Searchable symmetric encryption (SSE)

Tom Ristenpart
CS 6431
Client wants to store data up on Dropbox
- High availability, synch across devices
- Server includes much value-add functionality
  - Keyword search (find all files with “Tom” in text)
  - Deduplication (if two files same, store only one copy)
  - Thumbnail generation for images
The attached contract is ready for signature. Please print 2 documents and have Atmos ...

Upload documents

Search: “contract”

Table:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>contract</td>
<td>1, 7</td>
</tr>
<tr>
<td>signatur</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Email storage provider
Standard encryption

Email client

Upload encrypted documents

89123fdbc32a665befg8819890fbacda4320182321a1343187fabaedf3140fba

Encryption prevents keyword search on server side. Must store an index on client
Appended-PRF Searchable Encryption

Email client

Encrypt plaintext & keyed hash of keywords

89123fdbf32a665befg8819890fbcada4320182321a1343187fabaedf3140fba

\( H_K(\text{attach}) \quad H_K(\text{contract}) \quad H_K(\text{ready}) \ldots \)

Upload encrypted documents

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_K(\text{contract}) )</td>
<td>1, 7</td>
</tr>
<tr>
<td>( H_K(\text{signatur}) )</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Email storage provider
Appended-PRF Searchable Encryption

Email client
Encrypt plaintext & keyed hash of keywords
7813fed = H_K(contract)

Upload encrypted documents
Search: “7813fed”

Legacy compatible:
Works with existing plaintext storage interfaces

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>7813fed</td>
<td>1, 7</td>
</tr>
<tr>
<td>456abc3</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Email storage provider
Two more schemes to consider

(2) Unordered appended-PRFs

Randomize order of PRF values

| The attached contract is ready for signature. Please print 2 documents and have Atmos ... |
|-----------------------------------------------|-------------------------|
| $H_K(\text{contract})$ $H_K(\text{ready})$ $H_K(\text{attach})$ ... |

(3) Encrypted index

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_K(\text{contract})$</td>
<td></td>
</tr>
<tr>
<td>$H_K(\text{signatur})$</td>
<td></td>
</tr>
</tbody>
</table>

Encrypt each document list under keyword-specific key
Encrypted index schemes

Email client

Encrypt plaintext & keyed hash of keywords

7813fed = $H_K(1 \mid \text{contract1})$

$K_1' = H_K(0 \mid \text{contract})$

Upload encrypted documents

Search: “7813fed”, $K_1'$

Email storage provider

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_K(1 \mid \text{contract})$</td>
<td>$E_{K_1'}(1,7)$</td>
</tr>
<tr>
<td>$H_K(1 \mid \text{signatur})$</td>
<td>$E_{K_2'}(8,9,1,15,200)$</td>
</tr>
</tbody>
</table>

What still leaks to the server?
Informally, want to leak as little information as possible:
• Obfuscated co-occurrence matrix
• Total number of documents
• Document lengths
Cash et al. basic construction

Problem with the simple encryption of document ID lists:
• Leaks number of documents associated to (some) keyword

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>contract</td>
<td>1, 7</td>
</tr>
<tr>
<td>signatur</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

**Setup(DB):**

\[
K \leftarrow \{0, 1\} \quad // \text{pick a key } K \\
\text{For each } w \text{ in DB:} \quad // \text{all keywords in DB} \\
\quad K_i \ || \ K_i' \leftarrow H_K(w) \\
\quad c \leftarrow 0 \\
\text{For id in DB}(w): \quad // \text{all docs contain } w \\
\quad L \leftarrow H_{K_i}(c) \\
\quad d \leftarrow E_{K_i'}(id) \\
\quad \text{Put}(L,d) \text{ in dictionary} \\
\quad c++ \\
\text{Return dictionary}
\]
Cash et al. basic construction

Problem with the simple encryption of document ID lists:
• Leaks number of documents associated to (some) keyword

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>contract</td>
<td>1, 7</td>
</tr>
<tr>
<td>signatur</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Client Search(K, w):
Ki || Ki’ <- \(H_k(w)\)
Send Ki, Ki’ to server

Server Search(Ki, Ki’):
\(c \leftarrow 0\)
While \(d \neq \bot\):
\(d \leftarrow \text{Get}(H_{ki}(c))\)
Add Dec_{Ki’}(d) to IDs
Return IDs

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H_{k1}(1</td>
<td></td>
</tr>
<tr>
<td>(H_{k2}(5</td>
<td></td>
</tr>
<tr>
<td>(H_{k1}(2</td>
<td></td>
</tr>
<tr>
<td>(H_{k2}(2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Gen(1^k) : sample K_1, K_2, K_3 \in \{0, 1\}^k, generate K_4 \leftarrow \text{SKE2.Gen}(1^k) and output K = (K_1, K_2, K_3, K_4).

Enc_K(D) :

Initialization:
1. scan D and generate the set of distinct keywords \(\delta(D)\)
2. for all \(w \in \delta(D)\), generate \(D(w)\)
3. initialize a global counter \(ctr = 1\)

Building the array \(A\):
4. for \(1 \leq i \leq |\delta(D)|\), build a list \(L_i\) with nodes \(N_{i,j}\) and store it in array \(A\) as follows:
   (a) sample a key \(K_{i,0} \in \{0, 1\}^k\)
   (b) for \(1 \leq j \leq |D(w_i)| - 1:\)
       - let \(id(D_{i,j})\) be the \(j^{th}\) identifier in \(D(w_i)\)
       - generate a key \(K_{i,j} \leftarrow \text{SKE1.Gen}(1^k)\)
       - create a node \(N_{i,j} = \langle id(D_{i,j}) || K_{i,j} || \psi_K(ctr + 1)\rangle\)
       - encrypt node \(N_{i,j}\) under key \(K_{i,j-1}\) and store it in \(A\):
         \[ A[\psi_K(ctr)] \leftarrow \text{SKE1.Enc}_{K_{i,j-1}}(N_{i,j}) \]
   (c) for the last node of \(L_i\),
       - set the address of the next node to NULL: \(N_{i,|D(w_i)|} = \langle id(D_{i,|D(w_i)|}) || 0^k || \text{NULL}\rangle\)
       - encrypt the node \(N_{i,|D(w_i)|}\) under key \(K_{i,|D(w_i)|-1}\) and store it in \(A\):
         \[ A[\psi_K(ctr)] \leftarrow \text{SKE1.Enc}_{K_{i,|D(w_i)|-1}}(N_{i,|D(w_i)|}) \]
   - set \(ctr = ctr + 1\)
5. let \(s' = \sum_{w_i \in \delta(D)} |D(w_i)|\). If \(s' < s\), then set the remaining \(s - s'\) entries of \(A\) to random values of the same size as the existing \(s'\) entries of \(A\)

Building the look-up table \(T\):
6. for all \(w_i \in \delta(D)\), set \(T[\pi_{K_3}(w_i)] = \langle \text{addr}_{K_3}(N_{1,0}) || K_{1,0} \rangle \oplus f_{K_2}(w_i)\)
7. if \(|\delta(D)| < |\Delta|\), then set the remaining \(|\Delta| - |\delta(D)|\) entries of \(T\) to random values of the same size as the existing \(|\delta(D)|\) entries of \(T\)

Preparing the output:
8. for \(1 \leq i \leq n\), let \(c_i \leftarrow \text{SKE2.Enc}_{K_4}(D_i)\)
9. output \((I, c), \) where \(I = (A, T)\) and \(c = (c_1, \ldots, c_n)\)

\(\text{Trpr}_{K}(w)\) : output \(t = (\pi_{K_3}(w), f_{K_3}(w))\)

\(\text{Search}(I, t)\) :
1. parse \(t\) as \((\gamma, \eta)\), and set \(\theta \leftarrow T[\gamma]\)
2. if \(\theta \neq \perp\), then parse \(\theta \oplus \eta\) as \(\langle \alpha || K' \rangle\) and continue, otherwise return \(\perp\)
3. use the key \(K'\) to decrypt the list \(L\) starting with the node stored at address \(\alpha\) in \(A\)
4. output the list of document identifiers contained in \(L\)

\(\text{Dec}_K(c_i)\) : output \(D_i \leftarrow \text{SKE2.Dec}_{K_4}(c_i)\)

Figure 1: A non-adaptively secure SSE scheme (SSE-1)

[Curtmola et al. 2006] basic scheme
Evaluating security

Prove upper bound on information leaked:

• Assuming cryptographic components good, then nothing revealed beyond specified leakage model $\mathcal{L}$

Specify leakage $\mathcal{L}$:
$\mathcal{L}(DB)$ outputs total # of keywords in DB
$\mathcal{L}(st_L, w)$ outputs DB(w), which other queries were on w

Scheme secure if exists efficient $S$ s.t. no efficient adversary can distinguish Real vs Ideal
Cash et al. other results

- Improved HDD efficiency by batching docs
- Ability to add / delete documents
- Reported on implementation results for large document corpuses
Qualitative comparison of schemes

Appended-PRF scheme used in industry

Unordered appended-PRF used in research literature

Mimesis Aegis [Lau et al. 2014]
ShadowCrypt [He et al. 2014]

Encrypted index in literature & starting to appear in industry
Qualitative comparison of schemes

Appended-PRF scheme used in industry

Unordered appended-PRF used in research literature

Encrypted index in literature & starting to appear in industry

Ease of deployment

Formal security claims

More leakage

Less leakage
Leakage-abuse attacks

All searchable encryption leaks information about plaintexts and queries. Appended-PRF case:

\[ H_K(attach) \ H_K(contract) \ H_K(ready) \ldots \]

Upload encrypted documents

Search: “\( H_K(contract) \)”

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_K(contract) )</td>
<td>1, 7</td>
</tr>
<tr>
<td>( H_K(signatur) )</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Adversarial storage provider

[Islam, Kuzu, Kantarcioglu – 2013]
[Cash, Grubbs, Perry, R. – 2015]
Leakage-abuse attacks

All searchable encryption leaks information about plaintexts and queries. Appended-PRF case:

“Keyword 7813fed came second in Document 1”

(Keyword location)

“Keyword 7813fed searched often”

(Search frequency)

“Document 1 and 7 both contain 7813fed”

(Co-occurrence relationships)

Unordered appended-PRF: order of keywords not leaked

Encrypted index: order of keywords not leaked & leakage only after queries made

[Islam, Kuzu, Kantarcioglu – 2013]
[Cash, Grubbs, Perry, R. – 2015]
A lot of basic security questions:

• Does leakage damage confidentiality?

• How much more security does one achieve via more complex schemes?

• What adversarial capabilities are likely to arise in practice?
# Leakage-abuse attack taxonomy

<table>
<thead>
<tr>
<th>Attacker goal</th>
<th>Query recovery</th>
<th>Plaintext recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker capabilities</td>
<td>Passive</td>
<td>Observe queries and stored ciphertexts</td>
</tr>
<tr>
<td></td>
<td>Active</td>
<td>Force insertion of documents and/or queries</td>
</tr>
<tr>
<td>Document knowledge</td>
<td>Full</td>
<td>Know all plaintexts exactly</td>
</tr>
<tr>
<td></td>
<td>Partial</td>
<td>Know some plaintexts</td>
</tr>
<tr>
<td></td>
<td>Distributional</td>
<td>Know similar plaintexts</td>
</tr>
<tr>
<td></td>
<td>Support</td>
<td>Know support of distribution</td>
</tr>
</tbody>
</table>

IKK 2013 against encrypted index:  
- Query recovery: Passive  
- Full

Simulations with Enron email corpus: 80% of queries recoverable
Case studies of three attacks

1. Simple attack against appended-PRF

2. Query recovery against encrypted index schemes

3. Chosen-email attack against unordered appended-PRF
Partial plaintext recovery against appended-PRF

Known email
7813fed 18fda83 64a3b4 ...

Unknown email
ab34df 7813fed 873f63 ...

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>7813fed</td>
<td>1, 7</td>
</tr>
<tr>
<td>456abc3</td>
<td>8, 9, 1, 15, 200</td>
</tr>
</tbody>
</table>

Adversarial storage provider

[Cash, Grubbs, Perry, R. – 2015]
Partial plaintext recovery against appended-PRF

Simulations with Enron email corpus
- 30,109 emails from employee sent_mail folders
- Adversary knows 20 random emails (0.06%)
- Simply match keywords in known emails to unknown

The attached contract is ready for signature. Please print 2 documents and have Atmos execute both and return same to my attention. I will return an original for their records after ENA has signed. Or if you prefer, please provide me with the name / phone # / address of your customer and I will Fed X the Agreement.

attach contract signatur pleas print 2 document have execut both same will origin ena sign prefer provid name agreement
Randomizing hash order

Leaving hashes in document order makes attack easy

Simple change: randomize order of hashes to leak less information
(sort by hash value)

Plaintext recovery
Passive Partial

<table>
<thead>
<tr>
<th>Known email</th>
<th>7813fed</th>
<th>18fda83</th>
<th>64a3b4 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown email</td>
<td>ab34df</td>
<td>7813fed</td>
<td>873f63 ...</td>
</tr>
</tbody>
</table>

contract  file  today
Randomizing hash order

Leaving hashes in document order makes attack easy

Simple change: randomize order of hashes to leak less information
(sort by hash value)

<table>
<thead>
<tr>
<th>Known email</th>
<th>18fda83</th>
<th>64a3b4</th>
<th>7813fed</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown email</td>
<td>ab34df</td>
<td>7813fed</td>
<td>873f63</td>
<td>...</td>
</tr>
</tbody>
</table>

Order issue left implicit in prior work

Mimesis Aegis: randomizes order due to Bloom filter
ShadowCrypt: implementation randomizes order, paper does not discuss
IKK query recovery attack

Adversary knows *full plaintext corpus*
Goal is to uncover search query keywords used by client

Email client
Uniformly selects keywords to search

**Keyword** | **Documents**
---|---
$H_K$(contract) | 1, 7
$H_K$(signatur) | 8, 9, 1, 15, 200

IKK detail expensive attack using simulated annealing to solve NP-complete problem sufficient to reveal queries
Cash et al. 2016 count attack

Adversary knows **full plaintext corpus**

Goal is to uncover search query keywords used by client

---

**Email client**

Uniformly selects keywords to search

---

**Search:** \(H_K(\text{contract})\)

- Documents: *, *

**Search:** \(H_K(\text{signatur})\)

- Documents: *, *, *, *, *

---

**Adversarial storage provider**

**Keyword** | **Documents**
---|---
\(H_K(\text{contract})\) | 1, 7
\(H_K(\text{signatur})\) | 8, 9, 1, 15, 200

Attacker sees number of documents returned

Many keywords appear in a unique number of documents

Disambiguate with co-occurrence relationships
IKK vs “count” attack

Subset of Enron emails (known to attacker)
Most popular $x$ keywords considered
10% of keywords uniformly sampled and queried
Count attack with partial knowledge

![Graph showing query recovery rate vs. percentage of dataset known to server]
Chosen-email attacks
(aka file-injection attacks)

Attacker can plant chosen emails into client’s inbox

Email client

Insert new emails

Search: “$H_k(k_5)$”

Keyword | Documents
---|---
$H_k(k_0)$
$H_k(k_5)$
$H_k(k_2)$
... ... 

File 1: $k_0 \; k_1 \; k_2 \; k_3 \; k_4 \; k_5 \; k_6 \; k_7$

File 2: $k_0 \; k_1 \; k_2 \; k_3 \; k_4 \; k_5 \; k_6 \; k_7$

File 3: $k_0 \; k_1 \; k_2 \; k_3 \; k_4 \; k_5 \; k_6 \; k_7$

Search: “$H_k(k_5)$”

Say all keywords = \{k_0,k_1,...,k_7\}
Send three emails that include just shaded keywords

[Zhang et al. 2016]
Case studies of three attacks

1. Simple attack against appended-PRF

2. Query recovery against encrypted index schemes

3. Chosen-email attack against unordered appended-PRF
Summary of leakage-abuse attacks

*Provable security must be (at least) paired with empirical security analyses*

Lots of open questions:
- Leakage of richer queries
- Role of updates
- Effect of re-encryption
- Exploitability of active attacks in practice

And challenges:
- Better data sets for simulations
- Query traces
  - *Countermeasures*