rORAM: Efficient Range ORAM with Locality

Anrin Chakraborti, Radu Sion
Stony Brook University

Adam Aviv, Seung Geol Choi, Travis Mayberry, Daniel Roche
United States Naval Academy
It's all about the Clouds!

What do you want to be when you grow up—a cumulonimbus?

Data storage.

[Logos of Amazon, Azure, and Google Cloud]

© D. Reeder for CloudTweaks.com
Protecting Outsourced Data

- Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. Islam et al. NDSS, ‘12
- Connecting the Dots: Privacy Leakage via Write-Access Patterns to the Main Memory. John et al. HOST, ’17
- ...
Oblivious RAM (ORAM)

Observing the physical memory accesses, an adversary cannot learn
1. Which item has been accessed.
2. What operation has been performed.
Path ORAM [Stefanov et al. CCS ‘13]

<table>
<thead>
<tr>
<th>LogicalBlockID</th>
<th>LeafLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(V_2)</td>
</tr>
<tr>
<td>1</td>
<td>(V_0)</td>
</tr>
<tr>
<td>2</td>
<td>(V_3)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

STASH

CLIENT

SERVER

Position Map
Path ORAM Evictions

Can also evict along \textit{pre-determined} paths
Path ORAM: Performance Metrics

Bandwidth: $O(\log n)$, worst-case

Round-trips: 1 RT per access

Computational complexity: trivial

Locality of Access:
• # of seeks: $O(\log n)$
• Access seq. chunk: $O(chunkSize \times \log N)$
Why Locality of Access?

- HDD: 1 seek = 10,000x slower
- SSD: Random placement $\Rightarrow$ Significant wear
- File systems
  - caching, prefetching require data locality
- Applications with range queries e.g., GIS
Locality-Privacy Tradeoff

Data locality for “free”?  
No

What can we afford to leak?  
Sequential access size?  
[Asharov ‘17]

Why is this acceptable?
Range ORAM: Locality-Optimized Range Queries

Range ORAM [Asharov et al. ‘17]:

For range query of size $r$

✓ $O(\log^3 N)$ seeks
✓ $O(r \log^3 N)$ bandwidth required

rORAM:

✓ $O(\log N)x$ fewer seeks
✓ $O(\log N)x$ lower bandwidth required
rORAM

- \( l \in O(\log N) \) independent ORAMs
- Data is duplicated

Seek-optimized for querying ranges of size \( 2^l \)

- For \( R_l \):
  - # of seeks for reading (\( r=2^l \)) blocks in range: \( O(\log N) \) ind. of \( r \)
  - # of seeks for evicting (\( r=2^l \)) blocks: \( O(\log N) \) ind. of \( r \)
**Insight 1: Locality-Optimized Layout**

**Problem:** Evicting \( r \) blocks requires \( O(r \cdot \log N) \) seeks

**Observation:** Eviction Path Selection is Deterministic
- Paths for **consecutive** evictions known apriori
- Order in which nodes are accessed **per level** known apriori
- Perform evictions level-wise

![Eviction Path Selection Order Diagram](image)
Batching Evictions Example

Client

Stash

Level 1

Level 2

Level \log N

Server

LOGICAL VIEW

BatchEvict to \(V_0, V_{n/2}\)

Batch \(r\) evictions: \(O(\log N)\) seeks
Insight 2: Locality-Optimized Re-Mapping

**Problem:** Reading \( r \) blocks in range requires \( O(r \times \log N) \) seeks

**Idea:** Any \( r \) consecutive eviction paths can be read with \( O(\log N) \) seeks

### Map Blocks in Range to Consecutive Eviction Paths

**Eviction Path Selection Order**

\[ V_{n-1} \rightarrow V_0 \rightarrow V_{n/2} \rightarrow \ldots \rightarrow V_{j} \rightarrow V_{j+1} \rightarrow \ldots \rightarrow V_{K} \rightarrow V_1 \]

**Remap:** \([a, a+1, \ldots, a+r-1]\)

**Position Map**

<table>
<thead>
<tr>
<th>LogicalBlockID</th>
<th>LeafLabel</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>( v_j )</td>
</tr>
<tr>
<td>a+1</td>
<td>...</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>a+r-1</td>
<td>( v_k )</td>
</tr>
</tbody>
</table>

**ReadRange** \([a, b], b=a+r-1\): \( O(\log N) \) seeks
Access Protocol

Access\([2, 5]\)

# of Seeks:
- \(O(\log N)\) disk seeks for ReadRange from \(R_2\)
- \(O(\log N)\) disk seeks for BatchEvict to \(R_i\) - \(O(\log^2 N)\) seeks in total
Insight 3: Distributed Position Map

Insight: Reuse paths in ORAMs $R_0$, $R_1$, .. $R_l$

How do we know where block 2 is in $R_0$, $R_1$, ...?
- $O(\log N)$ position map accesses

Pointer-based Oblivious Data Structure
- With each block, store pointers to its location in other ORAMs
- Locate position for “free” with reads
# Asymptotic Performance

**Access**\([j, j + r - 1]:\)

- ✓ \(O(\log N)\times\) fewer seeks
- ✓ \(O(\log N)\times\) lower bandwidth required

<table>
<thead>
<tr>
<th></th>
<th>Seeks</th>
<th>Bandwidth</th>
<th>Server Space</th>
<th>Leakage</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathORAM</td>
<td>(O(r.\log^2 N))</td>
<td>(O(r.\log^2 N))</td>
<td>(O(N))</td>
<td>none</td>
</tr>
<tr>
<td><strong>rORAM</strong></td>
<td>(O(\log^2 N))</td>
<td>(O(\log^2 N))</td>
<td>(O(N\log N))</td>
<td>Range size</td>
</tr>
<tr>
<td>Asharov et al.</td>
<td>(O(\log^3 N))</td>
<td>(O(\log^3 N))</td>
<td>(O(N\log N))</td>
<td>Range size</td>
</tr>
<tr>
<td>Demertzis et al.</td>
<td>(O(r))</td>
<td>(O(r.N^{1/3}.\log^2 N))</td>
<td>(O(N))</td>
<td>none</td>
</tr>
</tbody>
</table>
Query Access Time

Local HDD
(logscale, higher is better)

Network Block Device
(logscale, higher is better)

30 – 50x speedup, range size >= 32 blocks

10x speedup, range size >= 64 blocks
Throughput

Local HDD
(higher is better)

Throughput (Queries/sec)

Benchmark

Path ORAM
Batched Evictions
rORAM

File Server = 5x, Video Server = 11x

Network Block Device
(higher is better)

Throughput (Queries/sec)

Benchmark

Path ORAM
Batched Evictions
rORAM

File Server = 2x, Video Server = 4x
Summary

Practical Range ORAM

✓ $O(\log N)x$ fewer seeks
✓ $O(\log N)x$ lower bandwidth required

Optimized for Real World Applications

Can we do better?

app-specific optimizations
What I am working on

I am on the job market!

Oblivious RAM [NDSS ’19, ‘19]

Plausible Deniability [PETS ‘17, ‘19]

Integrity-Preserving Block Storage [ApSys ‘17]

History Independence [TIFS ’15]

Secure CPU Architecture & Secure Virtualization

Query Authentication [TKDE]
Thank you!!