How to Search on Encrypted Data

SENY KAMARA
MICROSOFT RESEARCH
Encryption

- $\text{Gen}(1^k) \rightarrow K$
- $\text{Enc}(K, m) \rightarrow C$
- $\text{Dec}(K, c) \rightarrow m$

Secure Communication

Alice  

Bob

Eve
Encryption

- \( \text{Gen}(1^k) \implies K \)
- \( \text{Enc}(K, m) \implies C \)
- \( \text{Dec}(K, c) \implies m \)

Secure Storage

Alice

Eve
Encryption

- $\text{Gen}(1^k) \mapsto K$
- $\text{Enc}(K, m) \mapsto C$
- $\text{Dec}(K, c) \mapsto m$

Secure Cloud Storage
Cloud Storage

- **Software-as-a-service**
  - Email (e.g., Exchange, Gmail, Hotmail)
  - Personal storage (e.g., Dropbox, Skydrive)
  - Voicemail (e.g., Google voice)

- **Platform-as-a-service**
  - Databases (e.g., Amazon RDS/Dynamo/SimpleDB, Azure SQL DB/Tables)

- **Infrastructure-as-a-service**
  - Blob storage (e.g., Amazon S3, Azure Blob)
Encryption

- Gen($1^k$) $\implies K$
- Enc(K, m) $\implies C$
- Dec(K, c) $\implies m$

Secure Cloud Storage

Alice $\rightarrow$ Eve
Encrypted Search
Encrypted Search
Two Simple Solutions

Q: can we do better?

Large comm. complexity

Large local storage
More Advanced Solutions

- Property-preserving encryption
  [Bellare-Boldyreva-O’Neill06]

- Functional encryption
  [Boneh-di Crescenzo-Ostrovsky-Persiano06]

- Oblivious RAM
  [Goldreich-Ostrovsky92]

- Fully-homomorphic encryption
  [Gentry09]

- Searchable encryption
  [Song-Wagner-Perrig01]
Encrypted Search

\[ \text{Enc}_K \left( \right) + \text{database} \rightarrow L_1 \]

\[ W \quad \text{Enc}_K \left( \right) \quad \text{Enc}_K \left( \right) \rightarrow L_2 \]
Encrypted Search

- Size of EDB
- Storage leakage
- Search time
- Query leakage
- Rounds of interaction
Property-Preserving Encryption

- Encryption that supports public tests
- Examples:
  - Deterministic encryption
    [Bellare-Boldyreva-O’Neill06]
  - Order-preserving encryption
    [Agrawal-Kiernan-Srikant-Xu04, Boldyreva-Chenette-Lee-O’Neill09]
  - Orthogonality-preserving encryption
    [Pandey-Rouselakis12]
Property-Preserving Encryption

- \( \text{Gen}(1^k) \mapsto K \)
- \( \text{PPE}(K, m) \mapsto c \)
- \( \text{Test}(c_1, c_2) \mapsto \{T, F\} \)
- \( \text{Dec}(sk, c) \mapsto m \)

\[ \begin{align*}
\text{Enc}_K &\rightarrow \text{PPE}_K[W_1] \quad \text{PPE}_K[W_2] \quad \text{PPE}_K[W_3] \\
\text{Enc}_K &\rightarrow \text{PPE}_K[W_2] \quad \text{PPE}_K[W_3] \\
\text{Enc}_K &\rightarrow \text{PPE}_K[W_1] \quad \text{PPE}_K[W_4]
\end{align*} \]
Deterministic Encryption

[Bellare-Boldyreva-O’Neill06]

- \( \text{Gen}(1^k) \mapsto K \)
- \( \text{PPE}(K, w) \mapsto F_K(w) \)
- \( \text{Test}(c_1, c_2) \mapsto c_1 = c_2 \)
- \( \text{Dec}(sk, c) \mapsto m \)

\[ \text{EDB} \]

\[ \begin{align*}
\text{Enc}_K & \quad \rightarrow \\ F_K(W_1) & \quad F_K(W_2) & \quad F_K(W_3) \\
\text{Enc}_K & \quad \rightarrow \\ F_K(W_2) & \quad F_K(W_8) \\
\text{Enc}_K & \quad \rightarrow \\ F_K(W_1) & \quad F_K(W_4) \\
\end{align*} \]
PPE/DET-Based Solution

Security

- $L_1$ leakage
  - #DB
  - equality
  - PK: DB*

- $L_2$ leakage
  - access pattern
  - search pattern

Efficiency

- Search
  - Sub-linear in #DB
  - process EDB like DB

- Legacy

* Unless DB has high entropy
Functional Encryption

- Encryption that supports private tests

Examples:

- Identity-based encryption
  [Boneh-Franklin01, Boneh-diCrescenzo-Ostrovsky-Persiano06]

- Attribute-based encryption
  [Sahai-Waters05,]

- Predicate encryption
  [Shen-Shi-Waters]
Functional Encryption

- $\text{Gen}(1^k) \mapsto K$
- $\text{FEnc}(K, x, m) \mapsto c$
- $\text{Token}(K, f) \mapsto t$
- $\text{Dec}(c, t) \mapsto m \text{ if } f(x)=1$

$\text{Token}_K(f_w)$

$\text{Enc}_K \rightarrow \text{FEnc}_K(w_1, 1) \text{, } \text{FEnc}_K(w_2, 1)$

$\text{Enc}_K \rightarrow \text{FEnc}_K(w_3, 1)$

$\text{Enc}_K \rightarrow \text{FEnc}_K(w_6, 1) \text{, } \text{FEnc}_K(w_2, 1)$
Identity-Based Encryption

- $Gen(1^k) \rightarrow K$
- $IBE(K, id, m) \rightarrow c$
- $Token(K, id') \rightarrow t$
- $Dec(t, c) \rightarrow m$ if $id = id'$

**Diagram:**
- $Enc_K \rightarrow IBE_K(w_1, 1) \rightarrow IBE_K(w_2, 1)$
- $Enc_K \rightarrow IBE_K(w_3, 1)$
- $Enc_K \rightarrow IBE_K(w_6, 1) \rightarrow IBE_K(w_2, 1)$

**Tokenization:**
- $Token_K(f_w)$

**Encryption:**
- $Enc_K(\cdot)$
FE/IBE-Based Solution

Security

- $L_1$ leakage
  - #DB
  - Equality
  - PK: DB*

- $L_2$ leakage
  - access pattern
  - PK: keyword

Efficiency

- Slow search
  - Linear in #DB
Oblivious RAM

- Encryption that supports private reads and writes
- Examples:
  - Square-root scheme [Goldreich-Ostrovsky92]
  - Hierarchical scheme [Goldreich-Ostrovsky]
ORAM-Based Solution

- OStruct($t^k, \text{Mem}) \mapsto K, \Omega$
- ORead((K, i), \Omega) $\mapsto (\text{Mem}[i], \perp)$
- OWrite((K, i, v), \Omega) $\mapsto (\perp, \Omega')$

EDB = OStruct
ORAM-Based Solution

Security

- $L_1$ leakage
  - #DB
  - Equality
  - PK: DB*

- $L_2$ leakage
  - access pattern
  - PK: keyword

Efficiency

- Very slow search
  - 1 R/W = $\text{polylog}(n) \cdot \text{R+W}$
Homomorphic Encryption

- Encryption that supports computation

Examples:

- Fully-homomorphic encryption
  [Gentry09, …]

- Somewhat homomorphic encryption
  [Boneh-Goh-Nissim05, …]
Homomorphic Encryption

- $\text{Gen}(1^k) \mapsto K$
- $\text{Enc}(K, m) \mapsto c$
- $\text{Eval}(f, c_1, \ldots, c_n) \mapsto c'$
- $\text{Dec}(sk, c') \mapsto f(\text{Dec}(c_1), \ldots, \text{Dec}(c_n))$

$\text{EDB} = \text{FHE}_K(w)$

$\text{FHE}_K(id_4, \ldots, id_{13})$

$\text{Enc}_K([\text{Document}])$

$\text{Enc}_K([\text{Document}])$
FHE-Based Solution (1)

Security
- $L_1$ leakage
  - #DB
  - Equality
  - PK: DB*
- $L_2$ leakage
  - access pattern
  - PK: keyword

Efficiency
- Very slow search
  - Interactive (1 round)
  - Super-linear in $|DB|$ w/ very large constants
FHE-Based Solution (2)

Security

- $L_1$ leakage
  - #DB
  - Equality
  - PK: DB*

- $L_2$ leakage
  - access pattern
  - PK: keyword

Efficiency

- Very very slow search
  - Interactive (1-round)
  - Super-linear in $|DB+Data|$ with very large constants
Tradeoffs

Efficiency

PPE/DET

SSE

FEnc/IBE

ORAM

FHE-1

FHE-2

Security
Searchable Encryption
Searchable Encryption

- Encryption that supports very slow search
  [Song-Wagner-Perrig01]

- DB encryption that supports slow search
  [Song-Wagner-Perrig01, Goh03, Chang-Mitzenmacher05]

- DB encryption that supports fast search
  [Curtmola-Garay-K.-Ostrovsky06]

- **Very slow**: linear in |DB|
- **Slow**: linear in #DB
- **Fast**: sub-linear in #DB
Searchable Encryption

- \( \text{SSE(DB)} \Leftrightarrow (K, \text{EDB}) \)
- \( \text{Token}(K, w) \Leftrightarrow t \)
- \( \text{Search}(\text{EDB}, t) \Leftrightarrow t \)
- \( \text{Dec}(K, c) \Leftrightarrow m \)

EDB = SSE
Security Definitions

- Security against chosen-keyword attack
  [Goh03, Chang-Mitzenmacher05, Curtmola-Garay-K.-Ostrovsky06]

  **CKA1:** “Protects files and keywords even if chosen by adversary”

- Security against adaptive chosen-keywords attacks
  [Curtmola-Garay-K.-Ostrovsky06]

  **CKA2:** “Protects files and keywords even if chosen by adversary, and even if chosen as a function of ciphertexts, index, and previous results”
Security Definitions

- Universal composability
  [Kurosawa-Ohtaki12, Canetti01]

**UC:** “Remains CKA2-secure even if composed arbitrarily”
Simulation-based definition

```
given the encrypted DB, encrypted files and search tokens, no adversary can learn any information about the files and the search keywords other than what can be deduced from the access and search patterns…
```

```
…even if queries are made adaptively
```

- **access pattern**: pointers to (encrypted) files that satisfy search query
- **query pattern**: whether a search query is repeated
CKA2-Security

[Curtmola-Garay-K.-Ostrovsky06]

Real World

\[ \text{Enc}_k(\text{document}) \]

q

t

Ideal World

\[ ? \in \mathcal{L}_1(\text{document}) \]

\[ ? \in \mathcal{L}_2(q) \]

@#kj^%ks#
Equivocation

Ideal World

$q_1(q)$

$q_2(q)$

$s!l)cse#@\ CE!@$

$#kj%^ks#$
### Searchable Symmetric Encryption

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Updates</th>
<th>Security</th>
<th>Search</th>
<th>Parallel</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SWP00]</td>
<td>No</td>
<td>CPA</td>
<td>$O(</td>
<td></td>
<td>f</td>
</tr>
<tr>
<td>[Goh03]</td>
<td>Yes</td>
<td>CKA1</td>
<td>$O(n)$</td>
<td>$O(n/p)$</td>
<td>SKS</td>
</tr>
<tr>
<td>[CM05]</td>
<td>No</td>
<td>CKA1</td>
<td>$O(n)$</td>
<td>$O(n/p)$</td>
<td>SKS</td>
</tr>
<tr>
<td>[CGKO06] #1</td>
<td>No</td>
<td>CKA1</td>
<td>$O(OPT)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[CGKO06] #2</td>
<td>No</td>
<td>CKA2</td>
<td>$O(OPT)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[CK10]</td>
<td>No</td>
<td>CKA2</td>
<td>$O(OPT)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[vLSDHJ10]</td>
<td>Yes</td>
<td>CKA2</td>
<td>$O(\log m)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[KO12]</td>
<td>No</td>
<td>UC</td>
<td>$O(n)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[KPR12]</td>
<td>Yes</td>
<td>CKA2</td>
<td>$O(OPT)$</td>
<td>No</td>
<td>SKS</td>
</tr>
<tr>
<td>[KP13]</td>
<td>Yes</td>
<td>CKA2</td>
<td>$O(OPT\cdot\log(n))$</td>
<td>$O(\frac{OPT}{p}\cdot\log(n))$</td>
<td>SKS</td>
</tr>
<tr>
<td>[CJJKRS13]</td>
<td>No</td>
<td>CKA2</td>
<td>$O(OPT)$</td>
<td>No</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
### SSE-1

[Curtmola-Garay-K.-Ostrovsky06]

1. **Build inverted/reverse index**

2. **Randomly permute array & nodes**

<table>
<thead>
<tr>
<th>MSFT</th>
<th>F2</th>
<th>F10</th>
<th>F11</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOG</td>
<td>F2</td>
<td>F8</td>
<td>F14</td>
</tr>
<tr>
<td>AAPL</td>
<td>F1</td>
<td>F2</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>F4</td>
<td>F10</td>
<td>F12</td>
</tr>
</tbody>
</table>

Posting list:
SSE-1
[Curtmola-Garay-K.-Ostrovsky06]

2. Randomly permute array & nodes

3. Encrypt nodes
3. Encrypt nodes

4. “Hash” keyword & encrypt pointer

\[
\begin{array}{c}
\text{GOOG} \\
\text{IBM} \\
\text{AAPL} \\
\text{MSFT}
\end{array}
\]

\[
\begin{array}{c}
F_K(\text{GOOG}) \\
F_K(\text{IBM}) \\
F_K(\text{AAPL}) \\
F_K(\text{MSFT})
\end{array}
\]

\[
\begin{array}{c}
\text{Enc(●)} \\
\text{Enc(●)} \\
\text{Enc(●)} \\
\text{Enc(●)}
\end{array}
\]
Limitations of SSE-1

- CKA1-secure
- Getting CKA2-security
  - Idea #1 [Chase-K.-10]
    - replace encryption scheme with symmetric non-committing encryption
    - only requires a PRF + XOR
    - ☹: doesn't work for dynamic data
  - Idea #2
    - Use RO + XOR
Limitations of SSE-1

- Only works for static DBs
- Problem #1: Additions
  - Given new file $F_N = (AAPL, \ldots, MSFT)$
  - Append node for $F$ to list of every $w_i$ in $F$

1. Over unencrypted index

2. Over encrypted index

<table>
<thead>
<tr>
<th>MSFT</th>
<th>F2</th>
<th>F10</th>
<th>F11</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOG</td>
<td>F2</td>
<td>F8</td>
<td>F14</td>
<td></td>
</tr>
<tr>
<td>AAPL</td>
<td>F1</td>
<td>F2</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>F4</td>
<td>F10</td>
<td>F12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$F_K(\text{GOOG})$</th>
<th>Enc(●)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_K(\text{IBM})$</td>
<td>Enc(●)</td>
</tr>
<tr>
<td>$F_K(\text{AAPL})$</td>
<td>Enc(●)</td>
</tr>
<tr>
<td>$F_K(\text{MSFT})$</td>
<td>Enc(●)</td>
</tr>
</tbody>
</table>
Limitations of SSE-1

- Problem #2: Deletions
  - When deleting a file $F_2 = (AAPL, ..., MSFT)$
  - delete all nodes for $F_2$ in every list

1. Over unencrypted index

2. Over encrypted index ???
Dynamizing SSE-1

- Static EDB $\Rightarrow$ dynamic EDB
  - Idea #1
    - Memory management over encrypted data
    - Encrypted free list
  - Idea #2
    - List manipulation over encrypted data
    - Use homomorphic encryption (here just XOR) so that pointers can be updated obliviously
  - Idea #3
    - Deletion is handled using a “dual” SSE scheme
    - Given deletion/search token for $F_2$, returns pointers to $F_2$'s nodes
    - Then add them to the free list homomorphically
State-of-the-art 2012

- [K.-Papamanthou-Roeder12]
  - Based on SSE-1
  - CKA2-secure
  - Dynamic
  - Sub-linear search (optimal)

- Limitations
  - Single keyword search
  - Very complex
State-of-the-art 2013

- [Cash-Jarecki-Jutla-Krawczyk-Rosu-Steiner13]
  - CKA2-secure
  - Dynamic
  - Sub-linear search (optimal)
  - *Boolean queries!*
1.5 million emails & attachments

EDB is 13 GB

IBM Blade HS22

Search for \( w_1 \) and \( w_2 \) less than 0.5 sec
  - \( w_1 \) in 1948 docs
  - \( w_2 \) in 1 million docs

vs. cold MySQL 5.5
  - Single term: factor of 0.1 to 2 depending on term selectivity
  - Two terms: factor of 0.1 to ? depending on term selectivity

vs. warm MySQL 5.5
  - slower by order of magnitude
Summary

- Various ways to search on encrypted data
  - PPE, FE, ORAM, FHE, SSE
- Searchable encryption
  - Best tradeoffs between security and efficiency
  - Very fast search
  - Updates
  - Boolean queries
  - Parallel and I/O-efficient search
- **Caveats**
  - Leaks (controlled) information
  - We don’t really understand what we’re leaking
What’s Next?

- Framework for understanding leakage
- Leakage attacks
  - Access pattern
    - [Islam-Kuzu-Kantarcıoglu12] attack is NP-complete but can work in practice depending on auxiliary knowledge
  - Search pattern [Liu-Zhu-Wang-Tan13]
- Countermeasures to leakage
- Improved constructions
  - SQL? [we’re asking for stronger security than CryptDB]
  - Graph databases? [Chase-K.10]
The End