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

The database was discovered by MacKeeper researcher Chris Vickery on March 31, in the course of searching for random phrases on the domain s3.amazonaws.com.

“IT's as bad as I expected,” he tweeted. “Bank-related. Plaintext passwords. Big name company. I've reached out to them.”
Data Breach at Anthem May Forecast a Trend

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

Database of 191 million U.S. voters exposed on Internet: researcher
Extramarital affair website Ashley Madison has been hacked and attackers are threatening to leak data online
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...
Why Not Encrypt?

data breaches have become the norm rather than the exception...
Why Not Encrypt?

“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

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>

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

server stores encrypted database
Security for Encrypted Search

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

adversary only sees contents of encrypted database

active adversary

online attacks (e.g., active corruption)

snapshot adversary

offline attacks (e.g., passive snapshots)
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.
Order-Revealing Encryption [BLRSZZ’15]

secret-key encryption scheme

\[
\begin{align*}
ct_1 &= \text{Enc}(sk, 123) \\
ct_2 &= \text{Enc}(sk, 512) \\
ct_3 &= \text{Enc}(sk, 273)
\end{align*}
\]

Which is greater: the value encrypted by \(ct_1\) or the value encrypted by \(ct_2\)?
Order-Revealing Encryption [BLRSZZ’15]

given any two ciphertexts

$ct_1 = \text{Enc}(sk, x)$

$ct_2 = \text{Enc}(sk, y)$

$x > y$

there is a public function for performing comparisons

OPE [BCLO’09]: comparison function is numeric comparison on ciphertexts
Order-Revealing Encryption [BLRSZZ’15]

given any two ciphertexts

\[ ct_1 = \text{Enc}(sk, x) \]
\[ ct_2 = \text{Enc}(sk, y) \]

best-possible security: reveal just the ordering and nothing more

\[ x > y \]

in practice: constructions reveal some additional information
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>SqX9l9</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>4FBE00</td>
</tr>
</tbody>
</table>

encrypted database + public information

frequency and statistical analysis

<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>

plaintext recovery
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
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

ciphertexts naturally split into two components

\[ \text{Enc}(101) \]

\[ \text{ct}_L \quad \text{ct}_R \]

greater than

\[ \text{Enc}_L(101) \quad \text{Enc}_R(100) \]
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>

- **build encrypted index**
- **store right ciphertexts in sorted order**
- **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 search indices
Encrypted Range Queries

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

$$sk \xrightarrow{Enc_L(40)} \xrightarrow{Enc_L(45)}$$
Encrypted Range Queries

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

<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 $40 \geq \text{age} \geq 45$:

$\text{Enc}_L(40)$

$\text{Enc}_L(45)$

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

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)
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
Encrypted Range Queries

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

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

Client decrypts indices to obtain set of matching records.
Encrypted Range Queries

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

$\text{sk}$

Records 2, 3

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

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

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

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

Records 2, 3

$\text{Enc}(r_2) \quad \text{Enc}(r_3)$

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
Our New ORE Scheme

“small-domain” ORE with best-possible security

+ domain extension technique inspired by CLWW’16

“large-domain” ORE with some leakage

first practical ORE construction that can provide best-possible offline security!
Small-Domain ORE with Best-Possible Security

Suppose plaintext space is small: \( \{1, 2, \ldots, N\} \)

<table>
<thead>
<tr>
<th>1</th>
<th>( k_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( k_2 )</td>
</tr>
<tr>
<td>3</td>
<td>( k_3 )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( N )</td>
<td>( k_N )</td>
</tr>
</tbody>
</table>

associate a key with each value

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

Encrypting a value $i$

\[
\begin{array}{ccccccc}
1 & 1 & \cdots & 1 & 0 & \cdots & 0 \\
\end{array}
\]

\textbf{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
Small-Domain ORE with Best-Possible Security

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

![Diagram showing a small domain ORE with random permutation $\pi$ applied to slots.

Smallest Domain ORE with Best-Possible Security

Includes index $\pi(i)$

Semantically secure (right ciphertext)

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

- 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

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.
Performance Evaluation

| Scheme                      | Encrypt (μs) | Compare (μs) | |ct| (bytes) |
|-----------------------------|--------------|--------------|-----------------|
| OPE [BCLO’09]               | 3601.82      | 0.36         | 8               |
| Practical ORE [CLWW’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 [CLWW’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
Questions?

Paper:  https://eprint.iacr.org/2016/612
Website: https://crypto.stanford.edu/ore/
Code:   https://github.com/kevinlewi/fastore