Order-Revealing Encryption: How to Search on Encrypted Data

Kevin Lewi and David J. Wu
Stanford University
Searching on Encrypted Data

The information accessed from potentially exposed accounts "may have included names, email addresses, telephone numbers, dates of birth, hashed passwords (using MD5) and, in some cases, encrypted or unencrypted security questions and answers..."
Searching on Encrypted Data

Data Breach at Anthem May Forecast a Trend

By REED ABELSON and JULIE CREWSWELL   FEB. 8, 2015
Searching on Encrypted Data

Database of 191 million U.S. voters exposed on Internet: researcher
Searching on Encrypted Data

Extramarital affair website Ashley Madison has been hacked and attackers are threatening to leak data online.
Searching on Encrypted Data

eBay Asks 128 Million Customers To Change Their Passwords After Hack

Max Smolaks, May 21, 2014, 4:55 pm
Searching on Encrypted Data

Technology

Millions of hacked LinkedIn IDs advertised 'for sale'

18 May 2016 | Technology
Searching on Encrypted Data

data breaches have become the norm rather than the exception...
"because it would have hurt Yahoo’s ability to index and search messages to provide new user services”
~Jeff Bonforte (Yahoo SVP)
Searching on Encrypted Data

client holds a secret key (needed to encrypt + query the server)

server stores encrypted database

database

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alice</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Charlie</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Inigo</td>
<td>45</td>
<td>4</td>
</tr>
</tbody>
</table>
Security for Encrypted Search

<table>
<thead>
<tr>
<th>Active Adversary</th>
<th>Offline Attacks (e.g., passive snapshots)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversary sees encrypted database + queries and can interact with the database.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Snapshot Adversary</th>
<th>Online Attacks (e.g., active corruption)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversary only sees contents of encrypted database.</td>
<td></td>
</tr>
</tbody>
</table>
Security for Encrypted Search

adversary sees encrypted database + queries and can interact with the database

online attacks (e.g., active corruption)
offline attacks (e.g., passive snapshots)

adversary only sees contents of encrypted database

typical database breach: contents of database are stolen and dumped onto the web
Security / Performance Tradeoffs

- Property-Preserving Encryption [BBO’07, BCLO’09]
- Searchable Symmetric Encryption [CGKO’06]
- Oblivious RAM [GO’96]
- Fully Homomorphic Encryption [Gen’09]

Schemes deployed in practice are typically PPE-based [CipherCloud, Skyhigh Networks, Google Encrypted BigQuery]

Legacy-friendly schemes are not legacy-friendly.

Not drawn to scale
Order-Revealing Encryption [BLRSZZ’15]

secret-key encryption scheme

\[ \text{ct}_1 = \text{Enc}(\text{sk}, 123) \]
\[ \text{ct}_2 = \text{Enc}(\text{sk}, 512) \]
\[ \text{ct}_3 = \text{Enc}(\text{sk}, 273) \]

Which is greater: the value encrypted by \( \text{ct}_1 \) or the value encrypted by \( \text{ct}_2 \)?

(legacy-friendly) range queries on encrypted data
Order-Revealing Encryption [BLRSZZ’15]

given any two ciphertexts

\[ ct_1 = \text{Enc}(sk, x) \quad \text{ct}_2 = \text{Enc}(sk, y) \]

there is a public function for performing comparisons

\[ x > y \]

OPE [BCLO’09]: comparison function is numeric comparison on ciphertexts
The Landscape of ORE

- OPE [BCLO’09]
- Practical ORE [CLW’16]
- This work
- Schemes with precise leakage profile [CLW’16]
- Constructions based on mmaps [BLRSZZ’15] or obfuscation [GGGJKLSSZ’14]

not drawn to scale
Inference Attacks [NKW’15, DDC’16, GSBNR’16]

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>wpjOos</td>
<td>2wzXW8</td>
<td>SqX9I9</td>
<td>KqLUXE</td>
</tr>
<tr>
<td>XdXdg8</td>
<td>y9GFpS</td>
<td>gwilE3</td>
<td>MJ23b7</td>
</tr>
<tr>
<td>P6vKhW</td>
<td>EgN0Jn</td>
<td>S0pRJe</td>
<td>aTaeJk</td>
</tr>
<tr>
<td>orJRe6</td>
<td>KQWy9U</td>
<td>tPWF3M</td>
<td>4FBEO0</td>
</tr>
</tbody>
</table>

encrypted database + public information

frequency and statistical analysis

plaintext recovery

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>??</td>
<td>Alice</td>
<td>30-35</td>
<td>2</td>
</tr>
<tr>
<td>??</td>
<td>Bob</td>
<td>45-50</td>
<td>3</td>
</tr>
<tr>
<td>??</td>
<td>Charlie</td>
<td>40-45</td>
<td>2</td>
</tr>
<tr>
<td>??</td>
<td>??</td>
<td>40-45</td>
<td>4</td>
</tr>
</tbody>
</table>
Inference Attacks [NKW’15, DDC’16, GSBNR’16]

PPE schemes **always** reveal certain properties (e.g., equality, order) on ciphertexts and thus, are vulnerable to **offline inference attacks**.

Can we **fully** defend against offline inference attacks while remaining legacy-friendly?
This Work

Can we **fully** defend against offline inference attacks while remaining legacy-friendly?

Trivial solution: encrypt the entire database, and have client provide decryption key at query time

Desiderata: an ORE scheme that enables:
- perfect offline security
- limited leakage in the online setting

But zero online security!
ORE with Additional Structure

Focus of this work: performing range queries on encrypted data

Key primitive: order-revealing encryption scheme where ciphertexts have a “decomposable” structure

ciphertects naturally split into two components

c\text{Enc}(101)\hspace{1cm}c\text{Enc}_L(101)\hspace{1cm}c\text{Enc}_R(100)

c_{L}\hspace{1cm}c_{R}

greater than
ORE with Additional Structure

\[ \text{Enc}_L(101) \rightarrow \text{ct}_L \]
\[ \text{Enc}_R(100) \rightarrow \text{ct}_R \]

Comparison can be performed between left ciphertext and right ciphertext.

Right ciphertexts provide semantic security!

Robustness against offline inference attacks!
Encrypted Range Queries

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alice</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Charlie</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Inigo</td>
<td>45</td>
<td>4</td>
</tr>
</tbody>
</table>

store right ciphertexts in sorted order

build encrypted index

record IDs encrypted under independent key

separate index for each searchable column, and using independent ORE keys
## Encrypted Range Queries

### Encrypted database:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alice</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Charlie</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Inigo</td>
<td>45</td>
<td>4</td>
</tr>
</tbody>
</table>

Columns (other than ID) are encrypted using a semantically-secure encryption scheme.

Clients hold (secret) keys needed to decrypt and query database.
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\text{Enc}_L(40) \quad \text{Enc}_L(45)$
Encrypted Range Queries

Query for all records where \(40 \geq \text{age} \geq 45\):

\[\text{Enc}_L(40) \quad \text{Enc}_L(45)\]

<table>
<thead>
<tr>
<th>Age</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enc_R(31)</td>
<td>Enc(0)</td>
</tr>
<tr>
<td>Enc_R(41)</td>
<td>Enc(2)</td>
</tr>
<tr>
<td>Enc_R(45)</td>
<td>Enc(3)</td>
</tr>
<tr>
<td>Enc_R(47)</td>
<td>Enc(1)</td>
</tr>
</tbody>
</table>
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

- $\text{Enc}_L(40)$
- $\text{Enc}_L(45)$

use binary search to determine endpoints (comparison via ORE)

<table>
<thead>
<tr>
<th>Age</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Enc}_R(31)$</td>
<td>$\text{Enc}(0)$</td>
</tr>
<tr>
<td>$\text{Enc}_R(41)$</td>
<td>$\text{Enc}(2)$</td>
</tr>
<tr>
<td>$\text{Enc}_R(45)$</td>
<td>$\text{Enc}(3)$</td>
</tr>
<tr>
<td>$\text{Enc}_R(47)$</td>
<td>$\text{Enc}(1)$</td>
</tr>
</tbody>
</table>
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

```
Enc_L(40)
Enc_L(45)
```

use binary search to determine endpoints (comparison via ORE)
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

- Use binary search to determine endpoints (comparison via ORE).
- Return encrypted indices that match query.

<table>
<thead>
<tr>
<th>Age</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enc_R(31)</td>
<td>Enc(0)</td>
</tr>
<tr>
<td>Enc_R(41)</td>
<td>Enc(2)</td>
</tr>
<tr>
<td>Enc_R(45)</td>
<td>Enc(3)</td>
</tr>
<tr>
<td>Enc_R(47)</td>
<td>Enc(1)</td>
</tr>
</tbody>
</table>
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\text{Enc}(2)$ $\text{Enc}(3)$

The client decrypts indices to obtain a set of matching records.
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

$\text{Enc}(2) \land \text{Enc}(3)$

Records 2, 3

$\text{Enc}(r_2) \land \text{Enc}(r_3)$
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

- $\text{Enc}(2)$, $\text{Enc}(3)$
- Records 2, 3
- $\text{Enc}(r_2)$, $\text{Enc}(r_3)$

Client decrypts to obtain records.
Encrypted Range Queries

Query for all records where $40 \geq \text{age} \geq 45$:

\[ \text{Enc}(2) \quad \text{Enc}(3) \]

Records 2, 3

\[ \text{Enc}(r_2) \quad \text{Enc}(r_3) \]

some online leakage: access pattern + ORE leakage
Encrypted Range Queries

Encrypted database (view of the snapshot adversary):

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alice</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>47</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Charlie</td>
<td>41</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Inigo</td>
<td>45</td>
<td>4</td>
</tr>
</tbody>
</table>

encrypted database is semantically secure!
Perfect offline security
The Landscape of ORE

- Constructions based on mmaps [BLRSZZ’15] or obfuscation [GGGJLKSSZ’14] can provide security against snapshot adversaries.
- OPE [BCLO’09] and Practical ORE [CLW’16] are broken by inference attacks [NKW’15, DDC’16, GSBNR’16].
- This work

Not drawn to scale
Our New ORE Scheme

“small-domain” ORE with best-possible security

+ domain extension technique inspired by CLW‘16

“large-domain” ORE with some leakage
Suppose plaintext space is small: \{1,2, \ldots, N\}

associate a key with each value 

\begin{align*}
1 & \quad \vdots & \quad N \\
\hline 
  & k_1 & \\
  & k_2 & \\
  & k_3 & \\
  & \vdots & \\
  & k_N &
\end{align*}

\((k_1, \ldots, k_N)\) is the secret key (can be derived from a PRF)
Small-Domain ORE with Best-Possible Security

Encrypting a value $i$

Invariant: all positions $\leq i$ have value 1 while all positions $> i$ have value 0
Small-Domain ORE with Best-Possible Security

Encrypting a value $i$

To allow comparisons, also give out key for slot $i$
Small-Domain ORE with Best-Possible Security

Given two ciphertexts

Decrypt to learn ordering
Small-Domain ORE with Best-Possible Security

Given two ciphertexts

But this reveals $i$...
Small-Domain ORE with Best-Possible Security

**Solution:** apply random permutation $\pi$ (part of the secret key) to the slots

![Diagram showing the application of a random permutation to the slots of the secret key]
Small-Domain ORE with Best-Possible Security

Solution: apply random permutation $\pi$ (part of the secret key) to the slots

 Achieves best-possible security, but ciphertexts are big
Domain Extension for ORE

**Key idea:** decompose message into smaller blocks and apply small-domain ORE to each block

Split into two 4-bit chunks:

- Encrypt each chunk using an ORE instance with a secret key derived from the prefix.
Domain Extension for ORE

**Key idea:** decompose message into smaller blocks and apply small-domain ORE to each block

- **Keys derived from empty prefix**
  - $k_{\pi(i)}$
  - $1 \ 1 \ \cdots \ 1 \ 0 \ \cdots \ 0$

- **Keys derived from prefix $b_1 b_2 b_3 b_4$**
  - $k'_{\pi(i)}$
  - $1 \ 1 \ \cdots \ 1 \ 0 \ \cdots \ 0$

Split into two 4-bit chunks

Encrypt each chunk using an ORE instance with a secret key derived from the prefix
Domain Extension for ORE

\[
\begin{array}{cccc}
\underline{b_1} & \underline{b_2} & \underline{b_3} & \underline{b_4} \\
\underline{b_5} & \underline{b_6} & \underline{b_7} & \underline{b_8}
\end{array}
\]

Comparison proceeds block-by-block
## Domain Extension for ORE

### Case 1: Blocks encrypt distinct values

<table>
<thead>
<tr>
<th></th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$b_6$</th>
<th>$b_7$</th>
<th>$b_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{\pi(i)}$</td>
<td>$k_{\pi(1)}$</td>
<td>$k_{\pi(2)}$</td>
<td>$k_{\pi(i)}k_{\pi(i+1)}$</td>
<td>$k_{\pi(N)}$</td>
<td>$k'_{\pi(i)}$</td>
<td>$k'_{\pi(1)}$</td>
<td>$k'_{\pi(2)}$</td>
<td>$k'<em>{\pi(i)}k'</em>{\pi(i+1)}$</td>
</tr>
<tr>
<td>$k_{\pi(i)}$</td>
<td>$k_{\pi(1)}$</td>
<td>$k_{\pi(2)}$</td>
<td>$k_{\pi(i)}k_{\pi(i+1)}$</td>
<td>$k_{\pi(N)}$</td>
<td>$k'_{\pi(i)}$</td>
<td>$k'_{\pi(1)}$</td>
<td>$k'_{\pi(2)}$</td>
<td>$k'<em>{\pi(i)}k'</em>{\pi(i+1)}$</td>
</tr>
</tbody>
</table>
Domain Extension for ORE

Case 1: Blocks encrypt distinct values
Domain Extension for ORE

Subsequent blocks encrypted under independent keys

Case 1: Blocks encrypt distinct values
Domain Extension for ORE

\[ b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8 \]

Case 2: Blocks encrypt identical values
Domain Extension for ORE

Messages share a common prefix and the two next blocks are encrypted under the same key

Case 2: Blocks encrypt identical values
Domain Extension for ORE

<table>
<thead>
<tr>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_5$</td>
<td>$b_6$</td>
<td>$b_7$</td>
<td>$b_8$</td>
</tr>
</tbody>
</table>

Comparison proceeds block-by-block

Overall leakage: first block that differs
Domain Extension for ORE

Same decomposition into left and right ciphertexts:

Right ciphertexts provide semantic security!

Note: optimizations are possible if we apply this technique in a non-black-box way to the small-domain ORE. See paper for details.
The Landscape of ORE

Leakage: position of first differing block

- OPE [BCLO’09]
- Practical ORE [CLWW’16]
- This work
- Concurrent work [CLOZ’16, JP’16]
- Constructions based on mmaps [BLRSZZ’15] or obfuscation [GGGJKLSSZ’14]

not drawn to scale
## Performance Evaluation

| Scheme                        | Encrypt (μs) | Compare (μs) | |ct| (bytes) |
|-------------------------------|--------------|--------------|----------|----------|
| OPE [BCLO’09]                | 3601.82      | 0.36         | 8        |
| Practical ORE [CLW’16]       | 2.06         | 0.48         | 8        |
| This work (4-bit blocks)     | 16.50        | 0.31         | 192      |
| This work (8-bit blocks)     | 54.87        | 0.63         | 224      |
| This work (12-bit blocks)    | 721.37       | 2.61         | 1612     |

Benchmarks taken for C implementation of different schemes (with AES-NI). Measurements for encrypting 32-bit integers.
### Performance Evaluation

| Scheme                        | Encrypt (μs) | Compare (μs) | |ct| (bytes) |
|-------------------------------|--------------|--------------|-----------------|
| OPE [BCLO’09]                | 3601.82      | 0.36         | 8               |
| Practical ORE [CLW’16]       | 2.06         | 0.48         | 8               |
| This work (4-bit blocks)     | 16.50        | 0.31         | 192             |
| This work (8-bit blocks)     | 54.87        | 0.63         | 224             |
| This work (12-bit blocks)    | 721.37       | 2.61         | 1612            |

Encrypting byte-size blocks is 65x faster than OPE, but ciphertexts are 30x longer. Security is substantially better.
Conclusions

• Inference attacks render most conventional PPE-based constructions insecure

• However, ORE is still a useful building block for encrypted databases

• Introduced new paradigm for constructing ORE that enables range queries in a way that is mostly legacy-compatible and provides offline semantic security

• New ORE construction that is concretely efficient with strong security

• In paper: new impossibility results for security achievable using OPE
Open Problems

• What kind of inference attacks on possible in the online setting?
  • Indices encrypted separately, so multi-column correlations harder to infer
  • More limited leakage profile (between left and right ciphertexts)
• Can we construct small-domain OREs (with best-possible security) and sublinear (in the size of the domain) ciphertext size from simple assumptions?
• Can we construct left/right ORE (from PRFs) where both left and right ciphertexts are semantically secure?