Using cryptography in databases and web applications

Nickolai Zeldovich
MIT CSAIL

Joint work with Raluca Ada Popa, Stephen Tu, Emily Stark, Jonas Helfer, Steven Valdez, Hari Balakrishnan, Frans Kaashoek, Sam Madden
Problem: private data breaches

- No computation
  - Storage
  - Encryption (encrypted files, email)

Computation
  - Databases, web applications, mobile applications, machine learning, etc.

??
Common approach: prevent break-ins

Enforced at many levels: operating system, hardware, network, programming language, ...
Bad guys find ways to break in

• Complex software has bugs
  – Attackers find and exploit vulnerabilities

• Many people have access to infrastructure
  – Server administrators
  – Cloud / data center employees
  – Anyone that breaks into their accounts

• Compromises are inevitable
New approach: practical processing of encrypted data

client

server

Strawman:
CryptDB setup

trusted client-side

under attack

Application

Proxy

encrypted DB

Database server

Stores schema and master key

Minimal or no query execution

plain query

decrypted results

transformed query

encrypted results

Example

```
SELECT * FROM emp
```

```
SELECT * FROM table1
```

```
  col1/rank | col2/name | col3/salary
  60        | 100       | x4be219
  800       | 100       | x95c623
  60        |           | x2ea887
  100       |           | x17cea7
```

Randomized encryption (RND) - _semantic_
```
SELECT sum(salary) 
FROM emp
```

```
SELECT cdb_sum(col3) 
FROM table1
```

```
<table>
<thead>
<tr>
<th>col1/rank</th>
<th>col2/name</th>
<th>col3/salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x9eb81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x638e54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x122eb4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x578b34</td>
</tr>
</tbody>
</table>
```

---

**Example**

```
SELECT  sum(salary) 
FROM 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>x9eb81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x638e54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x122eb4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x578b34</td>
</tr>
</tbody>
</table>
```

---

**Deterministic encryption (DET)**

---

**Deterministic homomorphic encryption (HOM)**

---

**Summable encryption (DET)**

---

**Application**

---

**Proxy**

---

```
SELECT sum(salary)
FROM emp
```

---

```
SELECT cdb_sum(col3)
FROM table1
```

---

```
<table>
<thead>
<tr>
<th>col1/rank</th>
<th>col2/name</th>
<th>col3/salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x9eb81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x638e54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x122eb4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x578b34</td>
</tr>
</tbody>
</table>
```

---

```
<table>
<thead>
<tr>
<th>1060</th>
</tr>
</thead>
<tbody>
<tr>
<td>x72295a</td>
</tr>
</tbody>
</table>
```

---

```
<table>
<thead>
<tr>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>800</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>
```
Example

Application

```
SELECT sqrt(sum(salary))
FROM emp
```

Proxy

```
SELECT cdb_sum(col3)
FROM table1
```

```
<table>
<thead>
<tr>
<th>col1/rank</th>
<th>col2/name</th>
<th>col3/salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x9eab81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x638e54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x122eb4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x578b34</td>
</tr>
</tbody>
</table>
```

```
SELECT sqrt(sum(salary))
FROM emp
<table>
<thead>
<tr>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>800</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>
```

Example Application Proxy
SELECT \( \sqrt{\text{sum}(\text{salary})} \)
FROM emp

---

**Example**

- Application
  - SELECT \( \sqrt{\text{sum}(\text{salary})} \)
  - FROM emp
  - 32.55

- Proxy
  - SELECT cdb_sum(col3)
  - FROM table1
  - 1060
  - x72295a

- Table 1 (emp)
  - | col1/rank | col2/name | col3/salary |
  - |-----------|-----------|------------|
  - | x9eab81   | x638e54   | x578b34    |
  - | x122eb4   |           |            |

- Key
  - x72295a
  - x9eab81
  - x638e54
  - x122eb4
  - x578b34
Techniques

• Compute on encrypted data at the server
  – Use SQL-aware set of efficient encryption schemes
  – Adjust encryption of data based on queries

• Compute on decrypted data at the proxy
  – Can decrypt → can perform any computation
  – Choose optimal split to reduce bandwidth, proxy load
### SQL-aware encryption schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Construction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RND</td>
<td>AES in UFE</td>
<td>data moving</td>
</tr>
<tr>
<td>HOM</td>
<td>Paillier</td>
<td>addition</td>
</tr>
<tr>
<td>SEARCH</td>
<td>Song et al., ‘00</td>
<td>word search</td>
</tr>
<tr>
<td>DET</td>
<td>AES in CMC</td>
<td>equality</td>
</tr>
<tr>
<td>JOIN</td>
<td>our new scheme</td>
<td>join</td>
</tr>
<tr>
<td>OPE</td>
<td>our new scheme</td>
<td>order</td>
</tr>
</tbody>
</table>

#### Security
- **Approximate semantic security**
  - Reveals only repeat pattern
  - Reveals only order

#### 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

---

Security reveals only repeat pattern.
Onion of encryptions

Adjust encryption: strip off layer of the onion
CryptDB works well in practice

• Supports many database applications
  – Web sites, transactional processing, data analytics
  – Never reveals plaintext data on database server

• Modest performance overheads
  – 20-30% throughput loss for typical benchmarks

• Approach now used by Google (among others)
  – Encrypted BigQuery service
Compromised app. server?

users

application

CryptDB proxy

DB server

CryptDB SQL queries on encrypted DB
Compromised app. server?

Application

CryptDB proxy

DB server

Secret

CryptDB proxy

Secret

CryptDB proxy

Secret

users
Mylar: browser-side encryption

Decrypting data exists only in users’ browsers.
Challenge: computation in web applications

1. Client-side application framework
   - Most computation happens in client’s web browser (Javascript code)

2. Non client-side computation:
   - Data sharing – need a way to manage keys
   - Keyword search – need new cryptosystem: documents encrypted with *many keys*
Mylar supports many applications

- Ported 6 applications to Mylar
- Performance overheads are modest
- Data privacy despite server compromises
Future research directions

• Practical cryptography
  – Computing on data encrypted w/ many keys
  – Delegating limited functions over encrypted data

• Practical systems
  – Auditing for data disclosures
  – Protecting end-user computers