Why Your Encrypted Database Is Not Secure

Paul Grubbs     Tom Ristenpart
Vitaly Shmatikov
Outsourced Applications Today
Encrypt the data!
Encrypt the Data

App functionality no longer works :(
Encrypt the Data

- Searchable encryption
- Deterministic encryption
- Order-revealing encryption

use property-revealing encryption (PRE)
Building “Secure” Systems

Server

Encrypted data
Building “Secure” Systems

Server

“computing on encrypted data”
Building “Secure” Systems

- CryptDB (SOSP 2011)
- Mylar (NSDI 2014)
- Seabed (OSDI 2016)
- Arx
- Many others
- Lots of industry and government interest!!
What They Claim

CryptDB is a system that provides practical and provable confidentiality.

Using the “Sensitive” annotation, CryptDB ensures that even if an attacker steals an encrypted database, the database does not leak the values of sensitive fields, even if the attacker has side information.

Mylar, a platform for building web applications, which protects data confidentiality against attackers with full access to servers.

The server’s encrypted database provides semantic security, strong security guarantees: it provides an IND-CPA-like security to the database, which reveals nothing beyond sizing information.
CryptDB, on the other hand, manages to emulate fully homomorphic encryption for most of the functions of the SQL databases used in many applications, computing only with encrypted data and adding just 15% to 26% to those applications' computing time.
Claims

emulates fully homomorphic encryption
provable confidentiality
semantic security
the database does not leak the values
of sensitive fields, even if the attacker
has side information
Fallacy #1

Encryption scheme is “secure” does not mean

The system is “secure”
What This Talk Is About

... and build a completely insecure system from it

How to take a plausible encryption scheme
Unsafe at Any Speed

- CryptDB (SOSP 2011)
- Mylar (NSDI 2014)
- Seabed (OSDI 2016)
- Arx
- Many others
- Lots of industry and government interest!!

If you look at an actual commodity DBMS ...

... insecure under ANY real-world attack
Yeah, well, you know...
That's just, like... your opinion, man
Threat Models

"Snapshot"

Persistent passive

Active
Claims Meet Reality

• Secure against active attacks: false
  – Grubbs et al. “Breaking web applications built on top of encrypted data” (CCS 2016)

• Secure against “snapshot” attacks: false
  – Grubbs et al. “Why your encrypted database is not secure” (HotOS 2017)

• Sensitivity analysis helps: false
3.4 Threat model

Threats. Both the application and the database servers can be fully controlled by an adversary: the adversary may obtain all data from the server, cause the server to send arbitrary responses to web browsers, etc. This model subsumes a wide range of real-world security problems, from bugs in server software to insider attacks.

Mylar also allows some user machines to be controlled by the adversary, and to collude with the server. This may be either because the adversary is a user of the application, or because the adversary broke into a user’s machine.

We call this adversary active, in contrast to a passive adversary that eavesdrops on all information at the server, but does not make any changes, so that the server responds to all client requests as if it were not compromised.
Mylar

Insecure proxy re-encryption scheme
[see Van Rompay et al. 2017]

Server, you can convert all my searches to blue key. Here’s a token to do it.

I trust blue user

Add orange user
Mylar Under Active Attack

Hiring plan for 2017

My secret diary

Search(w)
some user machines ... collude with the server... because the adversary broke into a user’s machine

Myfair also allows some user machines to be controlled by the adversary, and to collude with the server. This may be either because the adversary is a user of the application, or because the adversary broke into a user’s machine.

We call this adversary active, in contrast to a passive adversary that eavesdrops on all information at the server, but does not make any changes, so that the server responds to all client requests as if it were not compromised.
Mylar Under Active Attack

My secret diary

Hiring plan for 2017

Unkeyed “hash” of keyword. Perform dictionary attack.
Guarantees. Mylar protects a data item’s confidentiality in the face of arbitrary server compromises, as long as none of the users with access to that data item use a compromised machine.

... as long as none of the users with access to that data item use a compromised machine
Mylar Under Active Attack

None of the users with access to this data item use a compromised machine.
Mylar in a Hospital

One nurse loses their laptop, server can compromise every doctor’s private files
"Snapshot" Threat Model

Existing systems explicitly claim security
... assuming there are no queries in the snapshot

False in any realistic snapshot attack on a commodity DBMS
A Simple System Abstraction

OS

DB

Volatile memory

Persistent storage
Actual Attacks

Full-system compromise
VM snapshot leak
SQL injection
Disk theft

OS
DB

Volatile memory
Persistent storage
## Case Study: MySQL

similar issues in any other commodity DBMS

<table>
<thead>
<tr>
<th>Attack</th>
<th>What MySQL leaks</th>
<th>Failed encrypted database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk theft</td>
<td>MVCC data structures</td>
<td>Arx’s range query index</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Past query statistics</td>
<td>Seabed’s SPLASHE scheme</td>
</tr>
<tr>
<td>Full system compromise or VM snapshot leak</td>
<td>Text of past queries</td>
<td>CryptDB, Lewi/Wu, etc.</td>
</tr>
</tbody>
</table>
Disk Theft

If this is your threat model, just use full-disk encryption.
Logs on Disk

General query log (not widely used)

Binary log records modifications, used for replication and recovery

Multi-version concurrency control using log data structures

Data modification queries can be reconstructed from these logs

[FHMW '10, FKSHW '12]

In all modern SQL databases!
Range queries via chained garbled circuits
Tree nodes become consumed, need replacing

I used up these nodes
Here, refresh nodes with these ciphertexts

\[ E_k(3) \quad E_k(2) \quad E_k(5) \]

Poddar et al.
Security Claim for Arx

“Arx protects the database with the same level of security as regular AES-based encryption”

Poddar et al.
Arx Under Snapshot Attack

Range queries via chained garbled circuits
Tree nodes become consumed, need replacing
Consumed nodes immediately replaced, stored in MVCC log

Query access pattern recorded on disk
Snapshot attacker can recover queries and plaintexts using variants of attacks from [GSBNR - S&P '17]
## SQL Injection

<table>
<thead>
<tr>
<th>Attack</th>
<th>What MySQL leaks</th>
<th>Failed encrypted database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk theft</td>
<td>MVCC data structures</td>
<td>Arx’s range query index</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Past query statistics</td>
<td>Seabed’s SPLASHE scheme</td>
</tr>
<tr>
<td>Full system compromise or VM snapshot leak</td>
<td>Text of past queries</td>
<td>CryptDB, Lewi/Wu, etc.</td>
</tr>
</tbody>
</table>
SQL Injection

SQL injection accounted for 51% of all Web application attacks in 2016 (source: Akamai)
Diagnostic Tables

information_schema stores current query for all users, contents of buffer cache

performance_schema stores current query for all threads, statistics for past queries

Separate counts for queries which involve different columns

- Inserts: 2
- Selects: 1
Order-preserving encryption reveals histogram of plaintext values

This is how Naveed et al. used frequency analysis to break CryptDB: match histogram to auxiliary model of data distribution
Each possible plaintext gets its own column
WHERE clause transformed to correct column

SELECT Count("Has …") WHERE "Has …" = 1
SELECT Count(C2)

Separate counts for queries which involve different columns

Papadimitriou et al. (OSDI 2016)
## Example

<table>
<thead>
<tr>
<th>country</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>100000</td>
</tr>
<tr>
<td>USA</td>
<td>100000</td>
</tr>
<tr>
<td>Canada</td>
<td>200000</td>
</tr>
<tr>
<td>USA</td>
<td>300000</td>
</tr>
<tr>
<td>Canada</td>
<td>500000</td>
</tr>
<tr>
<td>Canada</td>
<td>800000</td>
</tr>
<tr>
<td>India</td>
<td>100000</td>
</tr>
<tr>
<td>India</td>
<td>100000</td>
</tr>
<tr>
<td>Chile</td>
<td>200000</td>
</tr>
<tr>
<td>Iraq</td>
<td>300000</td>
</tr>
<tr>
<td>China</td>
<td>500000</td>
</tr>
<tr>
<td>Japan</td>
<td>800000</td>
</tr>
<tr>
<td>Israel</td>
<td>130000</td>
</tr>
<tr>
<td>U.K.</td>
<td>210000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>country</th>
<th>salaryUSA</th>
<th>salaryCanada</th>
<th>salaryOthers</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET(Chile)</td>
<td>ASHE(100000)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(Iraq)</td>
<td>ASHE(100000)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(China)</td>
<td>ASHE(0)</td>
<td>ASHE(200000)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(Japan)</td>
<td>ASHE(300000)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(Israel)</td>
<td>ASHE(0)</td>
<td>ASHE(500000)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(U.K.)</td>
<td>ASHE(0)</td>
<td>ASHE(800000)</td>
<td>ASHE(0)</td>
</tr>
<tr>
<td>DET(India)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(100000)</td>
</tr>
<tr>
<td>DET(India)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(100000)</td>
</tr>
<tr>
<td>DET(Chile)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(200000)</td>
</tr>
<tr>
<td>DET(Iraq)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(300000)</td>
</tr>
<tr>
<td>DET(China)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(500000)</td>
</tr>
<tr>
<td>DET(Japan)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(800000)</td>
</tr>
<tr>
<td>DET(Israel)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(130000)</td>
</tr>
<tr>
<td>DET(U.K)</td>
<td>ASHE(0)</td>
<td>ASHE(0)</td>
<td>ASHE(210000)</td>
</tr>
</tbody>
</table>
SQLi Extracts Diagnostic Tables

SELECT Count(C2)
SELECT Count(C3)

performance_schema:
Selects for C2: 1
Selects for C3: 2

Separate counts for queries which involve different columns

Use frequency analysis to recover plaintexts (see paper for details)
## Full-System Snapshot

<table>
<thead>
<tr>
<th>Attack</th>
<th>What MySQL leaks</th>
<th>Failed encrypted database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk theft</td>
<td>MVCC data structures</td>
<td>Arx’s range query index</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Past query statistics</td>
<td>Seabed’s SPLASHE scheme</td>
</tr>
<tr>
<td><strong>Full system compromise or VM snapshot leak</strong></td>
<td><strong>Text of past queries</strong></td>
<td><strong>CryptDB, Lewi/Wu, etc.</strong></td>
</tr>
</tbody>
</table>
Full-System Compromise

Leakage of sensitive data at OS level is well-studied [CPGR, DLJKSXSW]

We focus on DBMS address space, things inaccessible to users
Data Structures and Caches

Adaptive hash index tracks pages accesses, indexes automatically
MySQL query cache stores select queries and results
Other query caches (memcached)

MySQL manages internal heaps, does not zero freed memory!

essential for performance!
Token-Based Systems

CryptDB, Mylar, Lewi-Wu, other searchable encryption schemes cannot be semantically secure if attacker sees a single search token.

Select

Search token

1,000 random selects…

Waited a while…

100,000 more random selects…
Let Me Make Myself Perfectly Clear

These encrypted databases CANNOT be semantically secure under ANY real-world attack.
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
There is no such thing as a snapshot attack
“I Will Build My Own Database”

You can try…

Transaction logs needed to support ACID
Log-structured storage
Caching
Adaptive data structures adjust to workload

… everything in modern databases leaks information about past queries
### Sensitivity Analysis

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Ethnicity</th>
<th>Date Of Birth</th>
<th>Sex</th>
<th>ZIP</th>
<th>Marital Status</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>black</td>
<td>09/27/64</td>
<td>male</td>
<td>02139</td>
<td>divorced</td>
<td>obesity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>black</td>
<td>09/30/64</td>
<td>male</td>
<td>02139</td>
<td>divorced</td>
<td>hypertension</td>
</tr>
<tr>
<td></td>
<td></td>
<td>black</td>
<td>04/18/64</td>
<td>male</td>
<td>02139</td>
<td>married</td>
<td>chest pain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>black</td>
<td>04/15/64</td>
<td>male</td>
<td>02139</td>
<td>married</td>
<td>chest pain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>black</td>
<td>09/15/64</td>
<td>male</td>
<td>02138</td>
<td>married</td>
<td>shortness of breath</td>
</tr>
<tr>
<td></td>
<td></td>
<td>caucasian</td>
<td>03/13/63</td>
<td>male</td>
<td>02141</td>
<td>married</td>
<td>hypertension</td>
</tr>
<tr>
<td></td>
<td></td>
<td>caucasian</td>
<td>03/18/63</td>
<td>male</td>
<td>02141</td>
<td>married</td>
<td>shortness of breath</td>
</tr>
<tr>
<td></td>
<td></td>
<td>caucasian</td>
<td>09/13/64</td>
<td>female</td>
<td>02138</td>
<td>married</td>
<td>obesity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>caucasian</td>
<td>09/07/64</td>
<td>female</td>
<td>02138</td>
<td>single</td>
<td>chest pain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>caucasian</td>
<td>05/14/61</td>
<td>female</td>
<td>02138</td>
<td>single</td>
<td>obesity</td>
</tr>
</tbody>
</table>

- **Order-Preserving Encryption**: can sort
- **Deterministic Encryption**: can check for equality
- **“Strong” Encryption**
Auxiliary Data

Public auxiliary data
(e.g., previous release of similar datasets)

Distribution of values in each column

Correlations between columns
Bayesian Inference

- Distribution of ciphertexts in each column
- Distribution of values in each column

Public auxiliary data (e.g., previous release of similar dataset)
Multinomial Attack

Observed ciphertexts

Plaintext distribution (from auxiliary data)

$$\Pr\{f = f \mid \tilde{c} ; \rho\} = \frac{\Pr\{\tilde{c} \mid f = f ; \rho\} \cdot \Pr\{f = f ; \rho\}}{\Pr\{\tilde{c} ; \rho\}}$$

$$f_{\text{max}} = \arg \max_f \Pr\{f = f \mid \tilde{c} ; \rho\}$$

$$= \arg \max_f \Pr\{\tilde{c} \mid f = f ; \rho\} \cdot \Pr\{f = f ; \rho\}$$

Most likely mapping of ciphertexts to plaintexts

Density of multinomial distribution

$$\Pr\{\tilde{c} \mid f = f ; \rho\} = \Pr\{c_1, c_2, \ldots, c_n \mid f = f ; \rho\} = K_c \prod_{i=1}^{m} \rho_i^{c_{f(i)}}$$
Multinomial Attack

- **Optimal**
  - Maximum likelihood estimator for deterministic ORE
- **Outperforms previous heuristics**
  - Naveed et al. frequency analysis (CCS 2015)
  - Grubbs et al. non-crossing attacks (Oakland 2017)
- **Extends to multiple columns**
  - Condition distribution on previously recovered plaintexts for a dependent column
Inferring “Sensitive” Columns

Features

Prediction

Distribution of values in each column

Correlations between columns

Public auxiliary data (e.g., previous release of similar dataset)

Multinominal attack!

Ind-CPA

OPE

DET

IND-CPA

Bindschaedler et al.
Let’s Try with Real Data

- Over 7 million hospital discharge records each year
- Demographic + medical attributes

- Over 3 million records each year
- Demographic attributes, income

- Data dump from 2015 hack
- Names and addresses of over 600,000 police officers
Empirical Results

• HCUP-NIS hospital discharge records
  – Infer if patient has a mental health or substance abuse condition with 97% accuracy
  – … mood disorder with 96% accuracy

• U.S. Census American Community survey
  – Recover 90% of PRE-encrypted attributes
  – Infer income to within $8.4K

• Fraternal Order of Police (FOP) data dump
  – Exact home addresses of 5,500 police officers in PA
Remember

Encryption scheme is “secure”

does not mean

The system is “secure”
Advice to Practitioners

I WANT TO DEPLOY AN "ENCRYPTED DATABASE"

DON'T