Burst ORAM: Minimizing ORAM Response Times for Bursty Access Patterns

Jonathan Dautrich, Emil Stefanov, Elaine Shi
Usenix Security, San Diego
August 22, 2014
Problems to Solve

- Access pattern privacy for outsourced data
- Practical response times for bursty workloads

Oblivious RAM

3 Hours ➔ 56 ms
ObliviStore ORAM ➔ Burst ORAM

99.9% response times, NetApp trace simulation, 50ms latency
32TB capacity, 100GB client storage, 400Mbps bandwidth
Outline

- Private Data Outsourcing
  - Oblivious RAM
- Practical ORAM Response Times
  - Burst ORAM
- Burst ORAM Details
- Results
Outline

- Private Data Outsourcing
  - Oblivious RAM
- Practical ORAM Response Times
  - Burst ORAM
- Burst ORAM Details
- Results
Data Outsourcing (Cloud)

- Advantages
  - Accessibility and availability
  - IT overhead savings
Why care about data privacy?

- Inexpensive but untrusted provider
- Privacy Regulations
  - FERPA, HIPAA, ITAR, etc.
Access Pattern Privacy

- Encryption alone is insufficient
- *Access patterns* leak information
  - Patterns in plaintext (Dautrich & Ravishankar, EDBT 2013)
  - Search query contents (Islam et al., NDSS 2012)
Oblivious RAM (Goldreich & Ostrovsky, 1996)

- Provable access pattern privacy
- ORAM translates client requests to public IO
  - Online IO: Needed to satisfy request
Oblivious RAM (Goldreich & Ostrovsky, 1996)

- Provable access pattern privacy
- ORAM translates client requests to public IO
  - Online IO: Needed to satisfy request
  - Offline IO: After request completes (shuffling)
Outline

- Private Data Outsourcing
  - Oblivious RAM
- Practical ORAM Response Times
  - Burst ORAM
- Burst ORAM Details
- Results
ORAM Costs

Client Space

Bandwidth Cost

Server Space

Trusted (Client-Side)

Client

ORAM Protocol

Server

# Transfers

# Requests

1

Q Z J

J B Y O
Q F G H
N W K L
M E Z P
## Some Prior ORAM Schemes

*N blocks of size $B$ ($B$ generally 1KB to 1MB)*

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Client Space</th>
<th>Server Space</th>
<th>Bandwidth Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goldreich &amp; Ostrovsky 1996</td>
<td>$O(B \log N)$</td>
<td>$O(BN \log N)$</td>
<td>$O(\log^3 N)$</td>
</tr>
<tr>
<td>Kushilevitz et al. 2012</td>
<td>$O(B)$</td>
<td>$O(BN)$</td>
<td>$O(\log^2 N / \log \log N)$</td>
</tr>
<tr>
<td>Goodrich et al. 2012</td>
<td>$O(BN^{1/c})$</td>
<td>$O(BN)$</td>
<td>$O(\log N)$</td>
</tr>
<tr>
<td>Stefanov et al. 2013 (ObliviStore)</td>
<td>$~N \log_2 N$ + $BN^{1/2}$</td>
<td>$~2BN-4BN$</td>
<td>$~\log_2 N$</td>
</tr>
</tbody>
</table>
ORAM Costs

Client Space

Trusted (Client-Side)

Response Times

Client

1

# Transfers

# Requests

Bandwidth Cost

ORAM Protocol

Server Space

Server

J B Y O
Q F G H
N W K L
M E Z P

# Requests

# Transfers
Bursty Workload: Existing ORAMs

Legend
- Online IO
- Offline IO

Requests

Time

Start Burst 1 Requests Satisfied Start Burst 2 Requests Satisfied

Long Response Times
Burst ORAM

Goals

- Minimize burst response times
- Keep total bandwidth cost low

Based on ObliviStore

- Bandwidth-efficient
- Large client space
Burst ORAM Strategies

1: Reduce Online IO
   - XOR technique

2: Delay Shuffling (Offline IO)
   - Maximally utilize client space

3: Prioritize Efficient Shuffling
   - Less work to free same client space
   - Efficiency ≈ Space Freed / Required IO
   - Scheduling policy must be “oblivious”
Existing ORAMs

Legend
- Online IO
- Offline IO

Burst ORAM

Less online IO, more offline IO

Offline IO (shuffling) delayed until burst satisfied

Long Response Times

| Start Burst 1 | Requests Satisfied | Start Burst 2 | Requests Satisfied |

Short Response Times

| Start Burst 1 | Requests Satisfied | Start Burst 2 | Requests Satisfied |
Outline

- Private Data Outsourcing
  - Oblivious RAM
- Practical ORAM Response Times
  - Burst ORAM
- Burst ORAM Details
- Results
Layout: Burst ORAM & ObliviStore

- $\sqrt{N}$ partitions of $\sqrt{N}$ blocks each
- Each request routed to single partition
ObliviStore ORAM (Stefanov et al. 2013)

Online Cost
(Blocks Transferred During Read)

Offline Cost
(Blocks Transferred During Shuffling)

Req. 1
Req. 2
Req. 3
Idle Time
ObliviStore ORAM (Stefanov et al. 2013)

Real

Dummy

Online Cost
(Blocks Transferred During Read)

Offline Cost
(Blocks Transferred During Shuffling)

Req. 1

Req. 2

Req. 3

Idle Time
ObliviStore ORAM (Stefanov et al. 2013)

Online Cost
(Blocks Transferred During Read)

Req. 1

Offline Cost
(Blocks Transferred During Shuffling)

Req. 2

Req. 3

Idle Time
ObliviStore ORAM (Stefanov et al. 2013)

Online Cost
(Blocks Transferred During Read)

Req. 1
R D D D D

Req. 2
D R D D D

Req. 3

Idle Time

Offline Cost
(Blocks Transferred During Shuffling)

R D D D R D D R R

D R D R D R D R R

R D D R D D R R D
ObliviStore ORAM (Stefanov et al. 2013)

Online Cost (Blocks Transferred During Read)
- Req. 1: R D D D D
- Req. 2: D R D D D
- Req. 3: R D D D
- Idle Time

Offline Cost (Blocks Transferred During Shuffling)
- Req. 1: D R D R D R D R R
- Req. 2: R D D R D D R R D
- Req. 3: R D D R D D R R D
ObliviStore ORAM (Stefanov et al. 2013)

Online Cost
(Block transferred during Read)

- Req. 1: RDDDD
- Req. 2: DRDDD
- Req. 3: RDDDD
- Idle Time: DRDDD

Offline Cost
(Block transferred during Shuffling)

- Req. 1: DRDDDRD
- Req. 2: RDDRRDD
- Req. 3: DRRDDDR
- Idle Time: DRDDDRR
**Burst ORAM**

- **Reduce**
  - Req. 1
    - Online Cost: R D D D D
    - Offline Cost: D R D R D R D R R
  - Req. 2
    - Online Cost: D R D D
    - Offline Cost: R D D D R D D R R
  - Req. 3
    - Online Cost: R D D D
    - Offline Cost: D R D D R D R R
  - Idle Time
    - Online Cost: D R D D R D R R
    - Offline Cost: D R D D R D R D R

**Delay or Minimize**

- Burst ORAM

Burst ORAM: XOR Technique

At most one real block

Online Cost
(Blocks Transferred During Read)

Offline Cost
(Blocks Transferred During Shuffling)

Req. 1

Req. 2

Req. 3

Idle Time

Dummy blocks reconstructed locally and subtracted out
Burst ORAM

Online Cost
(Blocks Transferred During Read)

Offline Cost
(Blocks Transferred During Shuffling)

Req. 1

Req. 2

Req. 3

Idle Time

Small Amount of Shuffling to Free Space for Next Block on Client
Burst ORAM

Online Cost
(Block Transferred During Read)

- Req. 1
  - X
- Req. 2
  - X
- Req. 3
  - X
- Idle Time

Offline Cost
(Block Transferred During Shuffling)

- DR
Burst ORAM

Online Cost
(Block Transferred During Read)

Offline Cost
(Block Transferred During Shuffling)

Req. 1

Req. 2

Req. 3

Idle Time

Well suited to mobile device
Outline

- Private Data Outsourcing
  - Oblivious RAM
- Practical ORAM Response Times
  - Burst ORAM
- Burst ORAM Details
- Results
Burst ORAM – Extended Burst

32 TB ORAM, 100 GB client storage

- Burst ORAM
- ObliviStore
- Burst ORAM No Prioritization

Bandwidth Cost of Request

Request Index

client space full, shuffling begins
convergence - can no longer delay shuffling
prioritizing efficient jobs delays shuffling
99.9% Response Time Comparison on NetApp Trace
(50ms network latency, 32 TB ORAM, 100 GB client storage)
99.9% Response Time Comparison on NetApp Trace
(50ms network latency, 32 TB ORAM, 100 GB client storage)

- Burst ORAM
- ObliviStore
- Burst ORAM No Job Prioritization
- Without ORAM (Optimal)
NetApp Trace Bandwidth Costs
(50ms network latency, 32 TB ORAM, 100 GB client storage, 400Mbps bandwidth)
Conclusion

- Accomplishments
  - Practical response times during bursts
  - Maintains low bandwidth cost
- Limitations
  - Does not reduce *total* bandwidth cost
- Future
  - Lower bandwidth cost and low response times