Building Enclave-Native Storage Engines for Practical Encrypted Databases

Yuanyuan Sun, Sheng Wang, Huorong Li, Feifei Li
Alibaba Group
Introduction
Security in the Cloud

How do we ensure security of data during the operation of cloud databases?
Confidential Computing

**Trusted execution environments (TEEs)**
- Hardware extensions for trusted computing
  - E.g., Intel SGX and AMD SEV
- Guarantees confidentiality and integrity
  - Computation and data in it

**Hardware enclaves of Intel SGX**
- A trusted component in an untrusted system
  - Uses protected memory to isolate shielded execution from compromised OS
  - Proves that it is an authentic enclave running the desired code with attestation

---

![Diagram](attachment:image.png)

Client → Secure channel → Enclave → Untrusted server

- Client
- Secure channel
- Enclave
- Protected Memory
- Untrusted server
Challenges for SGX-based Databases

**Limited memory space in enclave (EPC)**
- Only ~94MB available for applications
- To support larger memory region, SGX supports secure page swapping
  - Significant overhead (about 40K cycles) for an EPC miss
  - Only about 200 cycles for an EPC hit
- Many memory-consuming operations in databases

**Huge cost from enclave interaction**
- Syscalls cannot be executed in enclave, causing expensive cost to exit the enclave via OCall functions (about 8K cycles)
  - TLB flushing, security checks, etc.
- Host process can only invoke enclave via ECall functions

How to significantly reduce EPC page swapping and enclave interaction?
Exploration to a Broader Design Space
Strawman: B+-tree with Encrypted Keys

Structure overview
- Unchanged logical semantics
- Most index processing logic remains unaffected
  - E.g., node split and merge
- Decrypt keys and return \( cmp \) results in plaintext
  - E.g., \( \text{ISearch}(E(42)) \)

Limitations
- Frequent enclave interaction
  - E.g., 5 ECalls are required for \( \text{ISearch}(E(42)) \)
- High overheads on storage
  - Huge storage amplification for smaller encryption granularity
- Severe information leakage
  - Key orders, parent-child relationships, etc.
### Table 1: Possible design choices for encrypted storage characterized in five dimensions. The choices made for *Enclave Index* are bolded and the choices for *Enclave Store* are tagged with an asterisk mark (*).

<table>
<thead>
<tr>
<th>Design Dimension</th>
<th>Design Choice</th>
<th>Influence Security (Info)</th>
<th>Performance</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encryption Granularity</strong></td>
<td>item-level encryption</td>
<td>leak structure</td>
<td>high storage overhead; fast for a single read</td>
<td>can fetch data w/o enclave</td>
</tr>
<tr>
<td></td>
<td>page-level encryption *</td>
<td>leak data value</td>
<td>very low storage overhead; fast for batched small reads</td>
<td>all data access must be in enclave</td>
</tr>
<tr>
<td><strong>Execution Logic in Enclave</strong></td>
<td>index: key comparison</td>
<td>leak key</td>
<td>performance from massive ECalls</td>
<td>can split or merge node w/o enclave</td>
</tr>
<tr>
<td></td>
<td>index: index node access</td>
<td>leak node</td>
<td>performance from a few ECalls</td>
<td>all index access must be in enclave</td>
</tr>
<tr>
<td></td>
<td>table: none</td>
<td>leak key</td>
<td>performance from no ECall</td>
<td>can fetch or scan record(s) w/o enclave</td>
</tr>
<tr>
<td></td>
<td>table: data page access *</td>
<td>leak node</td>
<td>performance from a few ECalls</td>
<td>all record access must be in enclave</td>
</tr>
<tr>
<td><strong>Memory Access Granularity</strong></td>
<td>item-level access *</td>
<td>leak key</td>
<td>performance from on-demand read</td>
<td>require small footprint in enclave</td>
</tr>
<tr>
<td></td>
<td>page-level access</td>
<td></td>
<td>performance from page copy</td>
<td>require large footprint in enclave</td>
</tr>
<tr>
<td><strong>Enclave Memory Usage</strong></td>
<td>minimum usage *</td>
<td></td>
<td>no EPC capacity requirement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fixed usage</td>
<td></td>
<td>low EPC capacity requirement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>proportional usage</td>
<td></td>
<td>high EPC capacity requirement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>unlimited usage</td>
<td></td>
<td>high EPC capacity requirement</td>
<td></td>
</tr>
<tr>
<td><strong>Record Identity Protection</strong></td>
<td>rid encryption *</td>
<td></td>
<td>no influence</td>
<td>only useful in some settings</td>
</tr>
<tr>
<td></td>
<td>ciphertext re-encryption</td>
<td></td>
<td>only useful in some settings</td>
<td></td>
</tr>
</tbody>
</table>
Design Space

Encryption granularity

Data page

Item Item Item ... Item

Execution logic in enclave

Host

Index node access

Data page access

Enclave

Key comparison

Memory access granularity

EPC Item ... Item

Address space

Enclave memory usage

• min usage
• Fixed usage
• Proportional usage
• Unlimited usage

Record identity protection

• No action
• Rid encryption
• Ciphertext re-encryption
Overall Architecture of Enclage

**Enclage - An enclave-native encrypted storage engine**
- Enclage Index: a B⁺-tree-like index
- Enclage Store: a heap-file-like table store

### Design choices made in Enclage

<table>
<thead>
<tr>
<th></th>
<th>Enclage Index</th>
<th>Enclage Store</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encryption Granularity</strong></td>
<td>Page-level encryption</td>
<td>Page-level encryption</td>
</tr>
<tr>
<td><strong>Execution logic in enclave</strong></td>
<td>Index node access</td>
<td>Data page access</td>
</tr>
<tr>
<td><strong>Memory access granularity</strong></td>
<td>Page-level access</td>
<td>Item-level access</td>
</tr>
<tr>
<td><strong>Enclave memory usage</strong></td>
<td>Fix usage</td>
<td>Minimum usage</td>
</tr>
<tr>
<td><strong>Record identity protection</strong></td>
<td>Rid encryption/ciphertext re-encryption</td>
<td>Rid encryption/ciphertext re-encryption</td>
</tr>
</tbody>
</table>
Overview of Enclave Index

Hierarchical architecture

- **Tier1: EBuffer**
  - trusted buffer in enclave
  - An unencrypted index node per page

- **Tier2: Mbuffer**
  - untrusted buffer in memory
  - Several encrypted nodes per page

- **Tier3: External storage**

Optimizations

- Reduction of EPC page swapping
- Mitigation of enc/dec costs
- Avoidance of unnecessary OCalls
Enclage Store & Delta Decryption

Enclage Store

- A heap-file-like table store
- Adopts append-only strategy
  - The active page: holds recently arrived records
  - Lower data locality
- Retrieves a record
  - Loads a page
  - Decrypts and extracts target record

Delta decryption protocol

- Built on top of AES-CTR mode
  - Allows a small block within a large cipher be solely decrypted
- Executing a TGet operation
  - Locates the page in MBuffer and loads it to enclave
  - Calculates the counter for the record and construct the IV
  - Only decrypts the target record
03 Evaluation
# Experimental Setup

## Hardware platform

<table>
<thead>
<tr>
<th>Server Node</th>
<th>Intel SGX (~94MB EPC), Intel Core E7-1270 (4 cores), 64GB DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Red Hat 6.4.0 with Linux 4.9.135 kernel, SGX driver and SGX SDK 2.6</td>
</tr>
</tbody>
</table>

## Compared Systems

### Enclage Index

<table>
<thead>
<tr>
<th>Encryption Granularity</th>
<th>Location of execution logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Item</td>
</tr>
<tr>
<td>ItemEnc</td>
<td>Item</td>
</tr>
<tr>
<td>ShieldStore (hash-based)</td>
<td>Item</td>
</tr>
<tr>
<td>Enclage Index</td>
<td>Page</td>
</tr>
</tbody>
</table>

### Enclage Store

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item-level</td>
<td>Heap file containing encrypted records</td>
</tr>
<tr>
<td>Page-level</td>
<td>Heap file containing encrypted pages</td>
</tr>
<tr>
<td>Delta-enc</td>
<td>On top of Page-level, adopts the delta decryption protocol (AES-CTR)</td>
</tr>
</tbody>
</table>
Overall Performance

- Enclave Index achieves about 100Kops/s, and outperforms Baseline (20.04x) and ItemEnc (5.34x).
  - Only 1 ECall during each operation, and each accessed node is decrypted at most once
- Enclave Index also achieves better performance, compared to Baseline (13.19x), ItemEnc (9.69x) and ShieldStore (7-12x).
  - The more frequent the ECall is invoked, the greater the performance gain from the mode
  - Frequent decryption in ShieldStore
Different Decryption Protocols

- When a access miss occurs,
  - Page-level: load and decrypt the entire desired page
  - Delta-enc: extract and decrypt the desired record (1.40x)
  - Item-level: directly extract the encrypted record (1.57x)

More experiments: Please check our paper
04 Summary & Conclusion
Summary & Conclusion

✓ Data confidentiality is one of the biggest concerns that hinders enterprise customers from moving their workloads to the cloud.

✓ Though TEEs provide a powerful building block, practical designs of TEE-based encrypted databases have not been well explored.

✓ Our contributions:
  Ø Provides a comprehensive exploration of possible design choices for building an enclave-based encrypted database storage
  Ø Proposes Enclage, an enclave-native storage engine that makes practical trade-offs

✓ Enclage improves the throughput by 13x and the storage efficiency by 5x.
Thanks

Email: yuanyuan.sun@alibaba-inc.com