CryptDB: Protecting Confidentiality with Encrypted Query Processing

Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan

Problem

- Need to protect *private* information for *online* applications
  - From malicious/curious admins
  - Attackers with physical access to servers
- Systems to support *existing applications* with *minimal overhead* but still providing *confidentiality*

Existing approaches?

- **Approach#1**: Encrypt all data, run all computations on clients.
  - *Pro:* Confidentiality when servers compromised
  - *Con:* Not applicable to large number of applications
    - due to scalability limitations, or infeasibility in changing existing server-side applications
- **Approach 2:** Fully homomorphic encryption.
  - *Pro:* Confidentiality when servers compromised; no changes to existing applications.
  - *Con:* Computations on encrypted data are extremely slow.

What does CryptDB do?

- An intermediate design point specifically for applications that use DBMS.

- **Approach:** Execute queries directly on encrypted data

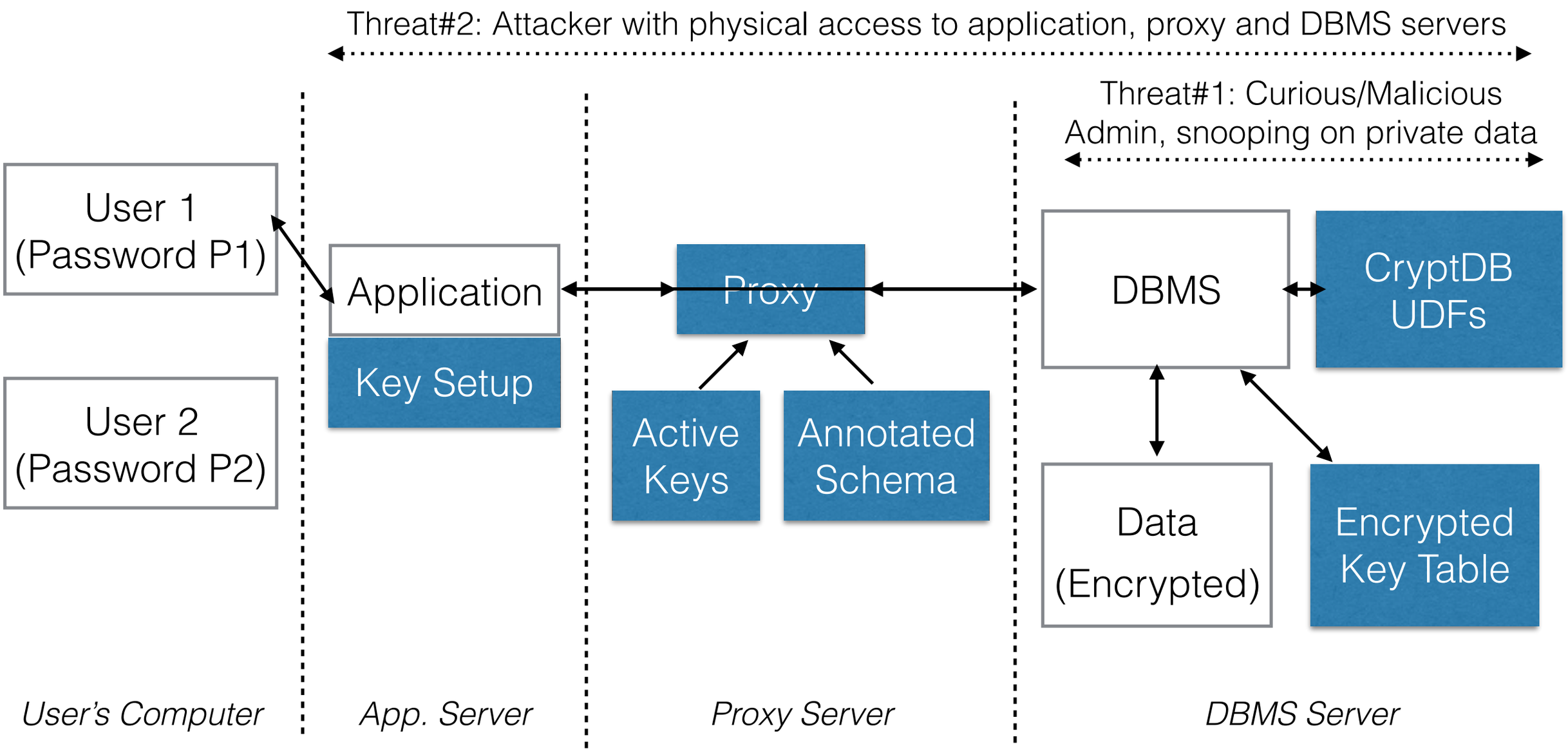
- How is this from homomorphic encryption?

  - SQL exposes a *limited* set of operators

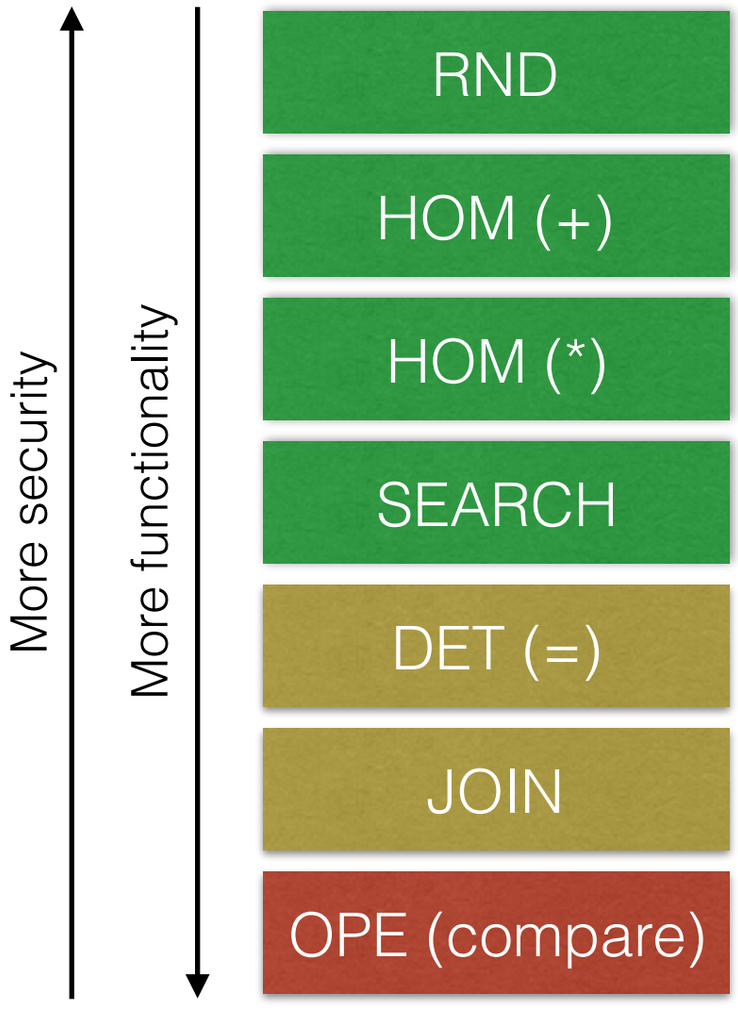
  - Support these operators *efficiently* on encrypted data

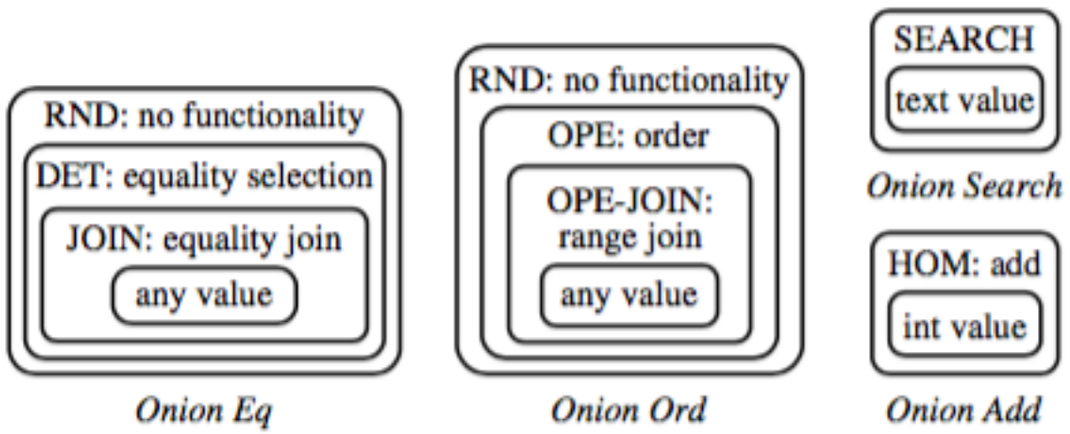
  - As opposed to supporting *arbitrary* computations

Architecture & Threat Model



Queries on Encrypted Data

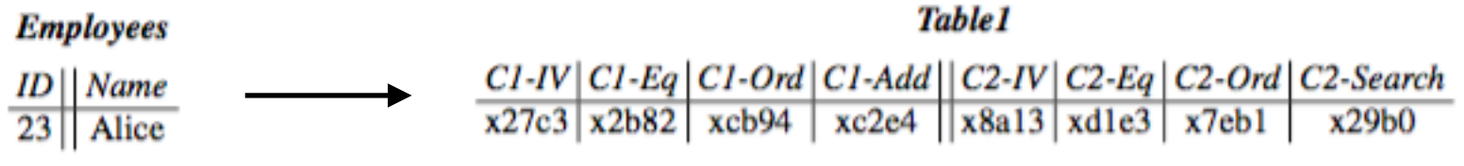




Entire column encrypted with the same key for each onion layer

Multiple onions per column to begin with (not necessarily all)

Onion layers “peeled off” (decrypted) as queries on columns require.



Employees

ID	*Name*
23	Alice

Table1

C1-IV	*C1-Eq*	*C1-Ord*	*C1-Add*	*C2-IV*	*C2-Eq*	*C2-Ord*	*C2-Search*
x27c3	x2b82	xcb94	xc2e4	x8a13	xd1e3	x7eb1	x29b0

Trade-offs?

- Minimizing amount of confidential data exposed to DBMS vs. efficient execution of queries
- CryptDB sacrifices “optimal” security (i.e., homomorphic encryption) for practicality (i.e., realistic query execution times)
    - For most real-world applications, most sensitive fields remain encrypted with highly secure encryption schemes.

Multiple Principals

- Similar to the concept of access control modifiers in DBMS
- Different “principals” (e.g., users, groups, messages) have have access to different data
    - Each principal is assigned its own encryption key
- Keys are *chained* to user passwords
    - Each data item in the database decrypted through a chain of keys rooted at user password
- *Guarantee:* CryptDB leaks at most the data of active users for the duration of the compromise

Multiple Principals

Table *privmsgs*

msgid	*subject*	*msgtext*
5	xcc82fa	x37a21f

Table *privmsgs_to*

msgid	*rcpt_id*	*sender_id*
5	1	2

Table *users*

userid	*username*
1	'Alice'
2	'Bob'

Principal Type	Principal (Key)
msgid	msg (Km)
sender_id	user (Ku)
rcpt_id	user (Ku)
user_id	user (Ku)
username	physical_user (Kp)

Key Chaining: Km ⟶ Ku ⟶ Kp

Principal	Stored Key
msg	E(Km, Ku)
user	E(Ku, Kp)

Evaluation

- **Applicability:** supports operations over encrypted data for 99.5% of 128,840 columns seen in a large trace (~126 million SQL queries)

- **Low overhead:** reduces throughput by 14.5% for phpBB, and by 26% for TPC-C, compared to unmodified MySQL

- **Minimal changes:** requires 11–13 unique schema annotations to secure more than 20 sensitive fields and 2–7 lines of source code changes for three multi-user web applications.

Impact

Adoption

Google	Google recently deployed a system for performing SQL-like queries over an encrypted database following (and giving credit to) the CryptDB design. Their service uses the encryption building blocks from CryptDB (RND, DET, HOM, and SEARCH), rewrites queries and annotates the schema as in CryptDB.
	Lincoln Labs added the CryptDB design on top of their D4M Accumulo no-SQL engine (using the RND, DET, OPE and HOM building blocks).
sql.mit.edu	sql.mit.edu is a SQL server at MIT hosting many MIT-ran applications. Volunteering users of Wordpress switched to running Wordpress through CryptDB.
Other companies using CryptDB's design	SAP AG and a new startup are applying CryptDB's design to their setting.

Press Coverage

Forbes

Slashdot

ZDNet

GIGAOM

HN

Recently…

MS researchers claim to crack encrypted database with old simple trick

CryptDB developer says researchers were using database in way no one else would.

by **Sean Gallagher** - Sep 4, 2015 9:40am PDT

Share Tweet

- Naveed et al. analyzed CryptDB's DET and OPE schemes on medical data from National In-patient Sample (NIS) database
  - Used Frequency analysis, along with three new attacks (Lp-optimization, sorting attack, cumulative attack)
  - Were able to recover mortality risk, patient death attributes and disease severity data for almost all patients.
- In response, a report titled "Guidelines for Using the CryptDB System Securely" claims that evaluation in Naveed et al. makes incorrect use of CryptDB by not marking fields as "sensitive" when they need to be.
  - The DET (equality) and OPE (order preserving) schemes can be avoided for most queries
  - e.g., execute equality on string columns using SEARCH encryption and ORDER BY queries without LIMIT at the proxy server

Discussion

- Are computations on encrypted data for SQL sufficiently expressive?
  - Currently, CryptDB does not support complex operations (arithmetic involving multiple columns, complex functions, UDFs)
- What happens when the most sensitive fields are the most queried ones?
- How does CryptDB scale the Proxy server? Is it a scalability bottleneck?