/blog/private-contact-discovery/
Private Contact Discovery in Signal

“An SGX enclave on the server-side would enable a service to perform computations on encrypted client data without learning the content of the data or the result of the computation.”

“Private contact discovery using SGX is fairly simple at a high level:
1. Run a contact discovery service in a secure SGX enclave.
2. Clients that wish to perform contact discovery negotiate a secure connection over the network all the way through the remote OS to the enclave.
3. Clients perform remote attestation to ensure that the code which is running in the enclave is the same as the expected published open source code.
4. Clients transmit the encrypted identifiers from their address book to the enclave.
5. The enclave looks up a client’s contacts in the set of all registered users and encrypts the results back to the client.”

https://signal.org/blog/private-contact-discovery/
Private Contact Discovery in Signal

“Unfortunately, doing private computation in an SGX enclave is more difficult than it may initially seem.”

“However, the host OS can still see memory access patterns, even if the OS can’t see the contents of the memory being accessed.”

“This class of problems has been studied under the discipline of Oblivious RAM (ORAM).”

“There are some elegant generalized ORAM techniques, like Path ORAM, but unfortunately they don’t work well for this problem.”

https://signal.org/blog/private-contact-discovery/
Private Contact Discovery in Signal

“By keeping one big linear scan over the registered user data set, access to unencrypted RAM remains “oblivious,” since the OS will simply see the enclave touch every item once for each contact discovery request.”

“The full linear scan is fairly high latency, but by batching many pending client requests together, it can be high throughput.”

https://signal.org/blog/private-contact-discovery/
Conclusions

Cloud-assisted services raise new security/privacy concerns
• But naïve solutions may conflict with privacy, usability, deployability, …

Cloud-assisted malware scanning
• Carousel approach is promising
• Implementing ORAM on SGX is hard!

In future
• Efficient oblivious data structures for trusted hardware
• New use cases for Carousel (e.g. leaked passwords, …)

 Asiaccs 2017
Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, N. Asokan.
The Circle Game: Scalable Private Membership Test Using Trusted Hardware