EnigMap: External-Memory Oblivious Map for Secure Enclaves

Afonso Tinoco\textsuperscript{1,2}, Sixiang Gao\textsuperscript{1}, and Elaine Shi\textsuperscript{1}

\textsuperscript{1}Carnegie Mellon University
\textsuperscript{2}Instituto Superior Tecnico
Private Contact Discovery

Signal users
Private Contact Discovery

“Here are your friends”

Signal users
My address book is top secret!

“Here are your friends”
Strawman Solution: Encryption

“Here are your friends”
Access patterns to even encrypted data leak sensitive information.
Access patterns to even encrypted data leak sensitive information.

Simply permuting the data doesn't solve the problem

⇒

Need Oblivious Algorithms
Signal 2017: batched linear scan

\[ O\left(\frac{n}{\beta}\right) \] overhead

500 servers

n: total # memory blocks
\beta: batch size

Signal 2022: Path ORAM

[SDS+13]

\[ O\left(\log^2 n\right) \] overhead

6 servers
Trusted hardware needs oblivious algorithms!

- Secret Network, Oasis Network, Flashbots
Oblivious Map: Key-Value Store

• Query Privacy  
  client doesn’t leak which keys it is querying

• Database Privacy  
  database is kept private
SGX & Oblivious Algorithms

- Oblivious map inside of SGX Enclave
- Instruction and memory trace should not leak information about private data ☐ use x86’s CMOVcc
SGX & External-Memory

- Limited EPC memory
- Larger External-Memory via EWB or OCALL
SGX & External-Memory - Microbenchmarks

• OCALL 46x to 66x more expensive than MOV

☐ External-Memory page swaps is an important metric for SGX’s algorithms
ORAM/algorithms

literature: **word RAM**

reads/writes **bytes**
ORAM/algorithms literature: **word RAM**

- **Compute** overhead
- **Memory** overhead

reads/writes bytes
ORAM/algorithms literature: **word RAM**

Secure enclaves: **external-memory**

- Reads/writes bytes
- Reads/writes 4KB pages
Enclave needs to fetch encrypted data stored in unprotected memory/disk

Secure enclaves: external-memory

reads/writes 4KB pages
Secure enclaves:

Page swap dominates

- System call
- Enc/Dec
- Possible disk swap

reads/writes

4KB pages
<table>
<thead>
<tr>
<th>Oblix</th>
<th>Compute</th>
<th>Page swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10%</td>
<td></td>
<td>&gt;90%</td>
</tr>
</tbody>
</table>
Oblix

Compute

<10%  

Page swap

>90%

Enigmap

[TGS'23]

40%  60%

Uses external-memory algo

10-100X faster than Oblix
Our Contributions

• Design and implement EnigMap - an oblivious map
• Instruction and Memory Trace Oblivious
• Both External-Memory and Instruction asymptotically more efficient than previous implementations (Oblix [MPCCP18])
• Concretely 13-53x faster than previous work
• Improved Initialization Algorithm
Our Solution

• Oblivious AVL Tree [WNLCSSH14]
• PathORAM storage [SDSCFRYD12]
• Optimized for the External-Memory model [EnigMap]
AVL Tree

• Binary Search Tree
• Each node corresponds to a key
• For an AVL tree with N nodes:
  • $1.44 \log(N)$ maximum depth
  • Search/Insert/Delete start from root
Oblivious AVL Tree

• Recall: “Instruction and *memory trace* should not leak information about private data”

• Oblivious Data Structures [WNLCSSH14] Store nodes in ORAM
PathORAM

• Full binary tree with N leafs
• Each bucket has Z=4 blocks
• Blocks can have data (an AVL node) or be fillers
• Each AVL node has a random position
• Access(key, position):
  □ Returns node with a given key knowing it is on path position
  □ Cost: Each access call will read and write that path
  □ Node with key gets assigned a new random position after access
Oblivious AVL Tree

- Oblivious Data Structures [WNLCSSH14]: Store nodes in ORAM → keep position of child nodes as part of parent nodes metadata
External memory

Not all buckets can be cached in EPC
### Our key optimizations

- Locality friendly layout
- Initialization algorithm

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Cost per batch of operations</th>
<th>Cost of initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>page swaps</td>
<td>compute</td>
</tr>
<tr>
<td>Signal [4]</td>
<td>$O(N/B)$</td>
<td>$O(\beta^2 + N)$</td>
</tr>
<tr>
<td>Oblix [2]</td>
<td>$O(\beta \log^2 N)$</td>
<td>$O(\beta \log^3 N)$</td>
</tr>
<tr>
<td>ENIGMAP</td>
<td>$O(\beta \log_B N \cdot \log N)$</td>
<td>$\tilde{O}(\beta \log^2 N)$</td>
</tr>
</tbody>
</table>
Locality Friendly layout

• PathORAM reads paths on the ORAM tree
• Disk pages store several buckets
• Our layout:
  • Store subtrees in the same page
Locality Friendly layout

Our layout

Heap layout
Locality Friendly layout – $\log_B N$ pages

Our layout

Heap layout
Locality Friendly layout - $\log_B N$ pages

- Experimental optimization
- Optimize pagesize ($B$)
- Pages with 4 levels (15 Buckets)
Integrity and freshness for free

- Encrypt each disk page with AES-GCM
- Keep nonce stored on parent page
- Implicit merkle tree
- Smaller EPC – no Version Array
Initialization Algorithm

- How to initialize a database if we have N key-value pairs in plaintext
- Naïve Initialization:
  - Do N insertions
  - $O(N \log_B N \log N)$ page swaps
  - $\tilde{O}(N \log^2 N)$ computation
- We can do better! → Read our paper for full details
Initialization Algorithm

• We receive a list of N keys – Key[N]
• Need to build both AVL tree and ORAM with correct positions

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>91</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>27</td>
</tr>
<tr>
<td>73</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
Initialization Algorithm – Asymptotic Results

- $O\left(\frac{N}{B} \log_{M} \frac{N}{B}\right)$ page swaps
- $O(N \log N)$ computation

→ Read paper for details
Other concrete optimizations

• **Cache blocks during insertions** □ ½ ORAM operations for insert
• **ORAM treetop caching** □ less external memory reads per ORAM operation
• **Single pass AVL insertion** □ ½ instructions for insertion
• **Optimize page size for OCALL** □ improve concrete external memory performance
• **Store values in separate ORAM** □ increasing the size of values doesn’t affect AVL performance
Experimental Results

• Private Contact Discovery – search
• Initialization
Private Contact Discovery

• Query $\beta$ contacts
• Vary database size ($N$)
• Speedup Signal $\beta=1000$:
  • $N=2^{28} \rightarrow 15x$
  • $N=2^{32} \rightarrow 132x$
• Comparison Oblix (SOTA)
  • $N=2^{24} \rightarrow 13x$
  • $N=2^{28} \rightarrow 53x$
• Comparison new signal:
  • 2-4x speedup
Initialization

- Vary database size (N)
- Compare with:
  - Oblix – SOTA
  - Naïve Initialization – N insertions
- Results:
  - 2-8x speedup
Oblivious Data Structure Library

• Open source oblivious algorithm library
• Memory and instruction trace oblivious
• External-Memory efficient

https://github.com/odslib/odsl
Conclusions

• Takeaways:
  • Efficient trace oblivious algorithms are possible and crucial for enclave security
  • External Memory is an important model for enclave algorithms
  • By focusing on external memory we achieved 13-53x query speedup compared to previous SOTA

• Resources:
  • Oblivious Data Structure Library  □ https://github.com/odslib/odsl
  • Extended Version  □ https://eprint.iacr.org/2022/1083
  • Artifacts  □ https://github.com/odslib/EnigMap/tree/usenix-artifacts-final
Questions?

Oblivious Data Structure Library  
[https://github.com/odslib/odsl](https://github.com/odslib/odsl)