TDDD17
Information Security
(VT 2019)

Topic: Database Encryption

Olaf Hartig

olaf.hartig@liu.se

LINKÖPING UNIVERSITY

Limitations of Access Control

… as a means to achieve the objectives of DB security (in particular, confidentiality and integrity)

- Authorizations enforced by DBMS may be bypassed
  - Intruder can try to mine the database footprint on disk
  - DB administrator has enough privileges to tamper the access control definitions and gain access
- Management of databases outsourced
  - "Database as a service" / cloud services
  - No other choice than trusting the service provider

Purpose of Database Encryption

- Complement and reinforce access control by resorting to cryptographic techniques
- Ensure confidentiality of DBs by keeping data hidden from unauthorized persons

Relevant Factors for Database Encryption

- Where should the encryption be performed?
  …in the storage layer? …in the database?
  …in the application that produces the data?
- How much data should be encrypted and exactly which?
- What encryption algorithm and mode of operation?
- Who should have access to the encryption keys?
- How to minimize the impact on performance?

Data Structures for Databases

A brief reminder before we continue ...

Database Files

- File is a sequence of records
  - Record is a set of fields that contain values
  - For instance,

    File = relation / table

    Record = tuple / row

    Field = attribute value / cell

Data File

ID#	SSN	Dept.	Salary
1	4945864	12	2000
2	7000111	13	4000
3	…		
4	…		

Database Files

- File is a sequence of records
  - Record is a set of fields that contain values
  - For instance,

    File = relation / table
    Record = tuple / row
    Field = attribute value / cell

- Files may consist of multiple blocks
  - Block is the unit of data transfer between disk and main memory
  - Each record is allocated to a block

- There exists different approaches to organize records in a file
  - e.g., heap files, sorted files

Data File

ID#	SSN	Dept.	Salary	
1	4945864	12	2000	Block 1
2	7000111	13	4000	
3	…			
4	…			
5	6487539			Block 2
6	7299990			
7	3452626			
8	9000013			
9	8232333			Block 3
10	…			
11	5012128			
12	…			

Indexes

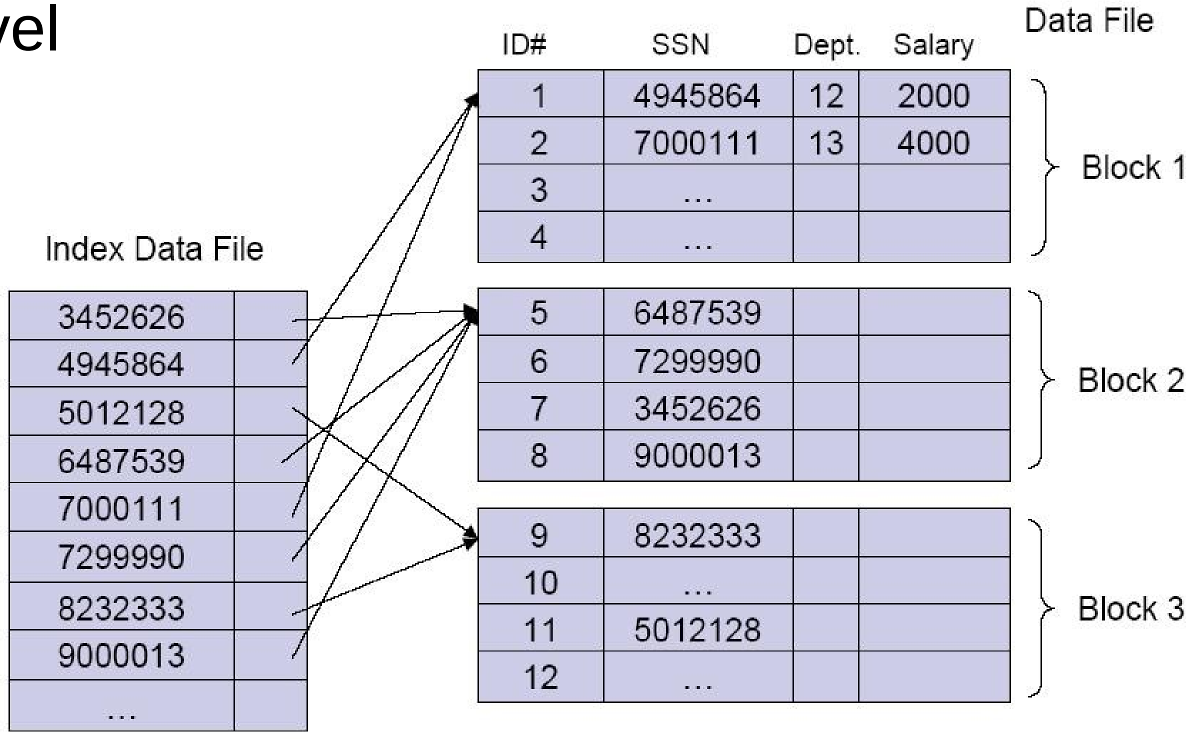
- Organization of data file determines primary method to access data (e.g., sequential scan, binary search)
- Indexes are additional files for secondary access methods
  - Goal: speed up access under specific conditions

Data File

ID#	SSN	Dept.	Salary	
1	4945864	12	2000	Block 1
2	7000111	13	4000	
3	…			
4	…			
5	6487539			Block 2
6	7299990			
7	3452626			
8	9000013			
9	8232333			Block 3
10	…			
11	5012128			
12	…			

Indexes

- Organization of data file determines primary method to access data (e.g., sequential scan, binary search)
- Indexes are additional files for secondary access methods
  - Goal: speed up access under specific conditions
- Example of a single-level secondary index on a non-ordering key field:



Encryption Granularity

How much data should be encrypted and exactly which?

Encryption Granularity

- Common levels of encryption granularity:
  - field
  - record
  - file
  - whole database
- Finer granularity has advantages:
  - allows for encryption of only the sensitive data
  - only relevant data need to be decrypted for query execution
  - different encryption keys may be used for different parts
- However, finer granularity is not always possible (see later)
- Note: sensitive data may not only be in the data file,
  but also in temporary files, log files, indexes, etc.

Encryption Layer

Where should the encryption be performed?

Storage-Level Encryption

- Use the storage subsystem to encrypt database files
  - i.e., file pages are encrypted/decrypted by the OS when written/read from disk
- Advantages:
  - Transparent from the DB perspective, i.e., no changes to the DBMS or the applications necessary
- Disadvantages:
  - Limited to file granularity
  - Cannot be related with user privileges or data sensitivity (because storage subsystem has no knowledge of DB objects or structure)

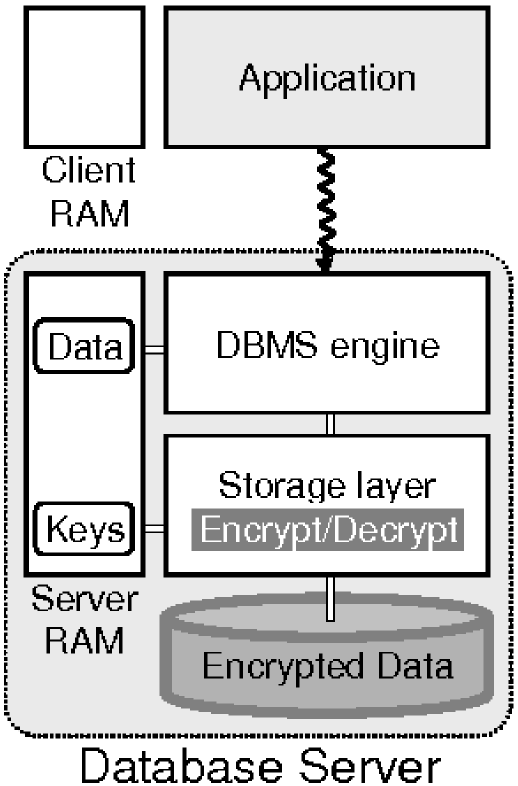


Figure from “Database Encryption” by Bouganim and Guo (2009).

Database-Level Encryption

- DBMS encrypts data when it is inserted into the database
- Advantage: Encryption strategy can be part of the database design (i.e., selective encryption possible, various granularities possible)
- Disadvantage: Performance degradation possible (e.g., encryption may make indexes useless)

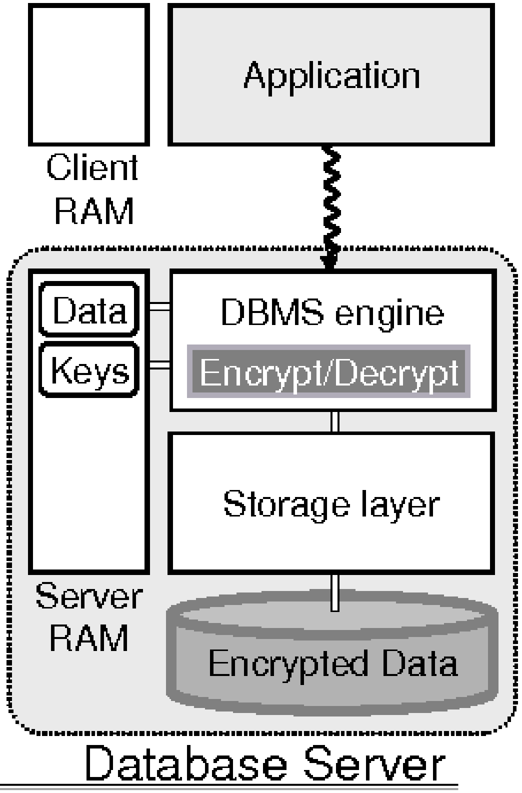


Figure from "Database Encryption" by Bouganim and Guo (2009).

Application-Level Encryption

- Application encrypts sensitive data before sending it to the DBS and decrypts data returned by the DBS
- Advantages:
  - Encryption keys separated from the encrypted data (i.e., no need to trust the DB administrator)
  - Highest flexibility in terms of granularity and key management
- Disadvantages:
  - Applications need to be modified
  - Performance overhead possible (e.g., prevents indexes for range queries)
  - No stored procedures and triggers

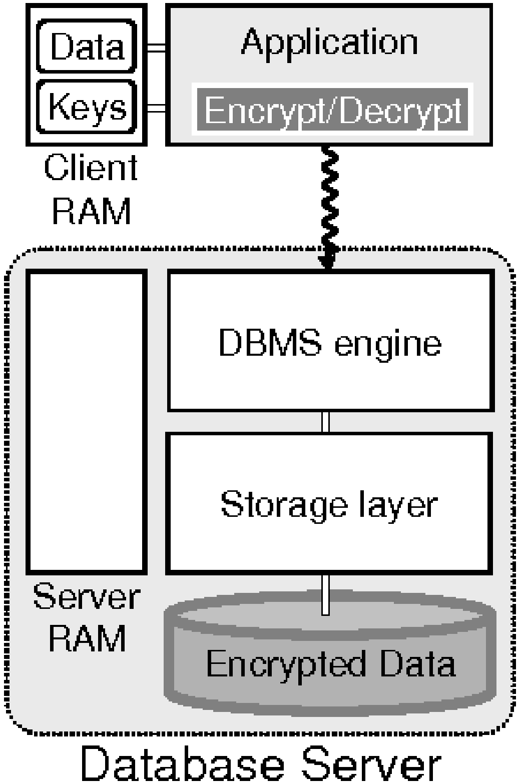


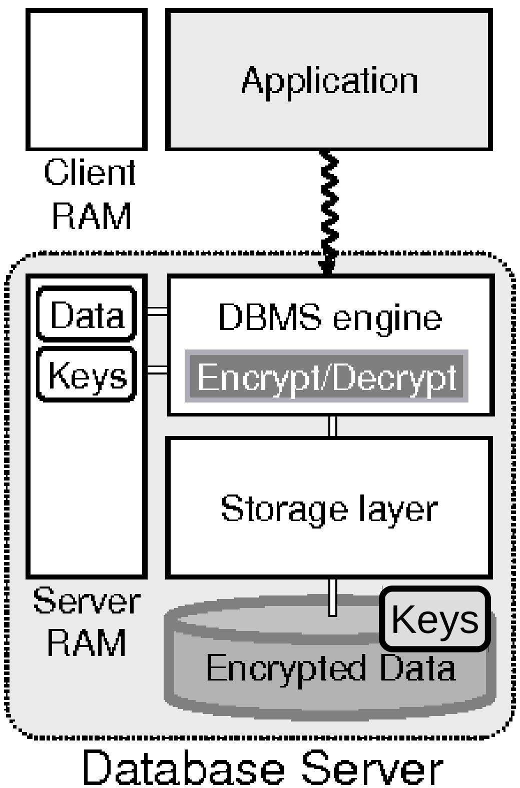
Figure from “Database Encryption” by Bouganim and Guo (2009).

Key Management

Who should have access
to the encryption keys?

Naive Solution (for DB-Level Encryption)

- Store keys in a restricted database table or file
- Potentially encrypt this table/file with a master key
  - Master key must also be stored on the database server

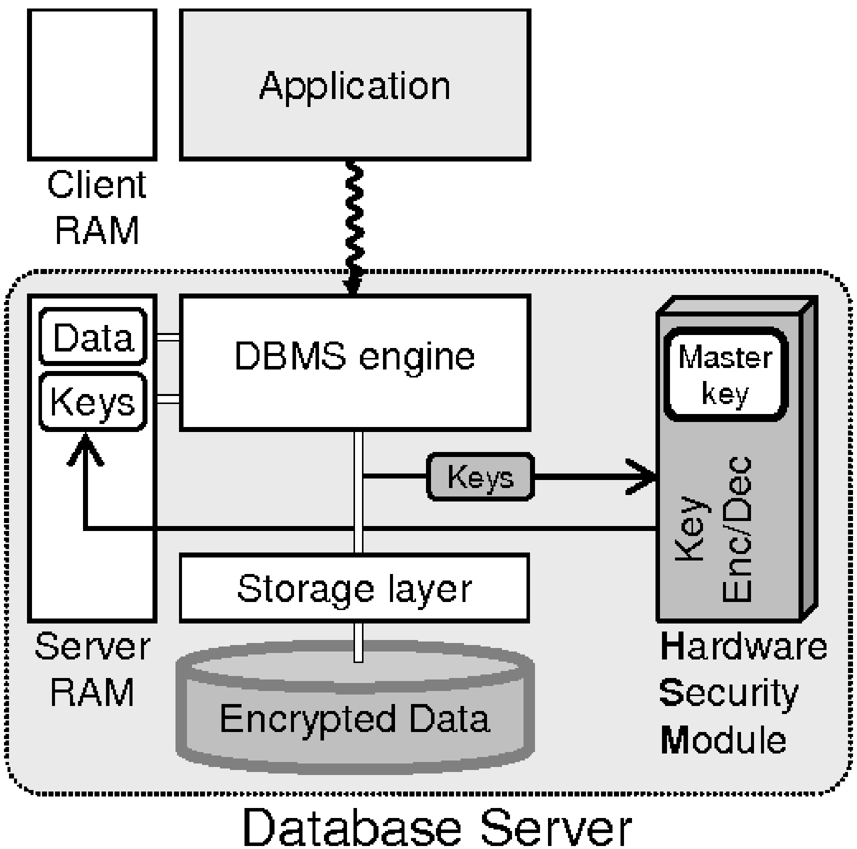


- Disadvantage:
  - Administrators with privileged access may use the keys to see and/or modify the data without being detected

Figure from "Database Encryption" by Bouganim and Guo (2009).

HSM Approach

- Use a hardware security module (HSM)
  - Specialized, tamper-resistant cryptographic chipsets



- Keys are stored encrypted in a restricted database table
- To encrypt/decrypt data the needed keys are decrypted by the HSM using the master key
- Decrypted keys are removed from main memory as soon as encryption/decryption of data has been performed

Figure from "Database Encryption" by Bouganim and Guo (2009).

Security Server Approach

- Move security-related tasks to distinct software on a distinct server that manages users, roles, privileges, encryption policies, and keys (potentially using an HSM)
  - Security module within the DBMS communicates with the security server
    - Clear distinction between DB administrator and security administrator

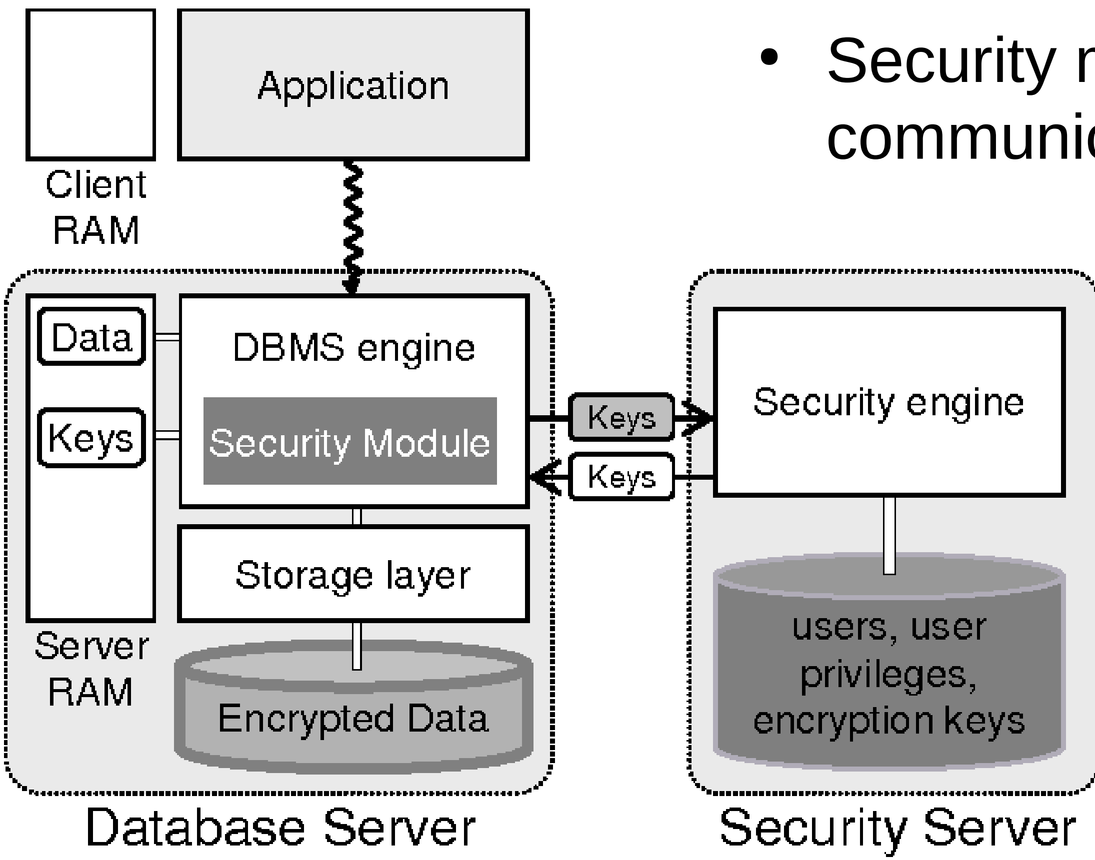


Figure from “Database Encryption” by Bouganim and Guo (2009).

An Example Approach: CryptDB

R.A.Popa et al. “CryptDB: Protecting Confidentiality with Encrypted Query Processing.” In Proc. of the 23rd ACM Symposium on Operating Systems Principles (SOSP), 2011.

R.A.Popa et al. “CryptDB: Processing Queries on an Encrypted Database.” Communications of the ACM 55(9) 2012.

Main Properties of CryptDB

- Executes a wide range of SQL queries over encrypted data
- Provides confidentiality even if an attacker has full read access to the data stored on the database server
  - DBMS sees only anonymized schema, encrypted data, and some auxiliary tables used by CryptDB
- Requires no changes to the DBMS nor to the applications
- Trusted proxy provides an encryption layer
  - between database-level and application-level encryption

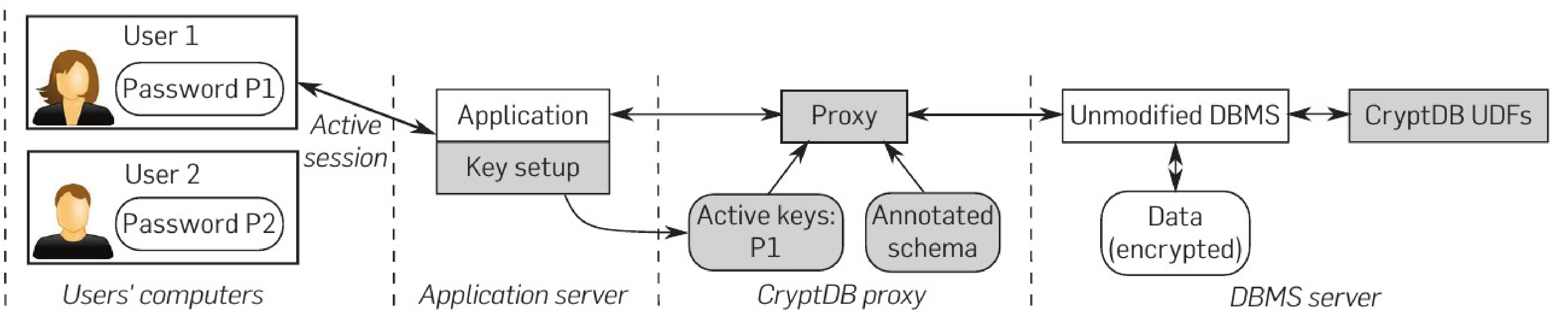


Figure from "CryptDB: Processing Queries on an Encrypted Database" by Ropa et al. (2012).

CryptDB Proxy

- Encrypts and decrypts all data
- Intercepts all SQL queries
- Rewrites queries to execute them on the encrypted data
  - Some operators are replaced by calls to user-defined functions (UDFs) that CryptDB registers in the DBMS

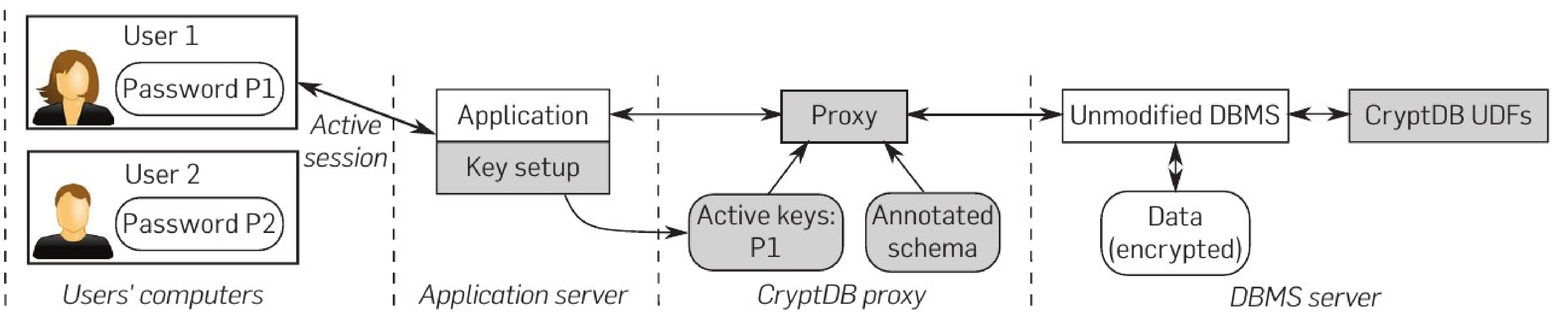


Figure from “CryptDB: Processing Queries on an Encrypted Database” by Ropa et al. (2012).

“Onion Encryption”

- Data values are wrapped in multiple layers of encryption
  - Decreasing in strength, but more operations possible
  - Idea: remove layers if necessary for queries
- Random (RND): Probabilistic scheme in which two equal values mapped to different ciphertexts with high probability
  - Maximum security
  - No query operations can be performed on the ciphertext
  - AES or Blowfish in CBC mode with random init. vector
- Deterministic (DET): Same ciphertexts for the same values
  - Allows for equality checks (incl. GROUP BY, COUNT, DISTINCT)
  - AES or Blowfish in CMC mode
- JOIN: Allows for equality checks between different columns

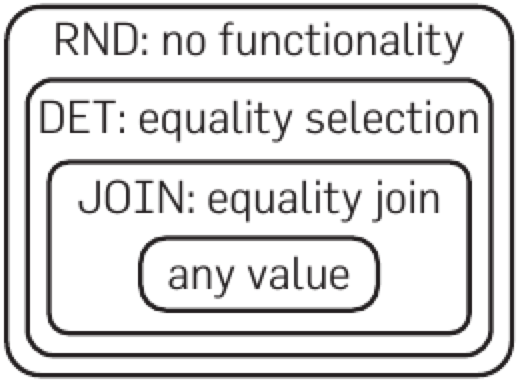
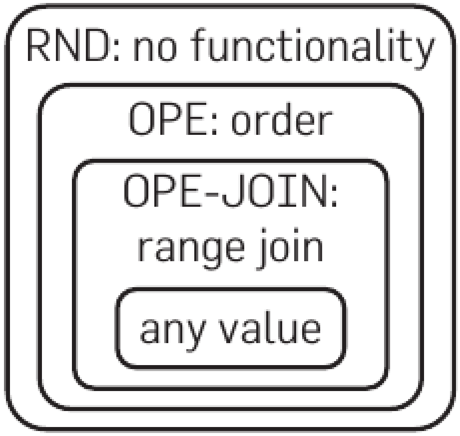


Figure from “CryptDB: Processing Queries on an Encrypted Database” by Ropa et al. (2012).

“Onion Encryption” (cont'd)

- Data values are wrapped in multiple layers of encryption
  - Decreasing in strength, but more operations possible
  - Idea: remove layers if necessary for queries
- Different types of “onions”
- Order-preserving encryption (OPE): For some encr. key K, if $x<y$, then $OPE_K(x)<OPE_K(y)$
  - Allows for range queries over the encrypted data, and also ORDER BY, MIN, MAX
  - Weaker than DET because it reveals order



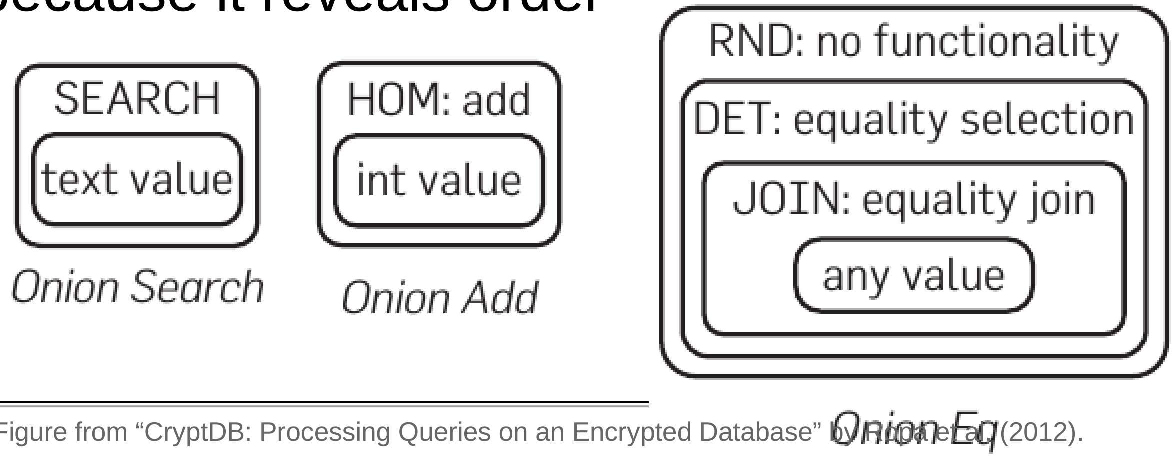


Figure from “CryptDB: Processing Queries on an Encrypted Database” by Popa et al. (2012).

CryptDB Encryption Scheme

- Multiple onion encryptions per column
- Example:

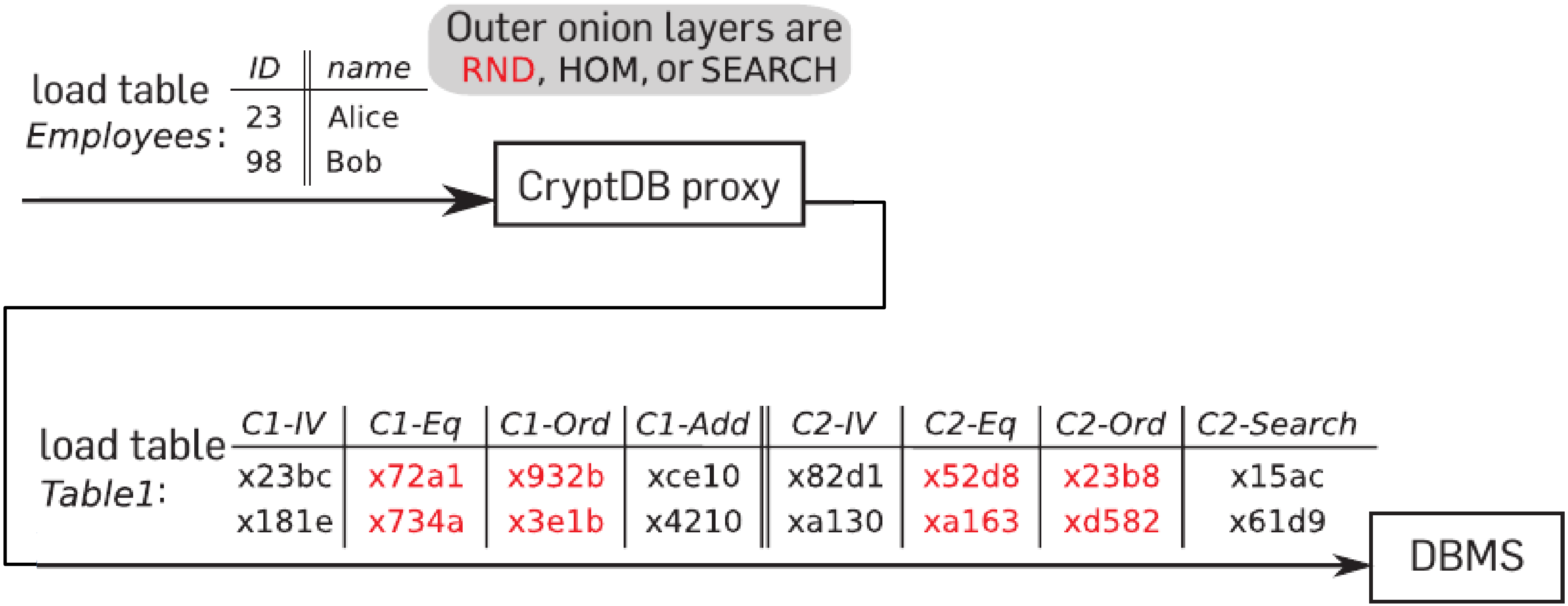


Figure from “CryptDB: Processing Queries on an Encrypted Database” by Ropa et al. (2012).

Query Execution in CryptDB

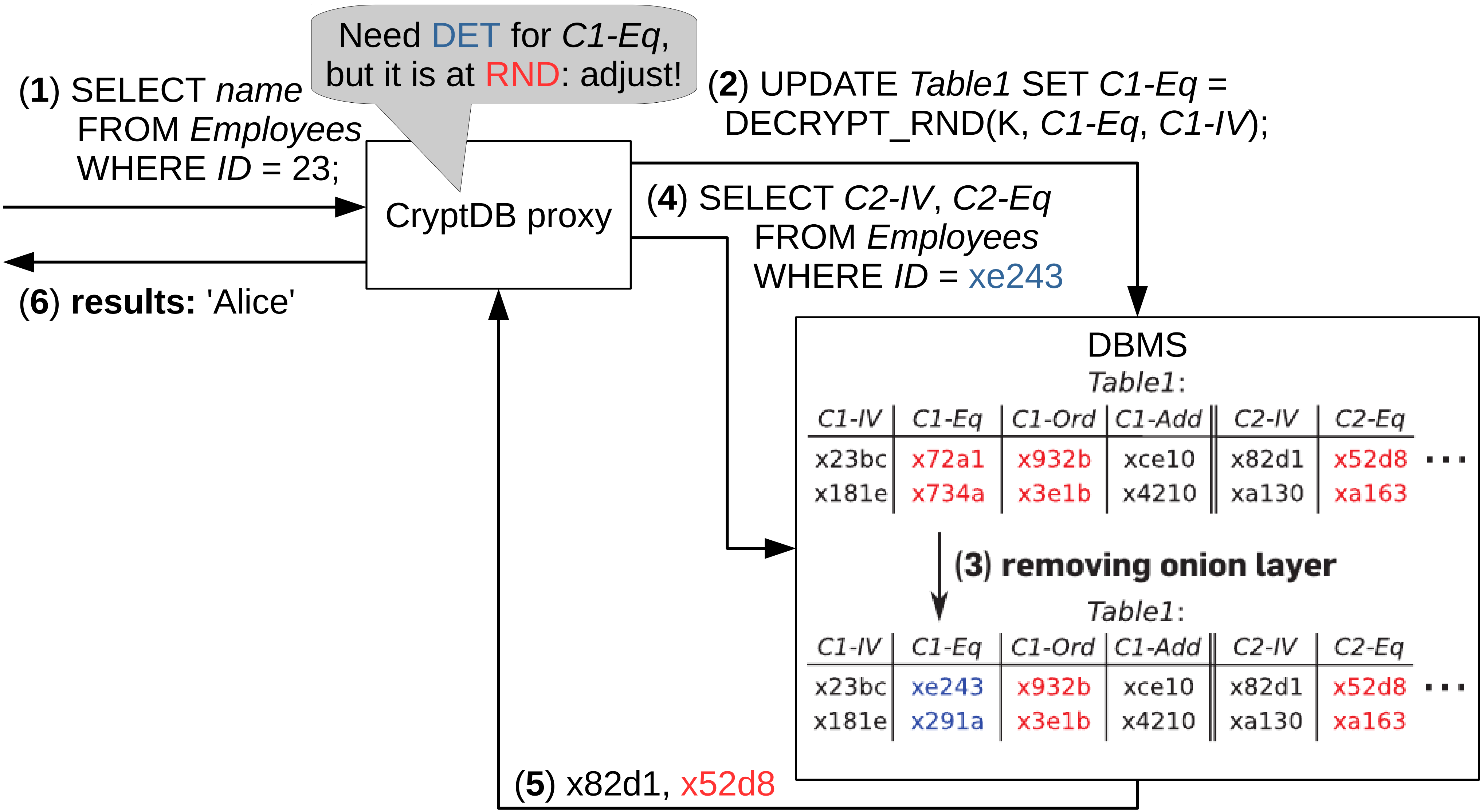


Figure from “CryptDB: Processing Queries on an Encrypted Database” by Ropa et al. (2012).

Summary

Summary

- Database encryption is still an active area of research
- Encryption granularity?
- Encryption layer?
- Key management?

www.liu.se

LINKÖPING UNIVERSITY