Oxymoron: Encrypted (Database) Search

Srinivasan Narayanamurthy (Srini)
NetApp
Agenda

- Survey
- Homomorphic Encryption 101
- Encrypted Search
- Tradeoffs
  - Leakage
  - Functionality
- Encrypted Databases
Survey

- Non-cryptographic methods
  - Differential Privacy (Noise)
  - Data Anonymization
  - Data Fragmentation

- Secret-Sharing based methods
  - Verifiable (collaborative)
  - Order Preserving

- Index based methods
  - Bucketization
  - Order-preserving
  - Searchable
Survey (Continued)

- Cryptographic
  - Functional Encryption
  - Searchable Encryption
  - Secure-Multiparty Computation
  - Homomorphic Cryptosystems
    - Fully (FHE)
    - Partial (PHE)

- State-of-the-art Systems
  - Systems based on Homomorphic (CryptDB)
  - Client-server splitting approaches (Monomi, Silverline)
  - Trusted Hardware Systems (TrustedDB, Cipherbase)
Symmetric Encryption

Key: 000102030405060708090a0b0c0d0e0f

The quick brown fox jumps over the lazy dog

Encrypt

a7be1a6997ad739bd8c9ca451f618b61
b6ff744ed2c2c9bf6c590cbf0469bf41
47f7f7bc95353e03f96c32bcfd8058df

Decrypt

The quick brown fox jumps over the lazy dog

Key: 000102030405060708090a0b0c0d0e0f
Asymmetric Encryption

Public key: 000102030405060708090a0b0c0d0e0f

Encrypt

The quick brown fox jumps over the lazy dog

a7be1a6997ad739bd8c9ca451f618b61
b6ff744ed2c2c9bf6c590cbf0469bf41
47f7f7bc95353e03f96c32bcfd8058df

Decrypt

a7be1a6997ad739bd8c9ca451f618b61
b6ff744ed2c2c9bf6c590cbf0469bf41
47f7f7bc95353e03f96c32bcfd8058df

The quick brown fox jumps over the lazy dog

Private key: 47b6ffedc2be19bd5359c32bcfd8dff5
AES + CBC mode

Key: 000102030405060708090a0b0c0d0e0f

Initialization Vector (IV)
0000000001...

The quick brown
Key: a7be1a6997a...

AES

fox jumps over
Key: b6ff744ed2c...

AES

the lazy dog
Key: 47f7f7bc953...

AES

Variable IV => Non-deterministic
AES + CBC mode (IV changes)

Key: 000102030405060708090a0b0c0d0e0f

Variable IV => Non-deterministic
Non-deterministic Encryption

The quick brown fox jumps over the lazy dog

Encrypt

The quick brown fox jumps over the lazy dog

Encrypt

Example: AES + CBC + Variable IV

Key: 000102030405060708090a0b0c0d0e0f

a7be1a6997ad739bd8c9ca451f618b61b6ff744ed2c2c9bf6c590cbf0469bf4147f7f7bc95353e03f96c32bcfd8058df

fa636a2825b339c940668a3157244d17247240236966b3fa6ed2753288425b6c69c4e0d86a7b0430d8cdb78070b4c55a

Key: 000102030405060708090a0b0c0d0e0f
AES + ECB mode

Key: 000102030405060708090a0b0c0d0e0f
Deterministic Encryption

Key: 000102030405060708090a0b0c0d0e0f

Encrypt

The quick brown fox jumps over the lazy dog

The quick brown fox jumps over the lazy dog

Key: 000102030405060708090a0b0c0d0e0f

Example: AES + ECB
Order Preserving Encryption

<table>
<thead>
<tr>
<th>Value</th>
<th>Enc (Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0001102789d5f50b2befffd9f3dca4ea7</td>
</tr>
<tr>
<td>2</td>
<td>0x0065fda789ef4e272bcf102787a93903</td>
</tr>
<tr>
<td>3</td>
<td>0x009b5708e13665a7de14d3d824ca9f15</td>
</tr>
<tr>
<td>4</td>
<td>0x04e062ff507458f9be50497656ed654c</td>
</tr>
<tr>
<td>5</td>
<td>0x08db34fb1f807678d3f833c2194a759e</td>
</tr>
</tbody>
</table>

\[ x; y \rightarrow Enc\ x; Enc'\ y(\) \]

Example: AES + FFX
**Homomorphic Encryption**

\[ Enc(1) \]

\[ \begin{array}{c}
7ad5fda789ef4e272bca100b3d9ff59f \\
bd6e7c3df2b5779e0b61216e8b10b689
\end{array} \]

\[ Enc(1) + Enc(1) = Enc(2) \]

\[ \begin{array}{c}
7ad5fda789ef4e272bca100b3d9ff59f \\
bd6e7c3df2b5779e0b61216e8b10b689
\end{array} \]

\[ + \]

\[ 7a9f102789d5f0b2beff9f3dca4ea7 \]

**Encryption key is not an input**
The Spectrum

Fully Homomorphic Encryption
( any function )

Impractical

Partial Homomorphic Encryption
Paillier Cryptosystem
( + )
Expensive

ElGamal Cryptosystem
( x )

Order Preserving Encryption
( ≤ )

Deterministic Encryption
( = = )

Non-deterministic Encryption
( Ø )

Practical
## Performance

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Space for one integer (bits)</th>
<th>Time for one operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Homomorphic Encryption</td>
<td>$2^{14}$</td>
<td>Cosmic time scales</td>
</tr>
<tr>
<td>Paillier</td>
<td>2048</td>
<td>$\sim$ ms</td>
</tr>
<tr>
<td>ElGamal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic Encryption</td>
<td>128</td>
<td>$\sim$ $\mu$s</td>
</tr>
</tbody>
</table>
Encrypted Search – Tradeoffs

- **Efficiency**
  - Computation & Communication complexity
  - E.g. sub-linear index

- **Security**
  - Leakage
  - E.g. Index, search & access pattern

- **Functionality**
  - Query expressiveness
  - (equality, boolean, subset, range queries, inner products)
Is Encryption == Security?

A Crypto Nerd's Imagination:

His laptop's encrypted. Let's build a million-dollar cluster to crack it.

No good! It's 4096-bit RSA!

Blast! Our evil plan is foiled!

What Would Actually Happen:

His laptop's encrypted. Drug him and hit him with this $5 wrench until he tells us the password.

Got it.
## Leakage Profile

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects that leak</td>
<td>Data objects, queries, query response (access control rules)</td>
</tr>
<tr>
<td>Type of information leaked</td>
<td>Same value, Matches the intersection of two sets</td>
</tr>
<tr>
<td>Which operation leaks</td>
<td>= (say equality) &gt;, &lt; (say, range)</td>
</tr>
<tr>
<td>Party that learns the leakage</td>
<td>Provider, Querier, Server</td>
</tr>
</tbody>
</table>
## Information leaked by Objects

<table>
<thead>
<tr>
<th>Information</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>String length, set cardinality, tree rep. of object</td>
</tr>
<tr>
<td>Identifiers</td>
<td>Pointers to objects</td>
</tr>
<tr>
<td>Predicates</td>
<td>Additional information, say, a. within a common (known) range b. matches the intersection of 2 clauses within a query</td>
</tr>
<tr>
<td>Equalities</td>
<td>Objects that have same value</td>
</tr>
<tr>
<td>Order (or more)</td>
<td>Numerical/lexicographic ordering of objects, or perhaps even partial plaintext data</td>
</tr>
</tbody>
</table>
# Queries on Encrypted Data

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Type of Queries</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured (DBs), Semi (XML/JSON)</td>
<td>Relational Algebra (SQL)</td>
<td>Set (Union, Intersection, Difference, Cartesian product), Selection, Projection, Join</td>
</tr>
<tr>
<td></td>
<td>Associative Arrays (NoSQL)</td>
<td>(Semi-ring): Construction, Find, AA (+, x), AA Element-wise (x)</td>
</tr>
<tr>
<td></td>
<td>Linear Algebra (NewSQL)</td>
<td>Construction, Find, Matrix (+, x), Element-wise (x)</td>
</tr>
<tr>
<td>Unstructured</td>
<td>Content-based</td>
<td>Query-by-example, Fuzzy queries → Exhaustive search Eg. filesystems</td>
</tr>
<tr>
<td></td>
<td>Information Retrieval</td>
<td>Indexes</td>
</tr>
<tr>
<td>Mixed</td>
<td>SELECT * FROM patient WHERE (age &gt; 40) AND (X-ray CONTAINS “lung cancer”)</td>
<td></td>
</tr>
</tbody>
</table>
## Base Queries

<table>
<thead>
<tr>
<th>Approach</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>Modifies data insertions and query requests</td>
<td>Property (equality or order) preserving, boolean queries and joins by combining the results of PPE. (CryptDB)</td>
</tr>
<tr>
<td>Custom</td>
<td>Special purpose protected indices</td>
<td>Inverted Index, Tree Traversal, Custom indices (Graph)</td>
</tr>
<tr>
<td>Obliv</td>
<td>Obscures object identifiers (say, pointers)</td>
<td>ORAM</td>
</tr>
</tbody>
</table>
# Composed Queries

<table>
<thead>
<tr>
<th>Composed Query</th>
<th>Base Query Calls</th>
<th>Additional Storage</th>
<th>Leakage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality (EQ)</td>
<td>1 range</td>
<td>none</td>
<td>Same as range</td>
</tr>
<tr>
<td>Disjunction (OR) of ( k ) EQs (or ranges)</td>
<td>( k ) EQs (or ranges)</td>
<td>None</td>
<td>Identifiers of records matching each clause;</td>
</tr>
<tr>
<td>Conjunction (AND) of ( k ) EQs</td>
<td>1 EQ</td>
<td>Same as EQ</td>
<td></td>
</tr>
<tr>
<td>Stemming</td>
<td>1 EQ</td>
<td>1</td>
<td>Identifiers of records sharing stem;</td>
</tr>
<tr>
<td>Proximity</td>
<td>1 EQ</td>
<td>Max. no. of neighbors</td>
<td>No leakage if refresh between queries;</td>
</tr>
<tr>
<td>Range w/ small domain</td>
<td>((2 + r)) EQs</td>
<td>1</td>
<td>No leakage if refresh between queries</td>
</tr>
<tr>
<td>Range</td>
<td>OR of (2 long ( m )) EQs</td>
<td>( \log m )</td>
<td>Distributional info;</td>
</tr>
<tr>
<td>Negation</td>
<td>AND of 2 ranges</td>
<td>1</td>
<td>Same as OR of ranges</td>
</tr>
<tr>
<td>Substring ((s = q))</td>
<td>1 EQ</td>
<td>( s - n + 1 )</td>
<td>Identifiers of records sharing ( n )-grams</td>
</tr>
<tr>
<td>Substring ((s \leq q))</td>
<td>1 range</td>
<td>( s - n + 1 )</td>
<td>Same as range, on ( n )-grams</td>
</tr>
<tr>
<td>Anchored substring ((s \geq q))</td>
<td>AND of ((q - n + 1)) EQs</td>
<td>( s - n + 1 )</td>
<td>( n, k, q ), if EQ, leaks ( \bullet )</td>
</tr>
<tr>
<td>Substring</td>
<td>OR of ((s - n + 1)) ANDs of ((q - n + 1)) EQs</td>
<td>( s - n + 1 )</td>
<td>( n, k, q ), if EQ, leaks ( \bullet )</td>
</tr>
<tr>
<td>Anchored Wildcard</td>
<td>AND of ((q - n + 1)) EQs</td>
<td>( s - n + 1 )</td>
<td>( n, k, q ), if EQ, leaks ( \bullet )</td>
</tr>
<tr>
<td>Wildcard</td>
<td>OR of ((s - n + 1)) ANDs of ((q - n + 1)) EQs</td>
<td>( s - n + 1 )</td>
<td>( n, k, q ), if EQ, leaks ( \bullet )</td>
</tr>
</tbody>
</table>

\( k \) – # of clauses in Boolean; \( r \) – # of query results; \( s \) – max length of the string; \( q \) – length of the query string; \( n \) – length of the grams

“Anchored” – search at the beginning or the end of the string

\( \bullet \) If EQ, leaks \( \bullet \)
Systems Landscape

Non-Homomorphic

Partial Homomorphic

Full Homomorphic

CryptDB

Monomi

“Blob” Store

AWS GovCloud

No Secure Location

Client

Secure Server

Crypto Coprocessor

FPGA

TrustedDB

Cipherbase
Encrypted Databases

- **CryptDB**
  - Query-aware encryption schemes
    - RND, HE, DET, OPE
  - Architecture
    - SQL-aware encryption
    - Adjustable query-based encryption
    - Chain cryptographic keys in user passwords
  - Supports only 2 out of 22 queries in TPC-H

- **Monomi (OLAP)**
  - Layout optimizer, Query planner
  - Intermediate results. Ex.: SUM / GROUP BY / HAVING
  - Supports 19 out of 22 queries
Summary

- Application security
  - DBMS is only a part of the overall system stack
- Usability
  - Clients need tools and interpretable security models to navigate security-performance tradeoffs
- Connections to other areas of security
  - Data privacy, access-control, auditing
Thank you!

MORDAC, THE PREVENTER OF INFORMATION SERVICES.
SECURITY IS MORE IMPORTANT THAN USABILITY.

IN A PERFECT WORLD, NO ONE WOULD BE ABLE TO USE ANYTHING.

To complete the log-in procedure, stare directly at the sun.