Building systems that compute on encrypted data

Raluca Ada Popa
MIT
Compromise of confidential data is prevalent

Privacy, security still top cloud concerns
Asia Cloud Forum editors | November 13, 2013
Asia Cloud Forum

An online survey of Microsoft partners has revealed that traditional concerns about
Problem setup

clients

server

no computation
storage
 encryption

computation
databases, web applications, mobile applications, machine learning, etc.

??
Current systems strategy

Prevent attackers from breaking into servers
Lots of existing work

- Checks at the operating-system level
- Language-based enforcement of a security policy
- Static or dynamic analysis of application code
- Checks at the network level
- Trusted hardware

...
Data still leaks even with these mechanisms because attackers eventually break in!
Attacker examples

**Attacker:**
- hackers
- cloud employees
- government

**Reason they succeed:**
- software is complex
- insiders: legitimate server access!
- e.g., physical access

Increasingly many companies store data on external clouds

Accessed private data according to

According to hackers, cloud employees, and government insiders: legitimate server access!
My work

Systems that protect confidentiality even against attackers with access to all server data
My approach

Servers store, process, and compute on encrypted data *in a practical way*
Computing on encrypted data in cryptography
[Rivest-Adleman-Dertouzos’78]

Fully homomorphic encryption (FHE) [Gentry’09]

prohibitively slow, e.g., slowdown $\times 1,000,000,000$

My work: practical systems

real-world performance + large class of real applications + meaningful security
My contributions

**System:**

**Databases:**  
- **CryptDB** [SOSP’11][CACM’12]  
  mOPE, adjJOIN  
  [Oakland’13]

**Web apps:**  
- **Mylar** [NSDI’14]  
  multi-key search

**Mobile apps:**  
- **PrivStats** [CCS’11]  
- **VPriv** [Usenix Security’09]

**Server under attack:**

**Theory:**

In general: **Functional encryption** [STOC’13] [CRYPTO’13]
Combine systems and cryptography

1. identify core operations needed

2. multiple specialized encryption schemes
   New schemes:
   - mOPE, adjJOIN for CryptDB
   - multi-key search for Mylar

3. Design and build system

strawman:
one generic scheme (FHE)
# My contributions

<table>
<thead>
<tr>
<th>System:</th>
<th>Server under attack:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Databases:</strong> CryptDB</td>
<td>![DB server]</td>
</tr>
<tr>
<td><strong>Web apps:</strong> Mylar</td>
<td>![web app server]</td>
</tr>
<tr>
<td><strong>Mobile apps:</strong> PrivStats, VPriv</td>
<td></td>
</tr>
</tbody>
</table>

**Theory:** Functional encryption

In general:
CryptDB

[SOSP’11: Popa-Redfield-Zeldovich-Balakrishnan]

First practical database system (DBMS) to process most SQL queries on encrypted data.
Related work

- **Systems work:** [Hacigumus et al.’02][Damiani et al.’03][Ciriani et al’09]
  - no formal confidentiality guarantees
  - restricted functionality
  - client-side filtering

- **Theory work:**
  - General computation: FHE  [Gentry’09]
    - very strong security: forces slowdown - many queries must always scan and return the whole DB
    - prohibitively slow ($10^9x$)
  - Specialized schemes [Amanatidis et al.’07][Song et al.’00][Boldyreva et al.’09]
Setup

trusted client-side  under passive attack

Use cases:
- Outsource DB to the cloud (DBaaS)
- e.g. Encrypted BigQuery
- Local cluster: hide DB content from sys. admins.
Setup

- Application
- Proxy
- DB server

**trusted client-side**

- plain query → Proxy
- decrypted results → Application

**under passive attack**

- transformed query → Proxy
- encrypted results → DB server

- Stores schema and master key
- No query execution

**related work**

- Not achieving my purpose, plus Robert did not like the yellow box

- Computation on encrypted data ≈ regular computation
Example

Application

SELECT * FROM emp

Proxy

SELECT * FROM table1

Randomized encryption (RND) - semantic

table1/emp

<table>
<thead>
<tr>
<th>col1/rank</th>
<th>col2/name</th>
<th>col3/salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x4be219</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x95c623</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x2ea887</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x17cea7</td>
</tr>
</tbody>
</table>

60
100
800
100
Example

Application

SELECT \textit{sum(salary)} FROM emp

Proxy

SELECT \textit{cdb\_sum(col3)} FROM table1

\textbf{table1 (emp)}

\begin{tabular}{|c|c|c|}
\hline
\textbf{col1/rank} & \textbf{col2/name} & \textbf{col3/salary} \\
\hline
 & & x9eab8 \\
 & & x638e5 \\
 & & x122eb4 \\
 & & x578b34 \\
\hline
\end{tabular}

1060

1

Deterministic encryption (DET)

“Summable” encryption (HOM) - semantic

Example Application Proxy
Techniques

1. Use SQL-aware set of efficient encryption schemes (meta technique!)
   - Most SQL can be implemented with a few core operations

2. Adjust encryption of data based on queries

3. Query rewriting algorithm
1. SQL-aware encryption schemes

<table>
<thead>
<tr>
<th>Security</th>
<th>Scheme</th>
<th>Construction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>≈ semantic security</td>
<td>RND</td>
<td>AES in UFE</td>
<td>data moving</td>
</tr>
<tr>
<td>reveals only repeat pattern</td>
<td>HOM</td>
<td>Paillier</td>
<td>addition</td>
</tr>
<tr>
<td>reveals only order</td>
<td>SEARCH</td>
<td>Song et al., ‘00</td>
<td>word search</td>
</tr>
</tbody>
</table>

SQL operations:
- e.g., SELECT, UPDATE, DELETE, INSERT, COUNT
- e.g., SUM, +
- restricted ILIKE
  - e.g., =, !, IN, GROUP BY, DISTINCT
- e.g., >, <, ORDER BY, ASC, DESC, MAX, MIN, GREATEST, LEAST

Encryption schemes:
- AES in UFE
- AES in CMC
- our new scheme

Security properties:
- x < y \iff Enc(x) < Enc(y)
How to encrypt each data item?

Goals:
1. Support queries
2. Use most secure encryption schemes

Challenge: may not know queries ahead of time
Onion
Onion of encryptions

Adjust encryption: strip off layer of the onion
Onions of encryptions

1 column

<table>
<thead>
<tr>
<th>RND</th>
<th>DET</th>
<th>JOIN</th>
<th>value</th>
</tr>
</thead>
</table>

Onion Equality

3 columns

<table>
<thead>
<tr>
<th>RND</th>
<th>OPE</th>
<th>value</th>
</tr>
</thead>
</table>

Onion Order

<table>
<thead>
<tr>
<th>HOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>int value</td>
</tr>
</tbody>
</table>

Onion Add

| OR |

| SEARCH |
| text value |

Onion Search

Same key for all items in a column for same onion layer
Onion evolution

- Start out the database with the most secure encryption scheme
- If needed, adjust onion level
  - Proxy gives decryption key to server
  - Proxy remembers onion layer for columns

Lowest onion level is never removed
Example

Logical table:

<table>
<thead>
<tr>
<th>rank</th>
<th>name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>'CEO'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'worker'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Physical table:

RND

DET

JOIN

'CEO'

Onion Equality

SELECT * FROM emp WHERE rank = 'CEO'
Example (cont’d)

UPDATE table1 SET col1-OnionEq = Decrypt_RND(key, col1-OnionEq)

SELECT * FROM table1 WHERE col1-OnionEq = xda5c0407
Security threshold

Data owner can specify minimum level of security

CREATE TABLE emp (..., credit_card SENSITIVE integer, ...)

RND, HOM, DET for unique fields

≈ semantic security
Security guarantee

Columns annotated as sensitive have semantic security (or similar).

Encryption schemes exposed for each column are the most secure enabling queries.

equality repeats sum semantic
no filter semantic

common in practice
Limitations & Workarounds

Queries not supported:

- More complex operators, e.g., trigonometry
- Certain combinations of encryption schemes:
  - e.g., $\text{salary} + \text{raise} > 100K$

use query splitting, query rewriting

HOM
Implementation

No change to the DBMS!
Largely no change to apps!
Evaluation

1. Does it support real queries/applications?
2. What is the resulting confidentiality level?
3. What is the performance overhead?
Real queries/applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Encrypted columns</th>
<th># cols with queries not supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>phpBB</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>HotCRP</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>grad-apply</td>
<td>103</td>
<td>0</td>
</tr>
<tr>
<td>TPC-C</td>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>sql.mit.edu</td>
<td>128,840</td>
<td>1,094</td>
</tr>
</tbody>
</table>

SELECT 1/log(series_no+1.2) ...

... WHERE sin(latitude + PI()) ...
## Confidentiality level

<table>
<thead>
<tr>
<th>Application</th>
<th>Encrypted columns</th>
<th>Min level: ≈semantic</th>
<th>Min level: DET/JOIN</th>
<th>Min level: OPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>phpBB</td>
<td>23</td>
<td>21</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HotCRP</td>
<td>22</td>
<td>18</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>grad-apply</td>
<td>103</td>
<td>95</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>TPC-C</td>
<td>92</td>
<td>65</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>sql.mit.edu</td>
<td>128,840</td>
<td>80,053</td>
<td>34,212</td>
<td>13,131</td>
</tr>
</tbody>
</table>

Final onion state:

- Most columns at semantic
- Most columns at OPE were less sensitive
Performance

MySQL:
- Application 1
- Application 2

CryptDB:
- Application 1
- Application 2

DB server throughput
- Plain database

Latency

Hardware: 2.4 GHz Intel Xeon E5620 – 8 cores, 12 GB RAM
TPC-C performance

Latency (per query): 0.10ms MySQL vs. 0.72ms CryptDB

Throughput loss over MySQL: 26%

No cryptography at the DB server in the steady state!

Homomorphic addition
Adoption

http://css.csail.mit.edu/cryptdb/

Google

Encrypted BigQuery

“CryptDB was really eye-opening in establishing the practicality of providing a SQL-like query interface to an encrypted database”

“CryptDB was [...] directly influential on the design and implementation of Encrypted BigQuery.”

SAP

SEEED implemented on top of the SAP HANA DBMS

MIT Lincoln Laboratory

Encrypted version of the D4M Accumulo NoSQL engine

sql.mit.edu

Users opted-in to run Wordpress over our CryptDB source code
Demo
Attack to all servers?

users

application

CryptDB proxy

DB server

CryptDB SQL queries on encrypted DB

Secret

SQL queries on encrypted DB
Attack to all servers?
Mylar

- Framework for building web applications
- Protects confidentiality against attacks to all servers

[NSDI’14: Popa-Stark-Valdez-Helfer-Zeldovich-Kaashoek-Balakrishnan]
Overview

Plaintext data exists only in browsers
Computation in web applications

1. Mylar is a client-side application framework

2. Non client-side computation: meta technique!
   - data sharing
   - search
   - Challenges:
     - Active attacker
     - key certification
     - Multiple keys
     - multi-key search
Applications

http://css.csail.mit.edu/mylar/

Few developer annotations to secure an application, modest overhead
My contributions

System:

Databases: **CryptDB** [SOSP’11][CACM’12]  
   mOPE, adjJOIN  
   [Oakland’13]

Web apps: **Mylar** [NSDI’14]  
   multi-key search

Mobile apps: **PrivStats** [CCS’11]  
   **VPriv** [Usenix Security’09]

Server under attack:

DB server

web app server

DB server

mobile app server

Theory: **Functional encryption** [STOC’13] [CRYPTO’13]
   - Proof of concept for general functions
   - Solved old open problem: **reusable garbled circuits**
System design principles

Assume all server data will leak!
Store, process, and compute on encrypted data.

Technique for practicality:
1. identify core operations
2. use an efficient encryption scheme for each
Future work

Other systems computing on encrypted data:

- Genomics analytics and machine learning
Future work

Other systems computing on encrypted data:

Genomics analytics and machine learning

Big data & compression

- big data
- compressed big data
- encrypted big data
- encrypted compressed big data

How to compute on it??
Future work

Other systems computing on encrypted data:

- Genomics analytics and machine learning
- Big data & compression

Security beyond confidentiality:

- Correctness of computation

Client-side security

systems <-> crypto
Collaborators

**CryptDB:** Catherine Redfield, Nickolai Zeldovich, Hari Balakrishan, Aaron Burrows

**Mylar:** Steven Valdez, Jonas Helfer, Nickolai Zeldovich, Frans M. Kaashoek, Hari Balakrishnan

**PrivStats, VPriv:** Andrew Blumberg, Hari Balakrishnan, Frank H. Li

**Functional encryption:** Shafi Goldwasser, Yael Kalai, Vinod Vaikuntanathan, Nickolai Zeldovich

and others for other projects.
Future work

Other systems computing on encrypted data:
- Genomics analytics and machine learning
- Big data & compression

Security beyond confidentiality:
- Correctness of computation

Client-side security

THANK YOU!