Breach-Resistant Structured Encryption

Ghous Amjad, Seny Kamara & Tarik Moataz
Databases

“A database is an organized collection of data” --- Wikipedia
Encrypted Database

\[ \text{Enc}_{K} \]
Q: can we encrypt DBs even in use?
Tradeoffs: Functionality vs. Efficiency

- SQL
- NoSQL

- FHE-based
- ORAM-based
- SK-FE-based
- PK-FE-based
- PPE-based
- STE-based
Tradeoffs: Efficiency vs. Security

- **STE-based**
- **PPE-based**
- **ORAM-based**
- **skFE-based**
- **pkFE-based**

The diagram illustrates the tradeoff between Efficiency and Leakage for different methods.

- **Efficiency**
  - **STE+ORAM-based**
  - **ORAM-based**
  - **STE+ORAM-based**
  - **skFE-based**
  - **pkFE-based**

- **Leakage**
Background: Data Structures

- DXs map labels to values
  - Get: $\text{DX}[w_3]$ returns $id_2$

- MMs map labels to tuples
  - Get: $\text{MM}[w_3]$ returns $(id_2, id_4)$
Structured Encryption [CK’10]

Setup($t^k$, DS) $\implies$ (K, EDS)

Token(K, q) $\implies$ tk

Query(EDS, tk) $\implies$ ans
Background: Encrypted Data Structures

[CK’10]

Encrypted Multi-Map = Encrypted Inverted Index =

Encrypted NoSQL DB  Encrypted Graph DB  Encrypted relational DB

Single-keyword SSE

[SWP’00], [Goh’03], [CGKO’06], [CK10], [KPR’12], [KP’13], [CJJKRS’13], [CJJJKRS’14], [Bost’16] ...
Adaptive Security for STE \([\text{CGKO’06,CK’10}]\)

Real

- Multi-map MM
- Encrypted Multi-map EMM
- \(w_i\)
- \(u_i\)

Ideal

- Multi-map MM
- Encrypted Multi-map EMM
- \(\mathcal{L}_S(\text{MM})\)
- \(\mathcal{L}_Q(\text{MM}, w_i)\)
- \(\mathcal{L}_U(\text{MM}, u_i)\)
Forward Privacy \[\text{[SPS'14]}\]

• Informally \[\text{[SPS'14]}\]

  “Updates cannot be correlated to previous queries”

• Formally \[\text{[Bost'16]}\]

  \[\mathcal{L}_Q(\text{MM}, (\text{op}, w, \overline{v})) = \#\overline{v}\]
Security of Encrypted Structures

[CGKO’06,CK’10]

• Definition guarantees security vs. adversary that
  • Holds encrypted structure & executes queries
  • Models an untrusted cloud provider
• Data breaches can occur even when server is trusted
  • Storage is compromised
  • Malicious employee
  • Government subpoena
• Adversary holds encrypted structure but does not see queries
Snapshot Security

- Adversary holds encrypted structure but does not see queries
- Discussed and formalized in [LW’16] for PPE
- Discussed in [PBP’16, GRS’17] but never formalized for STE
Q: What is snapshot security?
Snapshot Security

Real

\[ \mathcal{L}(\text{EMM}_0) \]

\[ \mathcal{L}(\text{EMM}_1, \text{op}) \]

\[ \mathcal{L}(\text{EMM}_2, \text{op}) \]

Ideal

\[ \mathcal{L}(\text{MM}_0) \]

\[ \mathcal{L}(\text{MM}_1, \text{op}) \]

\[ \mathcal{L}(\text{MM}_2, \text{op}) \]
Snapshot Security and Breach-Resistance

• Informally

“Breach-resistant leakage reveals at most the size of the current structure”

• Formally

\[ \mathcal{L}_{\text{Snp}}(\text{MM}, \text{op}_1, \ldots, \text{op}_i) = \mathcal{L}_S(\text{MM}_i) = \sum_{w \in \mathcal{W}} \#\text{MM}_i[w] \]
Tradeoffs: Efficiency vs. Security

Efficiency vs. Persistent Adversary!

- STE-based
- PPE-based
- skFE-based
- pkFE-based
- ORAM-based
- FHE-based
Tradeoffs: Efficiency vs. Security

Efficiency vs. Snapshot Adversary

- STE-based
- skFE-based
- pkFE-based
- PPE-based
- ORAM-based
- FHE-based
Snapshot Security

Static Structures

\[ \mathcal{L}_S(\text{MM}) = \sum_{w \in W} \#\text{MM}[w] \]

Breach-resistance

Dynamic Structures

Forward privacy

Breach-resistance

Insertion independence (variant of history independence)

Write-only obliviousness
Q: Can we design breach-resistant & forward-private EMMs?
Dual-Secure EMMs

• [SPS’14]
  • Query complexity

\[ O\left( \#\text{MM}[w] \cdot \text{polylog}\left( \sum_{w \in W} \#\text{MM}[w] \right) \right) \]

Q: Can we design efficient dual-secure EMMs?
$\pi_{\text{dyn}}$ [CJJJKRS’14]
\[ \pi_{\text{dyn}} \text{ [CJJJKRS'14]} \]

EMM.Setup \[1^k, \]

Multi-map MM

- \( w_1 \) connected to \( id_1, id_3, id_4 \)
- \( w_2 \) connected to \( id_3 \)
- \( w_3 \) connected to \( id_2, id_4 \)

\[ \xrightarrow{\cdot} \]

Encrypted MM

- \( F_{Kw1}(1) \) connected to \( id_1 \)
- \( F_{Kw1}(2) \) connected to \( id_3 \)
- \( F_{Kw1}(3) \) connected to \( id_4 \)
- \( F_{Kw2}(1) \) connected to \( id_3 \)
- \( F_{Kw3}(1) \) connected to \( id_2 \)
- \( F_{Kw3}(2) \) connected to \( id_4 \)

* PRF and Enc keys are different but derived from \( w_i \)
\[ \pi_{\text{dyn}} [\text{CJJJKRS'14}] \]
\( \pi_{d\text{yn}} \) [CJJJKRS’14]

- **EMM.Get**, \( K_{W1} \)

\[
\begin{align*}
1. & \text{DX.Get} & \text{DX}, F_{Kw1(1)} & \rightarrow \text{id}_1 \\
2. & \text{DX.Get} & \text{DX}, F_{Kw1(2)} & \rightarrow \text{id}_3 \\
3. & \text{DX.Get} & \text{DX}, F_{Kw1(3)} & \rightarrow \text{id}_4 \\
4. & \text{DX.Get} & \text{DX}, F_{Kw1(4)} & \rightarrow \perp
\end{align*}
\]
$\pi_{dyn}$ [CJJJKRS’14]

$F_{Kw1}(4)$  $\text{id}_9$

$\text{EMM.Edit}^+$

1. $\text{DX.Put}$
$\pi_{\text{dyn}}$ [CJJJKRS'14]
$$\pi_{\text{dyn}} \text{ [CJJJKRS'14]}$$
\[\pi_{\text{dyn}} \quad [\text{CJJJKRS'14}]\]

**EMM.Edit**

\[
\begin{align*}
\text{Dictionary DX} & \quad \text{Dictionary DX} \\
F_{Kw1}(1) & \quad \text{id}_1 \\
F_{Kw1}(2) & \quad \text{id}_3 \\
F_{Kw1}(3) & \quad \text{id}_4 \\
F_{Kw2}(1) & \quad \text{id}_3 \\
F_{Kw3}(1) & \quad \text{id}_2 \\
F_{Kw3}(2) & \quad \text{id}_4 \\
\end{align*}
\]

\[
\begin{align*}
\text{Dictionary DX} & \quad \text{Dictionary DX} \\
F_{Kw1}(1) & \quad \text{id}_1 \\
F_{Kw1}(2) & \quad \text{id}_3 \\
F_{Kw1}(3) & \quad \text{id}_4 \\
F_{Kw1}(4) & \quad \text{id}_3 \\
F_{Kw2}(1) & \quad \text{id}_3 \\
F_{Kw3}(1) & \quad \text{id}_2 \\
F_{Kw3}(2) & \quad \text{id}_4 \\
\end{align*}
\]
$$\pi_{\text{dyn}} \quad [\text{CJJJKRS'14}]$$

$$\omega_1 = K_{w_1}$$

$$O (\#\text{MM}[w] + \text{dels}_0(w))$$

Get

1. DX.Get $F_{Kw1(1)}(\quad)$
2. DX.Get $F_{Kw1(2)}(\quad)$
3. DX.Get $F_{Kw1(3)}(\quad)$
4. DX.Get $F_{Kw1(4)}(\quad)$

EMM.Get $[\text{EMM}, K_{w_1}]$
Forward-Private $\pi_{\text{dyn}}$

• Why is $\pi_{\text{dyn}}$ not forward-private?
  • new pairs encrypted under same key used for search,
    • $K_{wi} := F_K(w_i||1)$
    • so previously searched $w$'s can be linked to new pairs
• Making $\pi_{\text{dyn}}$ forward-private
  • use keys with version number that rotates at each search
    • $K_{wi} := F_K(w_i||\text{version}||1)$
  • To search send keys for all versions
    • $F_K(w_i||\text{version1}||1)$, ..., $F_K(w_i||\text{version8}||1)$
Efficiency

- Most dynamic EMM constructions handle deletes naively
  - forward-private or not
- Query complexity
  \[ O\left(\#\text{MM}[w] + \text{dels}_0(w)\right) \]
- Storage complexity
  \[ O\left(\sum_{w \in W} \#\text{MM}[w] + \text{dels}_0(w)\right) \]
Rebuilding

• Rebuild operation
  • Executed throughout lifetime of encrypted structure
  • Removes/prunes delete pairs

• Cost $\Omega\left(\sum_{w \in W} \#\text{MM}[w]\right)$

• Query complexity
  $O\left(\#\text{MM}[w] + \text{dels}_r(w)\right)$

• Storage complexity
  $O\left(\sum_{w \in W} \#\text{MM}[w] + \text{dels}_r(w)\right)$
Rebuilding Encrypted Structures

• Ideally a zero-leakage operation

• **Approach #1**
  • Client queries for each keyword and recovers encrypted id’s
  • Removes deleted id’s
  • Re-inserts new encrypted keywords and id’s

• Leakage

\[
\mathcal{L}_R(MM) = \left( \left( \mathcal{L}_G(MM, w) \right)_{w \in W}, \left( \mathcal{L}_U(MM, (w, \overline{id})) \right)_{w \in W} \right) \\
= \left( f(w), \#MM[w] + \text{dels}_r(w), g(w), \#MM[w] \right)_{w \in W}
\]

Leaks new information to persistent Adv: query leakage of *unsearched* keywords
Rebuilding Encrypted Structures

• Ideally a zero-leakage operation

• **Approach #2**
  • Keep track of searched and unsearched
  • Use Approach #1 for searched
  • For unsearched sample pair uniformly at random & re-insert

• Leakage $\mathcal{L}_R(MM) = \left( \left( f(w), \#MM[w] + \text{dels}_r(w), g(w), \#MM[w] \right), \sum_{w \in S} \#MM[w] \right)$
  
= \left( \left( f(w), \#MM[w], \text{dels}_r(w), g(w) \right), \sum_{w \in U} \#MM[w] \right)$
Rebuilding Encrypted Structures

- What about Snapshot security?
  \[ L_{\text{Snp}}(MM) = L_S(MM) = \sum_{w \in W} \#MM[w] \]

- Rebuild is not de-amortized
- Variant with de-amortized rebuild
  - When de-amortized rebuild occurs impacts snapshot leakage
  - Executed during Updates
  - Requires stash at client
## Forward-Private EMMs

<table>
<thead>
<tr>
<th>Year</th>
<th>Search</th>
<th>Client Storage</th>
<th>Forward Privacy</th>
<th>Snapshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPS’14</td>
<td>$O\left(#\text{MM}[w] \cdot \text{polylog}(#\text{MM})\right)$</td>
<td>$O\left(#\text{W}\right)$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B’16</td>
<td>$O\left(#\text{MM}[w] + \text{dels}_0(w)\right)$</td>
<td>$O\left(#\text{W}\right)$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>BMO’17</td>
<td>$O\left(#\text{MM}[w] + \text{dels}_0(w)\right)$</td>
<td>$O\left(#\text{W}\right)$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EKPE’17</td>
<td>$O\left(#\text{MM}[w] + \text{dels}_s(w)\right)$</td>
<td>$O\left(#\text{W}\right)$</td>
<td>Yes/No</td>
<td>No</td>
</tr>
<tr>
<td>This work</td>
<td>$O\left(#\text{MM}[w] + \text{dels}_r(w)\right)$</td>
<td>$O\left(#\text{W} + \text{ML}\right)$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Implementation

• Forward-private & response-hiding variant of $\pi_{\text{dyn}}$
  • de-amortized rebuild with $\lambda = 3$
  • Java (1114 LOC)
• Clusion encrypted search library
  • Lucene, Bouncy Castle
  • HMAC-SHA256 for PRFs and ROs
Experimental Setup

- Amazon EC2 c3.8xlarge instance
  - 32 vCPUs and 60GB of RAM
- Wikipedia
  - 26.5GB & 2,681,795 files
- Experiments (in memory)
  - time to setup EMM in function of pairs
  - Size of EMM & size of client state in function of pairs
  - Server query time in function of pairs for different selectivities
  - Server update time in function of pairs for different $\lambda$
  - Effect of rebuild on query time
Setup Time & Sizes

![Graphs showing setup time and size versus number of pairs (w, id).]
Query & Update Time
Thank you