Confiabilidade de Sistemas Distribuídos
Dependable Distributed Systems

DI-FCT-UNL, Henrique Domingos, Nuno Preguiça

Lect. 9
DBMS Security
The Case for CryptDB

2015/2016, 2nd SEM

MIEI
Mestrado Integrado em Engenharia Informática

DBMS Security

Part I

- DBMS Security Issues
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II – Other Dimensions

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

Bibliography

- For CryptDB
  - See Raluca Popa, C. Redfield, N. Zeldovich, H. Balakrisnan, CryptDB: protecting confidentiality with encrypted query processing, in Proc. SOSP Symposium on Operating System Principles, 2011
    - Also:
      - http://dl.acm.org/citation.cfm?id=2043566
      - https://css.csail.mit.edu/cryptdb/

- Stallings, Computer Security – Principles and Practice, 3rd Ed., Pearson
  - Chap. 5 **Database** and Cloud Security
    - Database part: pp. 155-180

DBMS Security

Part I

- DBMS
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II – Other Dimensions

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

Databases

- Structured collections of data, stored as possible common *data-backends* for one or more applications
  - Ex., Data-Layer in 3-N Tier Distributed Architectures
- Contains:
  - Data items
  - Relations between data items and groups of data items

- In some cases, databases are used to manage and store sensitive data (in the context of possible critical applications)
  - Data needs to be secured:
    - access control, confidentiality, privacy issues, inference (operations and relations)

Query Languages

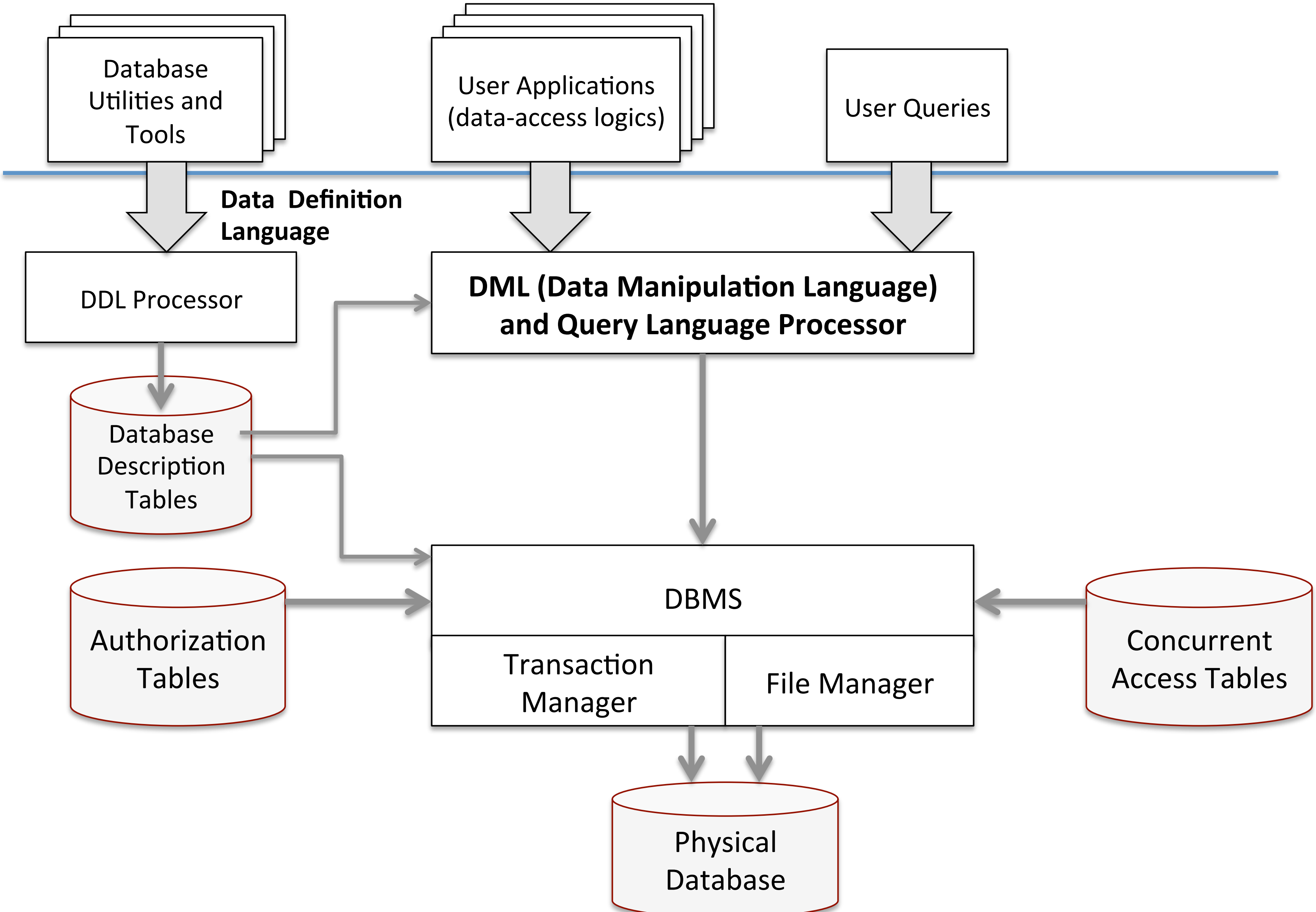
- Languages providing uniform query-interface to the database
  - Standard query languages are also used to access sensitive data and their relations
  - Commonly: SQL operations, SQL Queries

Generic DBMS Architecture

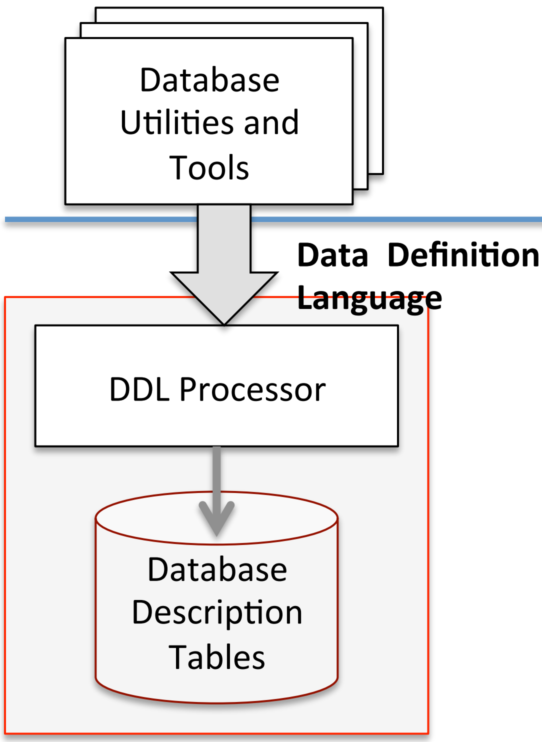
DBMS View (Systemic View)

- DDL - Data Definition Language
- DDL Processor
- DDTs - Data Description Tables
- DML – Data Manipulation Language and QL Processor
- DBMS
- Auth Tables
- Concurrency Control Tables
- Physical Database

DBMS Generic Architecture



DBMS Generic Architecture

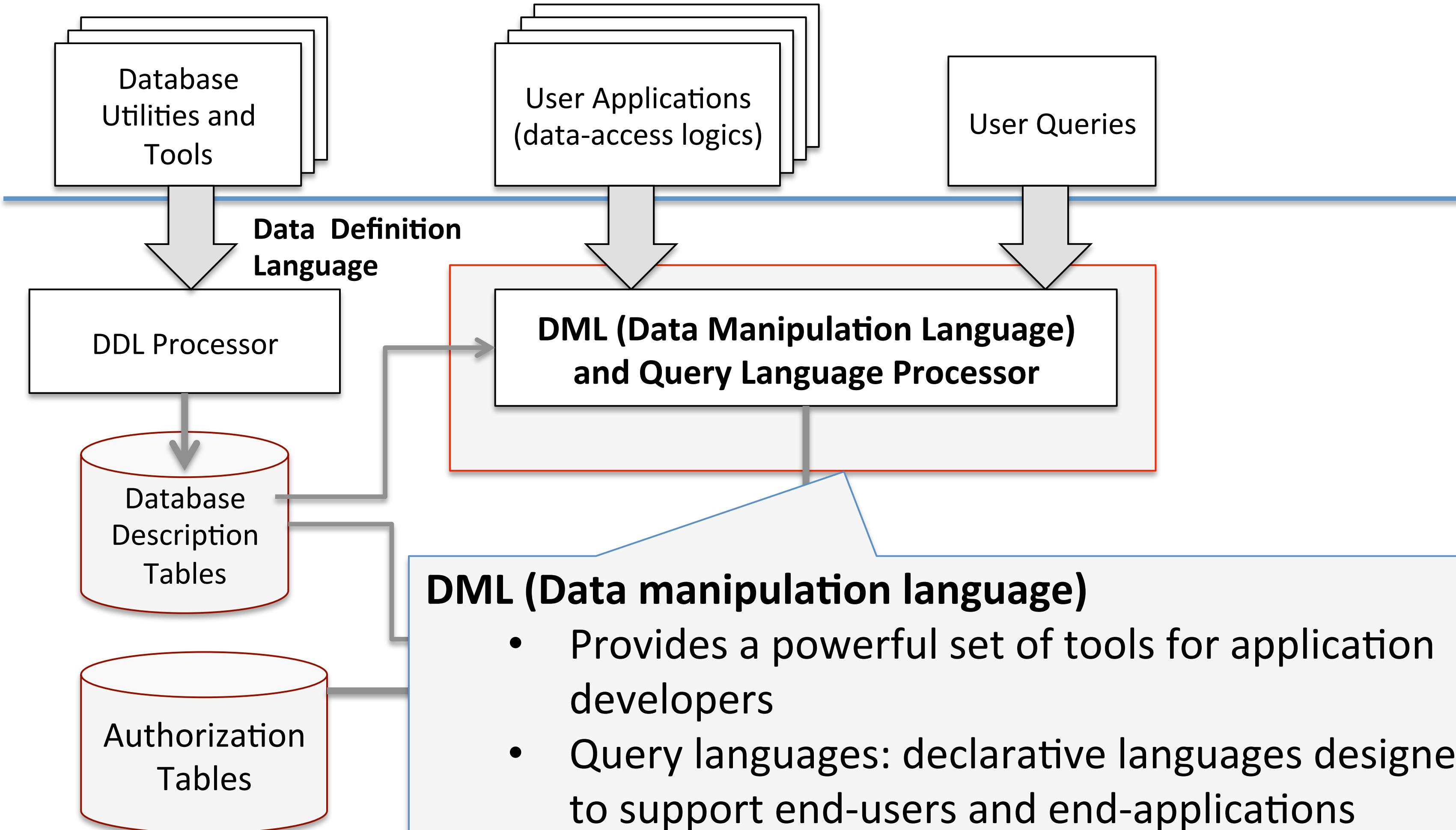


DDL (Data Definition Language)

Database designers and administrators make use of DDL:

- to define the database logical structure
- and procedural properties which are represented by a set of DDTs (Database Description Tables)

DBMS Generic Architecture



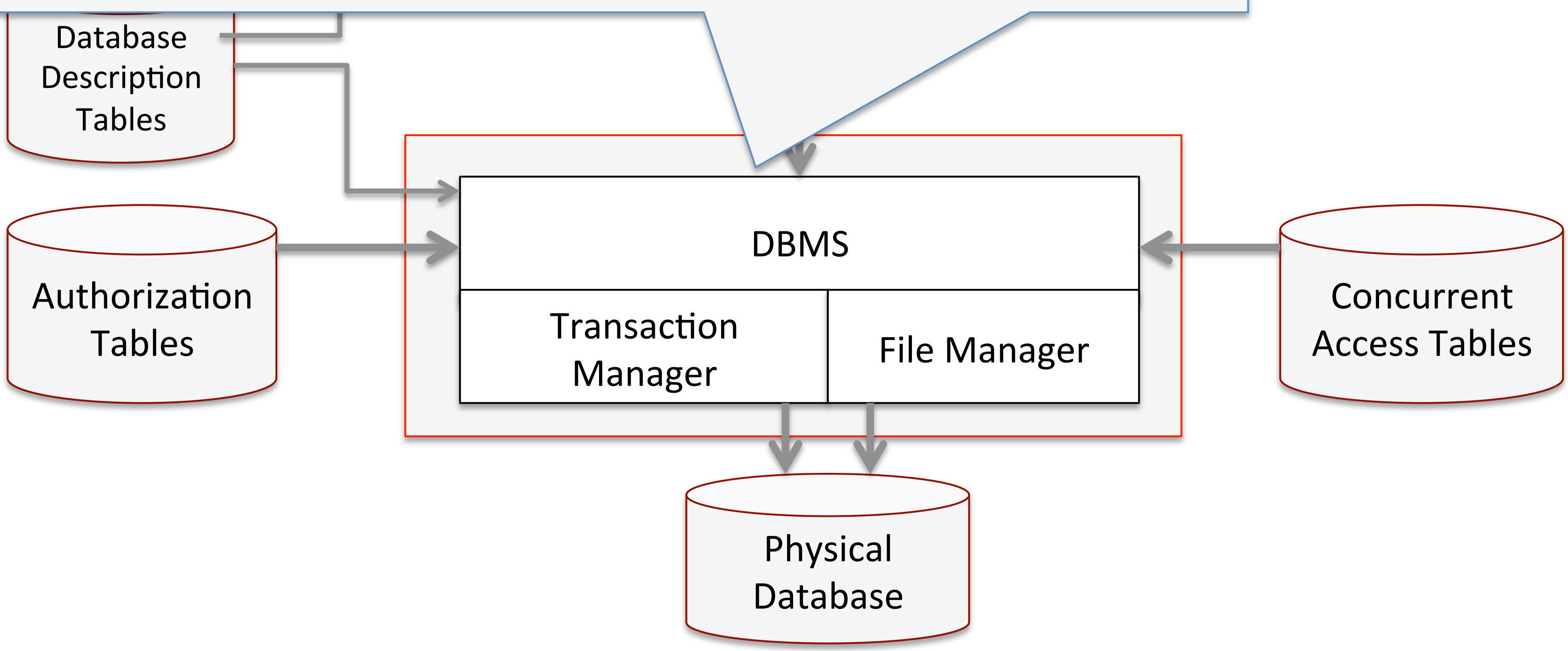
DML (Data manipulation language)

- Provides a powerful set of tools for application developers
- Query languages: declarative languages designed to support end-users and end-applications
- Data manipulations are described by the query language statements
- Ex., SQL Language

DBMS Generic Architecture

Database management system

- Makes use of the database description tables to manage the physical database.
- The interface to the database:
    - through a File Manager Module and a Transaction Manager Module.



DBMS Generic Architecture

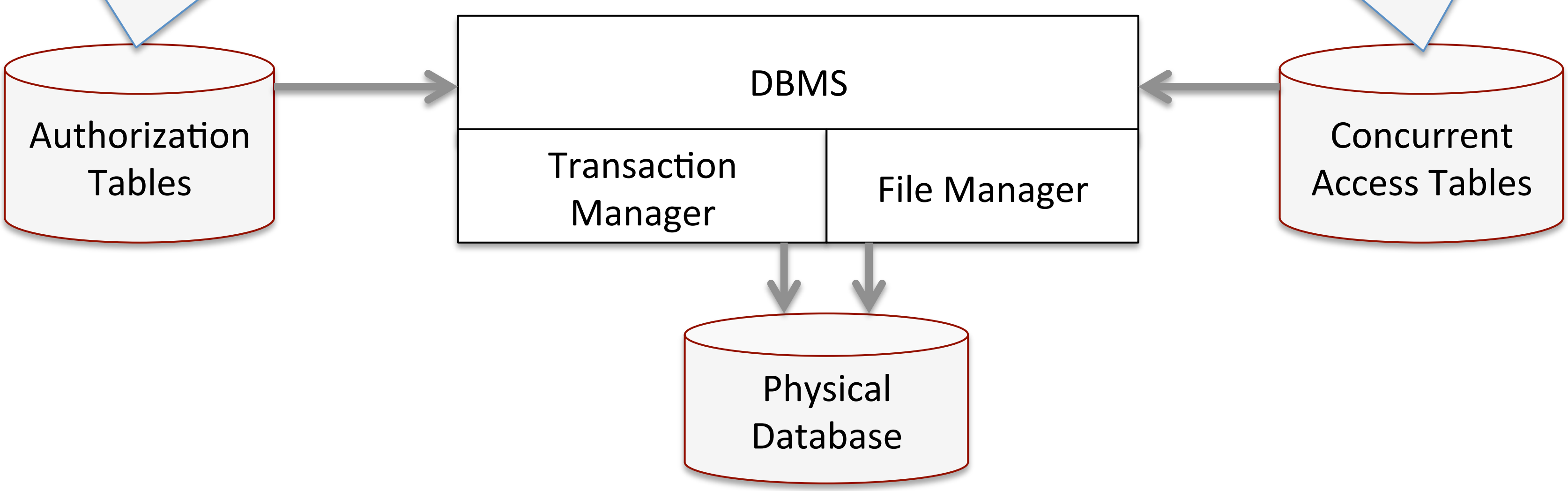
… In addition to DDTs

Authorization Tables

Ensure the user has permission to execute the query language statement on the database
(***Access Control Support***)

Concurrent Access tables

Prevent conflicts when simultaneous (concurrent) conflicting commands are executed.
(***Concurrency Control***)



DBMS Security

Part I

- DBMS
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

DB Architecture and Security Issues

- Database systems provide efficient access to large volumes of related data that are vital to the operation of many organizations.

- Because of their complexity and criticality, database systems generate security requirements that are beyond the capability of typical OS-based security mechanisms or stand-alone security packages.

Database security

- Database attacks and countermeasures must be considered, orthogonally to other defenses:
  - Network-Access Control Services
  - Firewalls and IPS - Intrusion Prevention Systems
  - IDS - Intrusion Detection Systems

  - Operating Systems Security Services and Mechanisms
  - Software Security Mechanisms and Techniques

  - Other security Management and Operational Issues in Datacenters
    - Security management and risk assessment
    - IT security controls, plans and procedures
    - Physical and infrastructure security
    - Human resources security
    - Security auditing
    - Legal and Ethical Aspects

Database security problems

Primary concerns

- Security services and mechanisms, as countermeasures against:
  - Attacks by possible (malicious) SQL injection
  - Inband SQLi attacks
  - Inferential attacks
  - Out-of-Band attacks
  - **Data Access Control and Granularity Issues**
  - **Data-confidentiality and Privacy Concerns**
- Particularly relevant for outsourced databases or Cloud-Based DBaaS environments
  - **Confidentiality, Honest-but-curious adversary models**
  - **How to outsource DBs without outsource Data Control ?**

DBMS

Granularity Issues and Access Control

DB Architecture and Security Issues

- Base Security Services for the Data-Access Layer (orthogonal to OSes, Applications, Middleware Logics) must be supported with appropriate "**fine-grain access control enforcement**"

- Access Control Requirements for RDBMS Model

Access Control Granularity Issues

Granularity and access-control enforcements and flexibility Issues

Operating system security mechanisms typically **control with DAC models read/write/execution access to entire files (the base object-granularity)**, **under OS-User granularity**

- OS security mechanisms could be used to allow a user to read or write information in, for example, a personnel file, fils sharing authorizations etc …
  - But **those mechanisms could not be used to limit access to specific records or fields or specific entries in that file** (as a file containing specific data-structures)

DB Access Control Granularity

- What if only support file-grain access control to be specified ?

- Ok for **single tables** as **flat files** ? Problem ?

Example:

A typical telephone directory contains:

one entry for each subscriber with columns for name, telephone number, and address

Subscriber	Name	Telef.	Address

Limitation on using single tables (as flat files) in a DBMS

- For the telephone directory, there might be a number of subscribers with the same name, but the telephone numbers should be unique,
  - so that the telephone number is ok to serve as a unique identifier for a row.
- **But… (problem):**
  - What if two or more people sharing the same phone number might each be listed in the directory ?

Drawbacks with single tables (as flat files)

- To continue to hold all of the data for the telephone directory in a single table and to provide for a unique identifier for each row, we could require a separate column for secondary subscriber, tertiary subscriber, and so on …
  - The result would be that for each telephone number in use, there is a single entry in the table.

- The drawback is that some of the column positions for a given row may be blank (not used).

- Also, any time a new service or new type of information is incorporated in the database, more columns must be added
  - Consequence of these structural change: database and accompanying software must be redesigned and rebuilt.

Fine-Granularity Issues

- A DBMS typically does allow a **type of more detailed and also file-grain access control to be specified.**
- It also usually enables access controls to be specified over a wider range of commands, such as to:
  - **select, insert, update, or delete operations**
  - **over specified items in the database.**

- Thus, DB security services and mechanisms are needed (beyond OS services)
  - They must be designed specifically for, and integrated with, DBMS architecture

DBMS Security

Part I

- DBMS
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

RDBMS (checklist)

- What is a RDBMS ? What is the RDBMS Building Blocks ? Tables (Raws: as tuples; Columns: as attrributes), Table-Links/Relations and Query Language
- What are Primary Keys (uniqueness) vs. Foreign Keys (non-uniqueness)
- Querying on Multiple Tables
  - How to create relatioships between tables ?
- What are Views ... Views as Virtual Tables
- SQL Querying, SQL Operations, SQL Programming

RDBMS

- **Relational Database Management Systems**
- **Base building block: Table of Data**
  - A Table consists of rows and columns
    - Each column holds a particular type of data
    - Each row contains a specific value for each column
  - **Ideally has one column where all values are unique, forming an identifier/key for that row**

RDBMS: Tables, Table-Links, Relations and Query Languages

- Multiple tables are created and linked together by a unique identifier that is present in all tables
  - Use a relational query language to access the database
  - Allows the user to request data that fit a given set of criteria, expressed in query-language statements

Querying on Multiple Tables

- **More flexibility**: the software figures out how to extract the requested data **from one or more tables**.
  - With a relational design model, **we can have a main table** (**or primary table**) **that never requires a reconstruction**
  - **And we can structure multiple tables, relating then with a primary key**
    - DB administrator can define new tables,
      - each one with a column for the primary key
      - and any number of columns with the required information

Querying on Multiple Tables

- **For example, a telephone company representative could retrieve:**
  - a subscriber's billing information

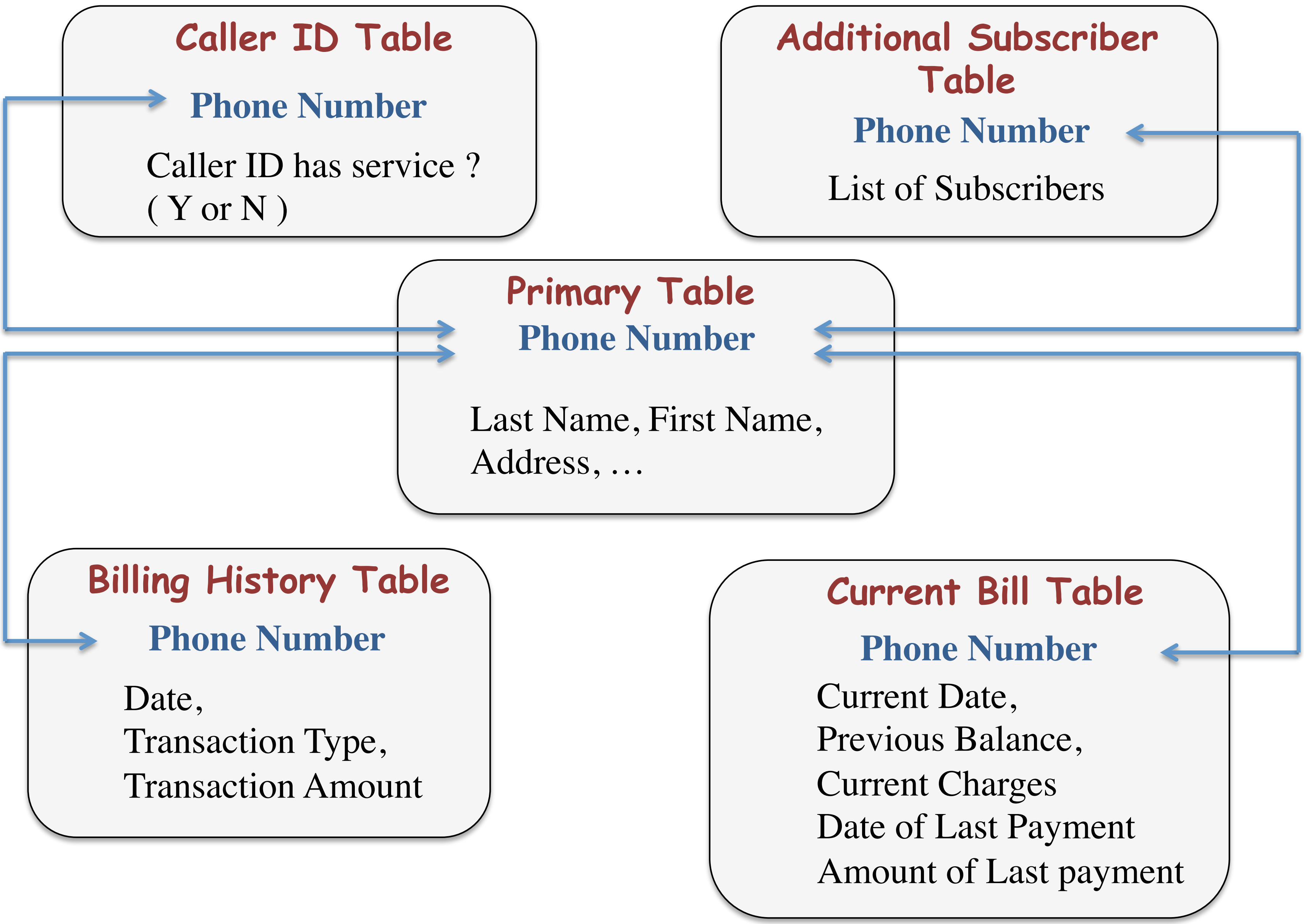
    AND

  - as well as, the status of special services

    OR

    - the latest payment received, all displayed on one screen.

RDBMS Tables (example)



RDBMS elements: Relation, Row, Column and Primary Key

- **Relations**: which are flat tables
- **Rows**: are tuples
- **Columns**: are attributes
- **Primary Key**
  - is defined to be **a portion of a row used to uniquely identify a row in a table**
    - Used as the row unique identifier
    - A **portion** means that a primary key **may be one or more column names**

RDBMS base design elements

- **Relations**
  - Table/File
- **Rows**
  - Tuple/Record
- **Columns**
  - Attributes/Fields

Formal Name	Common Name	Also Known As
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

- **Primary Key**
  - One or more (unique) column names
- **Foreign Key**
  - Links a table to attributes in another table

- **View**
  - Result of a query (as selected rows and columns from one or more tables)

RDMS Table: Abstract Model

M atributes

N individuals, or entities

Attributes

Records

	A_1	• • •	A_j	• • •	A_M
1	x_{11}	• • •	x_{1j}	• • •	x_{1M}
•	•		•		•
•	•		•		•
•	•		•		•
i	x_{i1}	• • •	x_{ij}	• • •	x_{iM}
•	•		•		•
•	•		•		•
•	•		•		•
N	x_{N1}	• • •	x_{Nj}	• • •	x_{NM}

Each attribute A_j has $|A_j|$ possible values
With x_{ij} denoting the value of attribute j for entity i .

RDBMS elements:
Table Relationships and Foreign Keys

- **How to create a relationship between two tables ?**
  - the attributes that define the primary key in one table must appear as attributes in another table
  - In that table they will be considered as the foreign key
- **Uniqueness of primary keys**
  - The value of a primary key must be always unique for each tuple (row) of its table,
- **Non-uniqueness of foreign keys**
  - But a foreign key value can appear multiple times in a table
  - So that there is a one-to-many relationship between a row in the table with the primary key and rows in the table with the foreign key.

RDBMS elements: Views

- **What is a VIEW ?**
  - **A view is a virtual table**
  - In essence, **a view is the result of a query that returns selected rows and columns, from one or more tables.**

- **Views are often used for security purposes**
  - **A view can provide restricted access to a relational database**
    - so that **a user or application only has access to certain rows or columns.**

Examples

Two possible tables in a database

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

Did: primary key

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

Did: foreign key; Eid: primary key

A **view** that can be derived from the database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

SQL (Structured Query Language)

- **SQL is a Standardized language**
  - That can be **used to define schema, manipulate, and query data in a relational database**
  - **Several similar versions of ANSI/ISO standard**
    - **Variety of different implementations**
    - **All following the same basic syntax and semantics**

- **SQL Statements**
  - can be used:
    - to create tables
    - insert and delete data in tables
    - create views
    - and retrieve data with query statements.

DBMS Security

Part I

- DBMS
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

Confidentiality

- Today DBs are typically the most important resources for many organizations
  - Therefore protected by multiple security layers and services
    - Firewalls, IDs, Authentication/SSO Systems, Access-Control Services, and specific DB Access Control Services and Mechanisms
  - But additionally, for particular sensitive data, DB encryption must be provided for
    - Data Confidentiality Guarantees
    - Privacy-Preserving Operation
    - (as the last line of defense: example:
      - Protection from external hackers and Intruders that overcome all the perimeter-defense mechanisms or vulnerabilituie sin the above systems
      - Protection from insider “honest-but-curious DB/System Administrators)

Confidentiality

- Particular Issue today, considering:
  - Outsourced Databases / Outsourced DataCenters
  - Cloud-Based Solutions
  - DBaaS

Attractive Solutions (ex., Cloud-Provided Solutions)

Cheap/Pay-Per-Use Models, High-Availability, Scalability, Elasticity, Easy-to-Deploy, Disaster & Recovery Prevention, Efficient-Ubiquitous Access and Update …

But Confidentiality and Privacy are the Main Concerns !

Sensitive Data and Critical Databases

- Corporate Financial Data
- Confidential Phone Records
- Customer and employee information, such as Name, Social Security Number, Salary, Bank Account Information, Credit Card Information
- Proprietary product information, Customer-information, Commercial Proposals, …
- Health care management information, Health Records and Medical records, BioBanking Data
- Etc …

Disadvantages and Difficulties to DB Encryption

- Need effective Key-Management Services
- It is not an easy-task:
  – Multi-user environment, Multi-Role Responsibilities
  – Multi-Access Vectors
    - Different Applications, Different Middleware Services, for example in the context of 3-to-N Tier SW Architectures

Disadvantages and Difficulties to DB Encryption

- Inflexibility issues
  - More difficult to perform queries, record searching, logging-control
  - How to support full-fledged operations ?

Disadvantages and Difficulties to DB Encryption

- Granularity Issues
  - Encrypt the Entire Database ?
  - Encrypt at the Record Level (Selected Records, Lines) ?
  - Encrypt at Attribute Level (Columns) ?
  - Encrypt Individual Fields ?

  A number of approaches exist:

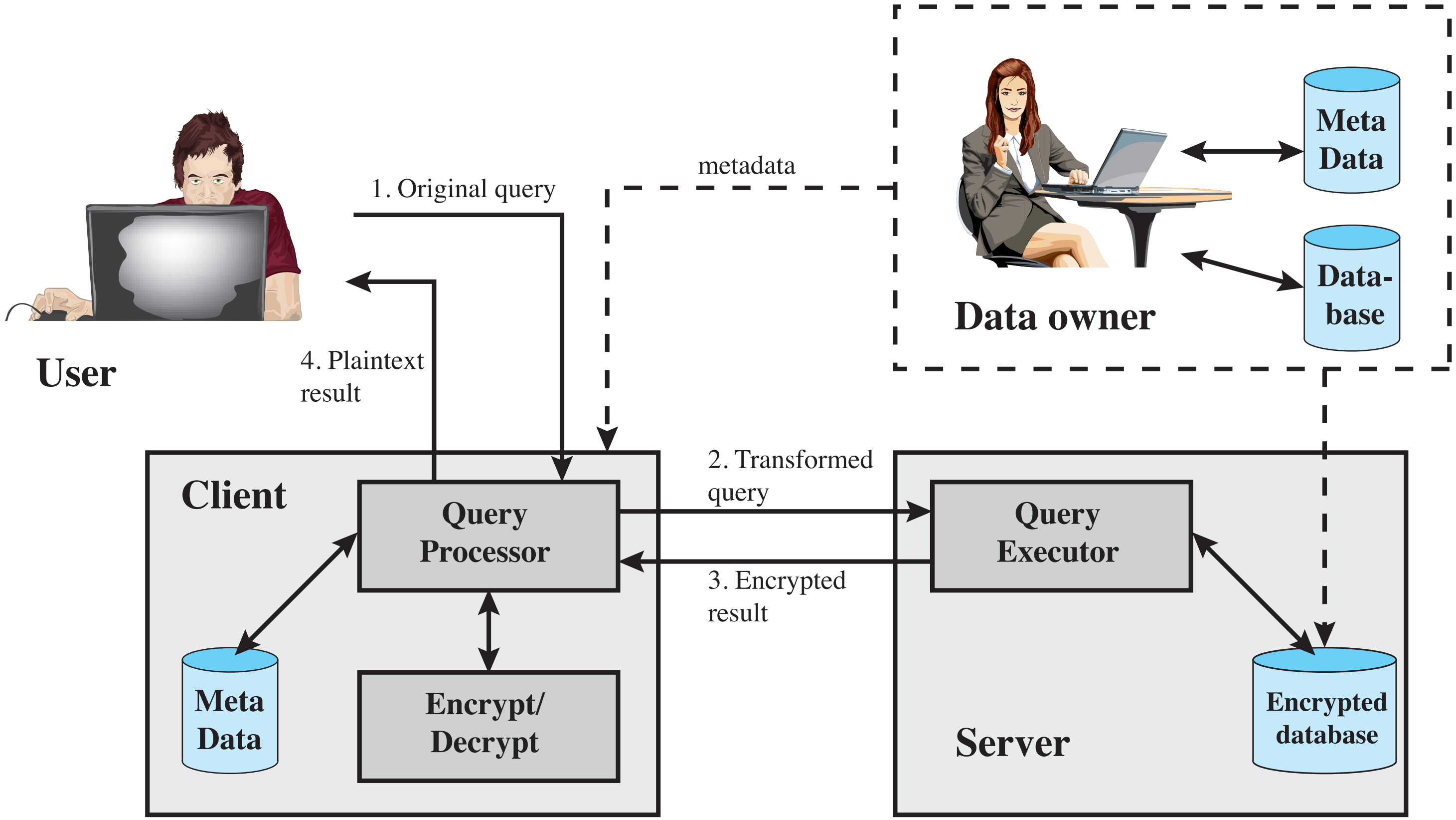
  - Industrial Solutions
  - Research Solutions

Disadvantages and Difficulties to DB Encryption

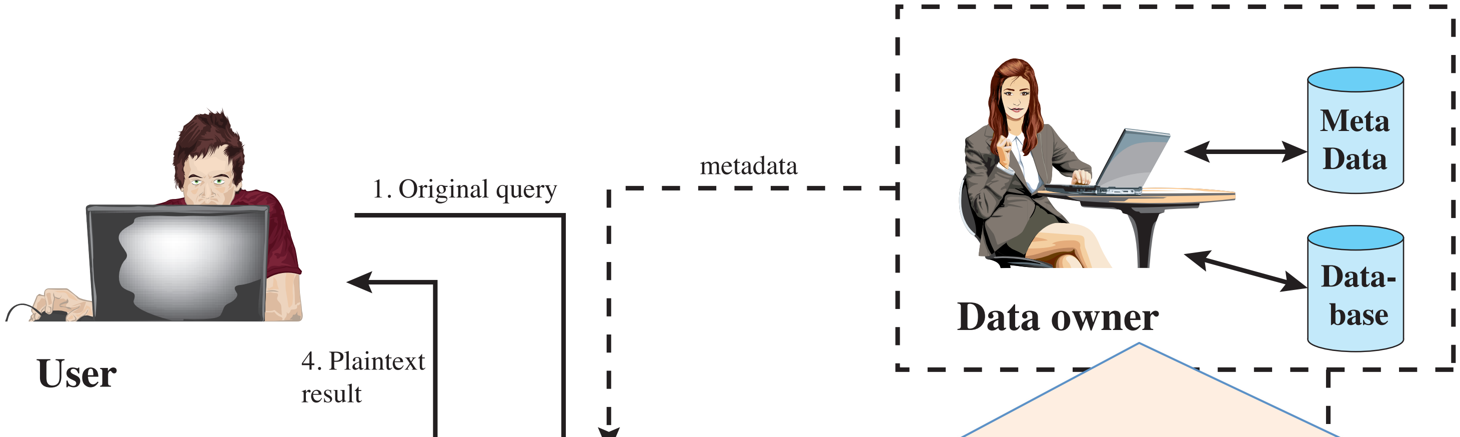
- A straightforward solution
  - Encrypt the entire DB (or entire Portions)
  - Keys in the DB Side (DB Admin/Management Side)
  - Key-Management in the DB Provider side

- And this means “outsource control”
- Or such a solution is not flexible
  - User has little ability to access individual data items based on searches or indexing on key parameters
  - Rather would have to download entire tables, decrypt tables and work with the results
    - This is known as “Security-on-the-Rest” solutions

Database Encryption Model

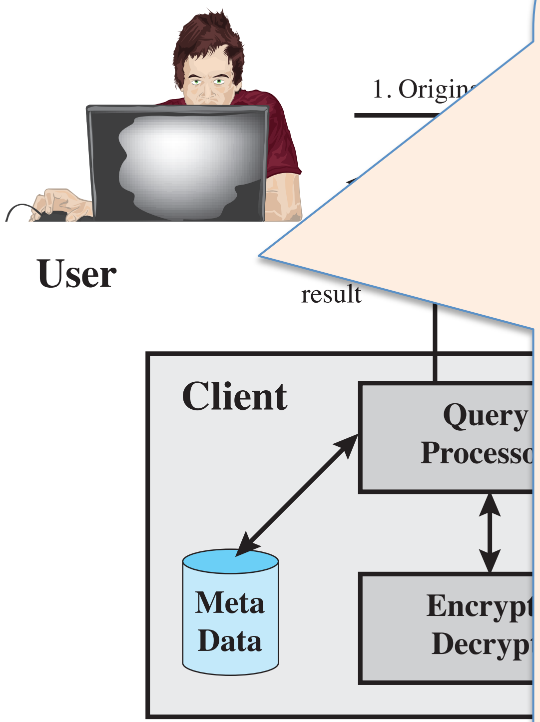


Database Encryption Model



Data-Owners (Data-Subjects):
An Organization or User that Produces Data to be Made Available for Controlled Release, either within organizations or to external users

Database Encryption Model



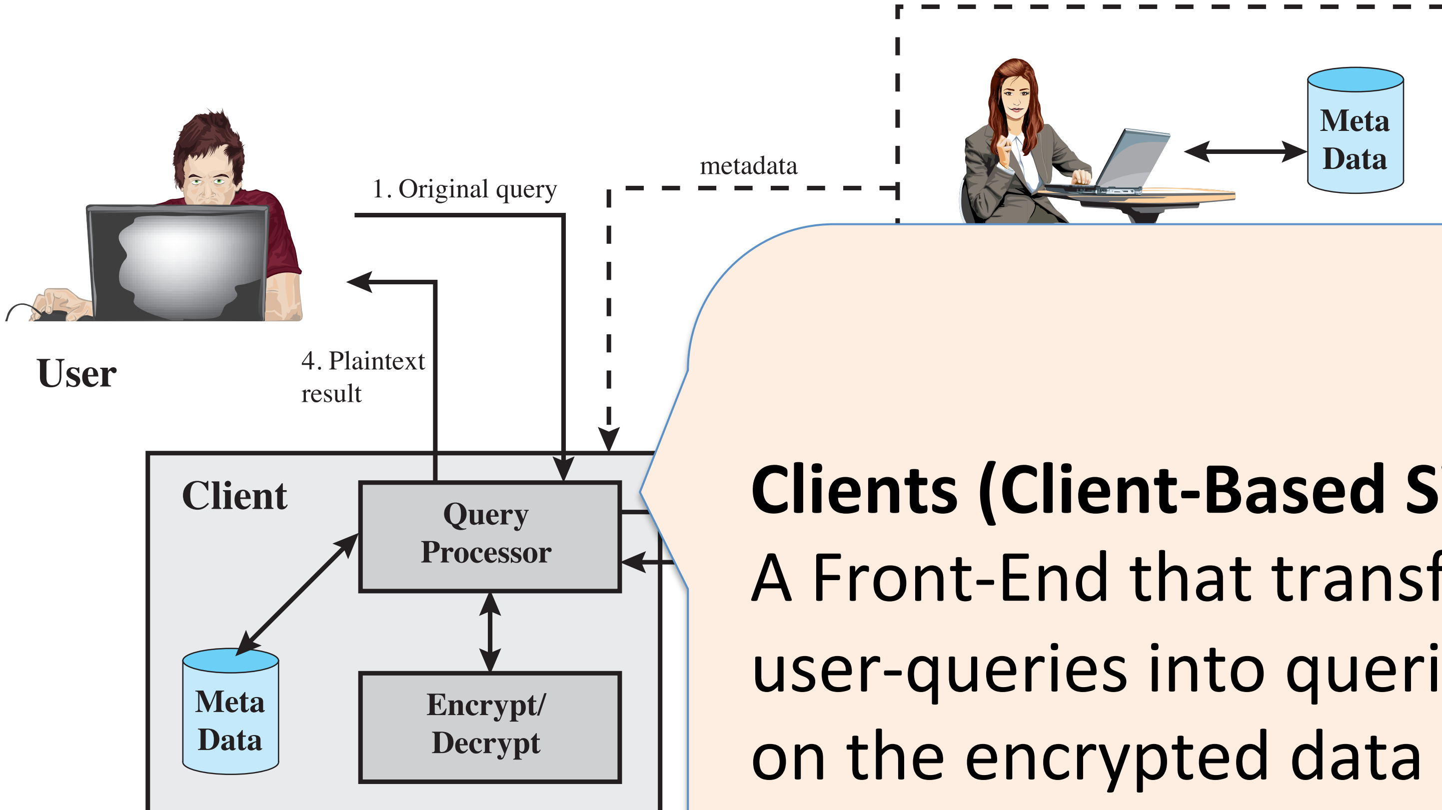
Users :

Human entity that submit requests (queries)

Could be an employee of an organization who is granted access to the database, via a service (front-end, web-app server, …)

Can be an external user who can access data after an authentication and access-control process

Database Encryption Model

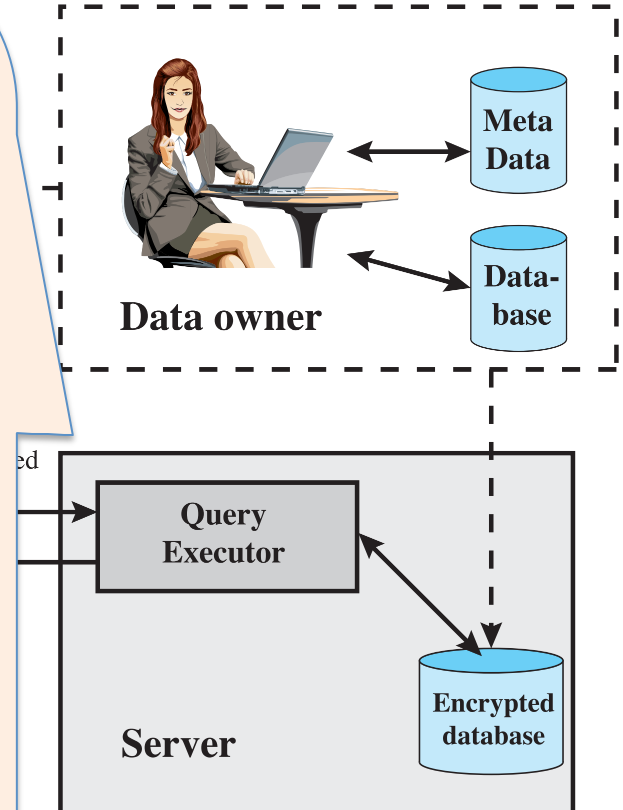


Clients (Client-Based SW) :
A Front-End that transforms user-queries into queries on the encrypted data stored on the DB server

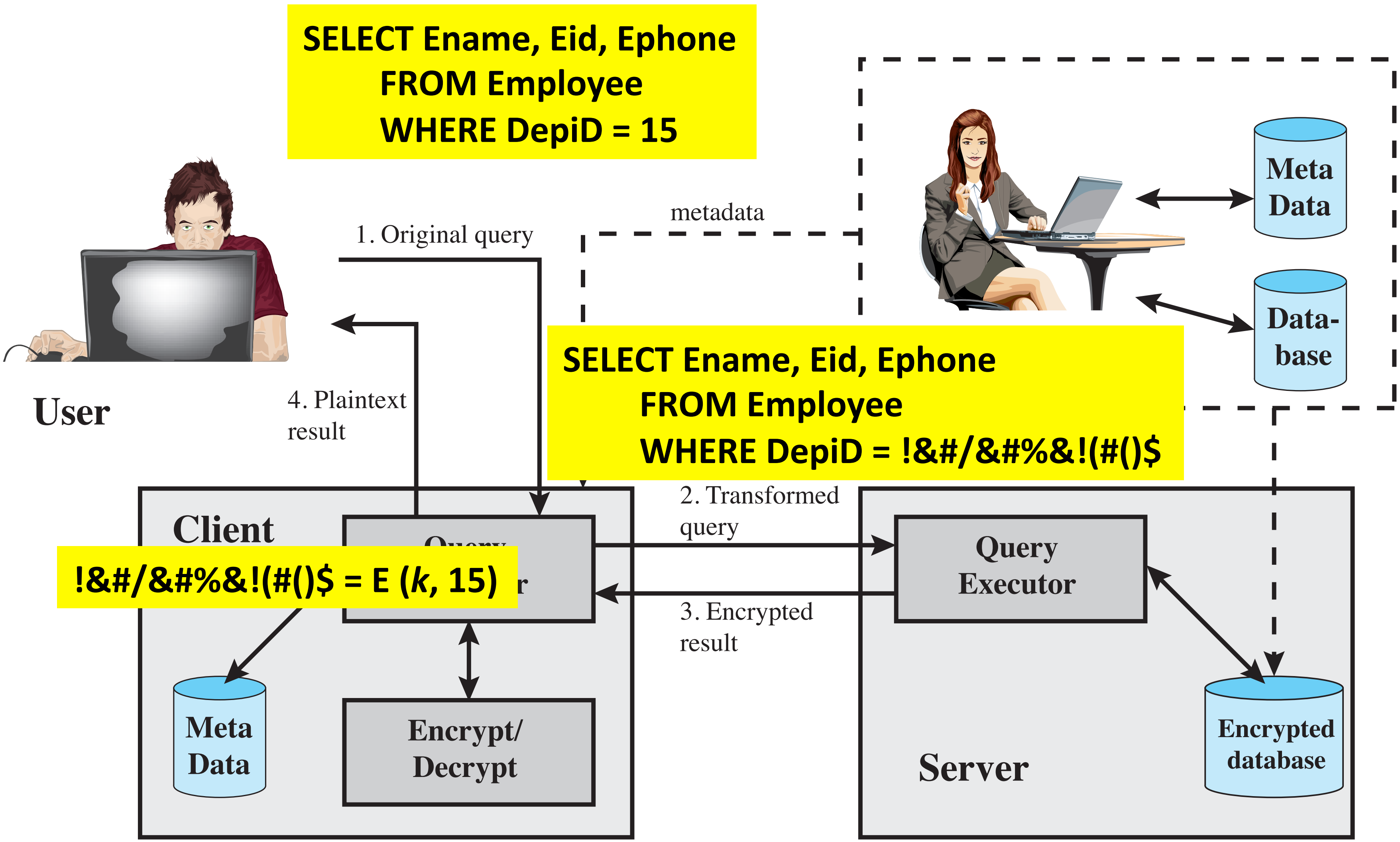
Database Encryption Model

Server (or Service)
An Organization (entity, resource) that receives encrypted data from data-owners and makes them available for client-access / distribution

Could be owned by the Data-Owner (or Data-Subject) but, **more typically, is a facility owned and maintained by an external provider**



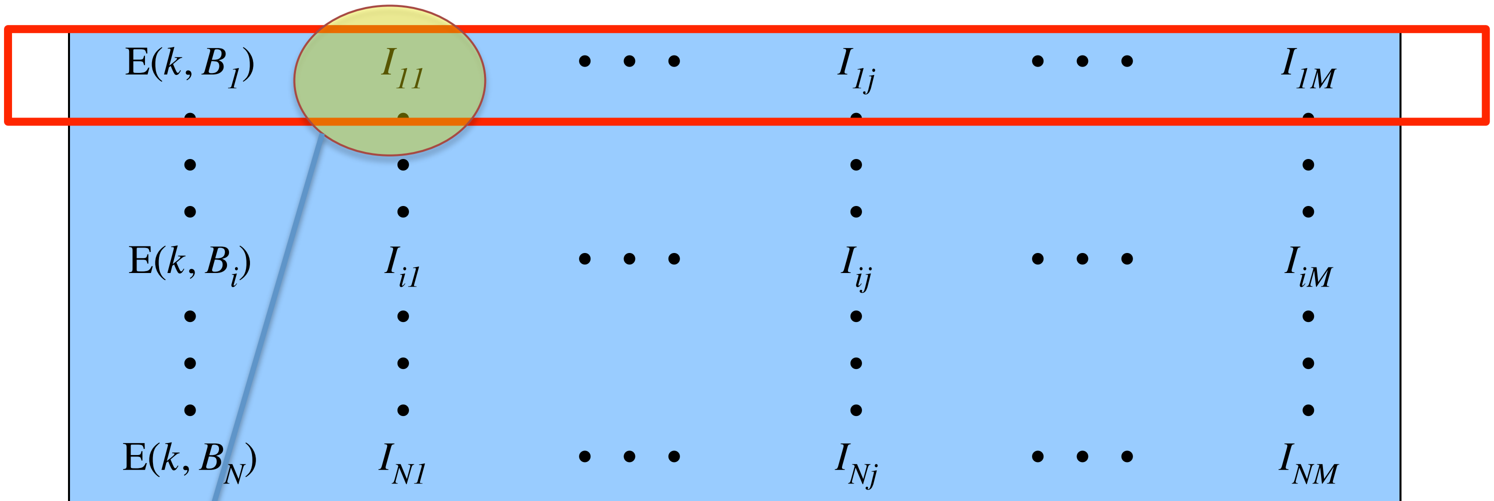
Database Encryption Model



Flexibility Problems to have a Full-Fledged Solution ?

A Base Encryption Scheme for a DB

Encrypt lines as a block: a contiguous block Bi = (Xi1 || Xi2 || … Xin)



$B_i = (x_{i1} \parallel x_{i2} \parallel \ldots \parallel x_{iM})$

Other possible approaches ?
Problems (other required services) ?

A sequence of BITS in the Block

E (k, Bi) = E (k, (xi1 || xi2, xi3 , …… xiN) => [E(k,Bi), Ii1,, Ii2, Ii3, …. IiN)

Example

Employee Table

eID	eName	Salary	Addr	DepID
23	Tom	70K	Marple Road, 23	45
860	Mary	60K	Main Rooad, 1	83
320	John	50K	River Street, 2	50
875	Jerry	55K	Hopewell Av. 456	92

Supposing we know that the iID values are in the range [1, 1000], we can divide these values in five partitions, assigning indexes, ex:

[1, 200] ….. 1
[201, 400] ….. 2
[401, 600] ….. 3
[601, 800] ….. 4
[801, 1000] ….. 5

Meta-Data on the Indexing Process Only-Known for the Client (Not Stored in the DB Server)…

Example

Employee Table

eID	eName	Salary	Addr	DepID
23	Tom	70K	Marple Road, 23	45
860	Mary	60K	Main Rooad, 1	83
320	John	50K	River Street, 2	50
875	Jerry	55K	Hopewell Av. 456	92

For text field we can derive an index from the first letter of the attribute value, ex:

A,B …… 1

C,D …… 2

Etc …

And we can make the same for the other attributes (columns)

Table Transformation

Employee Table

E(*k*, B)	I(eID)	I(eName)	I(Salary)	I(Addr)	I(DepID)
110011011....1111100101	1	10	3	7	4
1021101012010111O...100	5	7	2	7	8
10011000101011..110110	2	5	1	9	5
11110111010001....11010	5	5	2	4	9

Problem:
Some Inference is possible for an adversary ?

Can we avoid it ?
Yes … Randomize the used Indexes

Example

Employee Table

eID	eName	Salary	Addr	DepID
23	Tom	70K	Marple Road, 23	45
860	Mary	60K	Main Rooad, 1	83
320	John	50K	River Street, 2	50
875	Jerry	55K	Hopewell Av. 456	92

Supposing we know that the iID values are in the range [1, 1000], we can divide these values in five partitions, assigning indexes, ex:

[1, 200] ….. 2
[201, 400] ….. 3
[401, 600] ….. 5
[601, 800] ….. 1
[801, 1000] ….. 4

Because Meta-Data are not Stored In the Server Side, the Attacker will Not know nothing about …

Other possible enhancements

To increase the efficiency of accessing records by means of the primary key, **the system could use the encrypted value of the primary key attribute values, or a hash-value**

In both cases, **the row corresponding to the primary key value could be retrieved individually**

Other possible enhancements

Different Portions of the Database could be encrypted with different keys

To have more appropriate granularity …

So that users would only have access to that portion of the DB for which they had the corresponding decryption keys

… This can be better mapped to a Role-Based Access Control Model

Encrypted Database Techniques

- Particularly relevant for:
  - Outsourced databases
  - Cloud-based databases
- Two different models:
  - Security "on the rest": Classic Approach
  - On-Line Security: More Interesting

State-of-Art Related Research

- How to provide more flexibility with Database operations on encrypted data ?

  * Outsourceable Encryption Techniques

  * Homomorphic Encryption Techniques, Schemes and Algorithms

DBMS Security

Part I

- DBMS
  - DBMS and Security Issues
  - RDBMS
- Confidentiality and Database Encryption
- Encrypted Databases and the case for CryptDB

Part II

Other DBMS Security DImensions

- SQL Injection Attacks
- Database Access Control
- Inference Attacks

Material delivered by the authors corresponding to the CryptDB Presentation at SOSP 2011

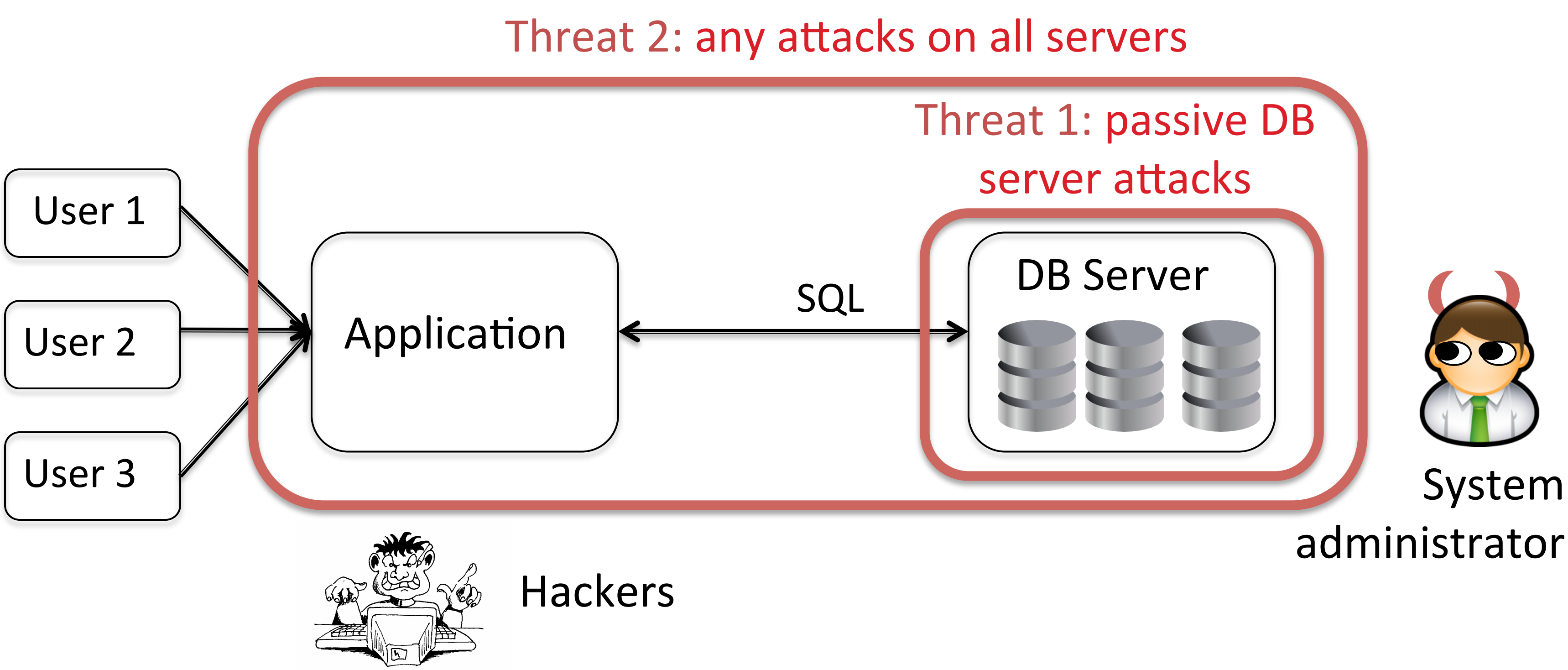
CryptDB: Protecting Confidentiality with Encrypted Query Processing

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

Raluca Popa, C. Redfield, N. Zeldovich, H. Balakrisnan, CryptDB: protecting confidentiality with encrypted query processing, in Proc. SOSP Symposium on Operating System Principles, 2011

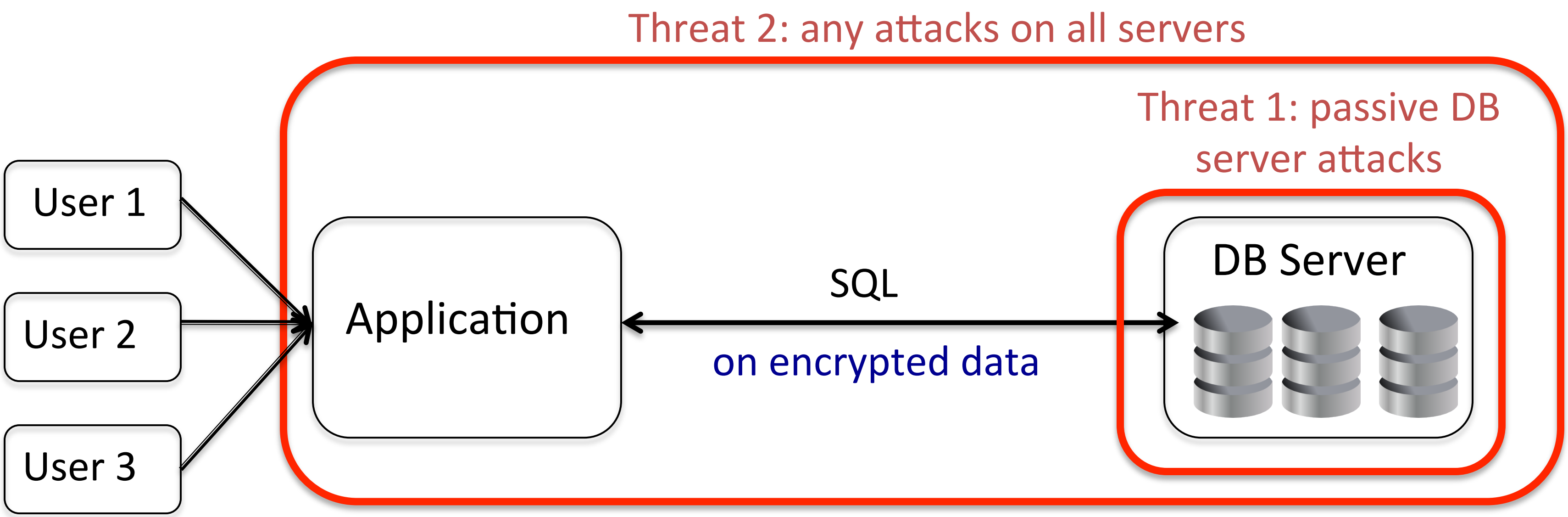
Problem

- Confidential data leaks from databases
  - E.g., Sony Playstation Network, impacted 77 million personal information profiles



CryptDB in a nutshell

- Goal: protect confidentiality of data



1. Process SQL queries on encrypted data
2. Use fine-grained keys; chain these keys to user passwords based on access control

Contributions

1. First practical DBMS to process most SQL queries on encrypted data

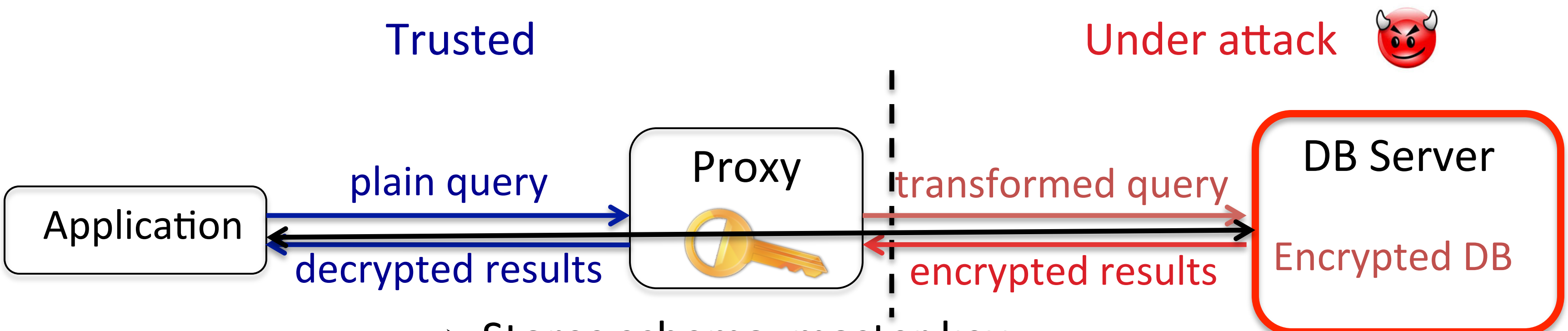
   ➡ Hide DB from sys. admins., outsource DB

2. Protects data of users logged out during attack, even when all servers are compromised

   ➡ Limit leakage from compromised applications

3. Modest overhead: 26% throughput loss for TPC-C
4. No changes to DBMS (e.g., Postgres, MySQL)

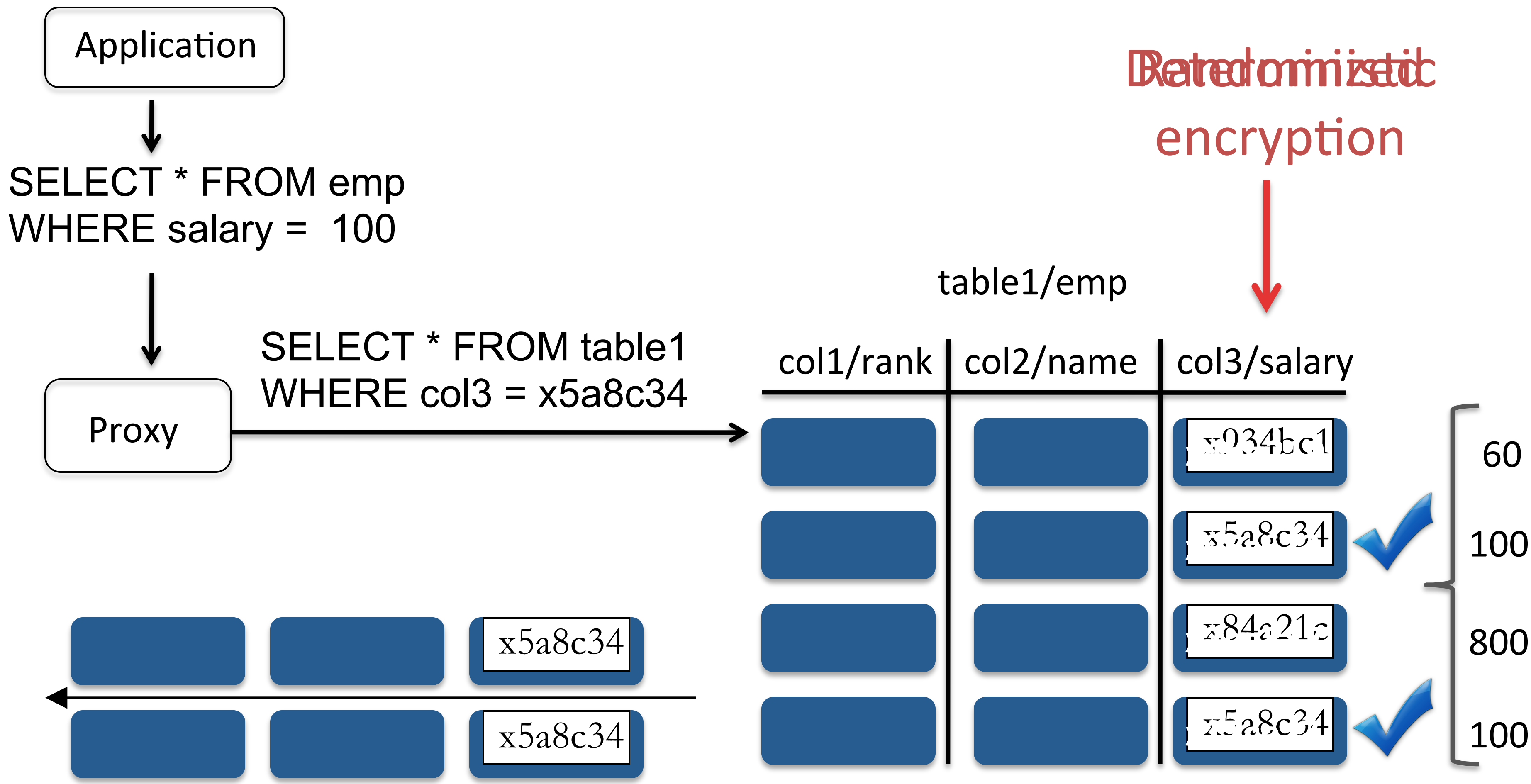
Threat 1: Passive attacks to DB Server

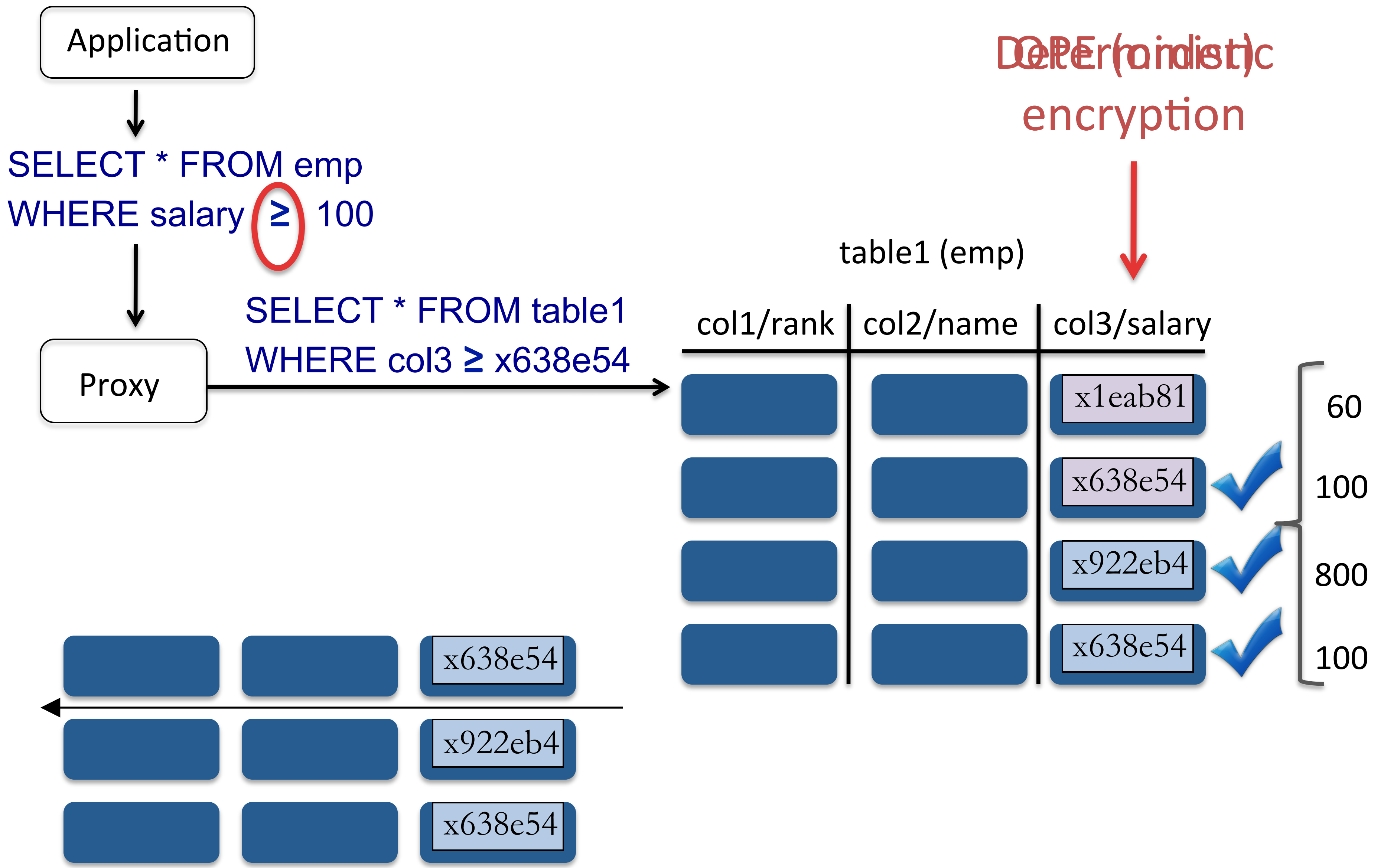


- Stores schema, master key
- No data storage
- No query execution

- Process queries completely at the DBMS, on encrypted database

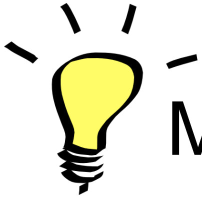
- Process SQL queries on encrypted data


Application
SELECT * FROM emp
WHERE salary = 100
Proxy
SELECT * FROM table1
WHERE col3 = x5a8c34
Deterministic
Randomized
encryption
table1/emp
col1/rank
col2/name
col3/salary
x934bc1
x5a8c34
x84a21c
x5a8c34
60
100
800
100
x5a8c34
x5a8c34


Application
SELECT * FROM emp
WHERE salary ≥ 100
Proxy
SELECT * FROM table1
WHERE col3 ≥ x638e54
Deterministic
OPE (order)
encryption
table1 (emp)
col1/rank
col2/name
col3/salary
x1eab81
x638e54
x922eb4
x638e54
60
100
800
100
x638e54
x922eb4
x638e54

Two techniques

1. Use SQL-aware set of encryption schemes



Most SQL uses a limited set of operations

2. Adjust encryption of database based on queries

Encryption schemes

Highest

Security

Scheme	Construction	Function
RND	AES in CBC	none
HOM	Paillier	+, *
SEARCH	Song et al.,'00	word search
DET	AES in CMC	equality
JOIN	our new scheme	join
OPE	Boldyreva et al.'09	order

e.g., sum

restricted ILIKE

e.g., =, !=, IN, COUNT, GROUP BY, DISTINCT

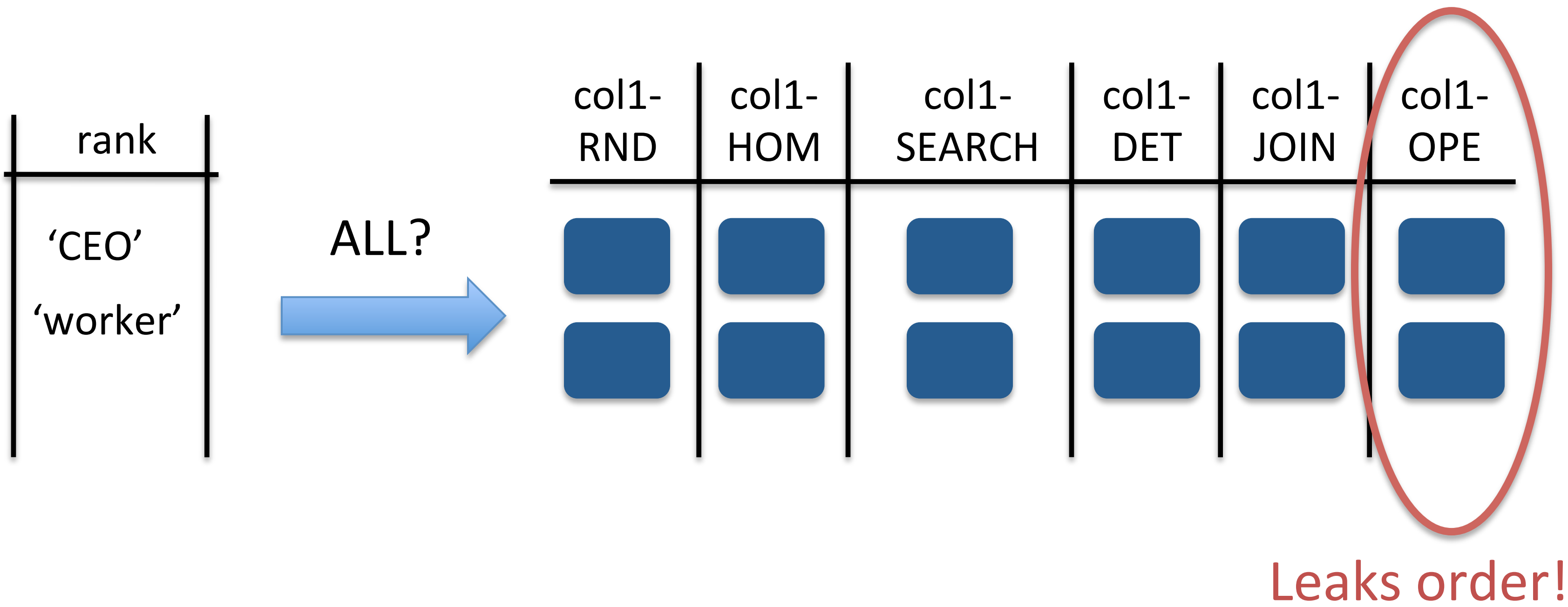
see paper

e.g., >, <, ORDER BY, SORT, MAX, MIN

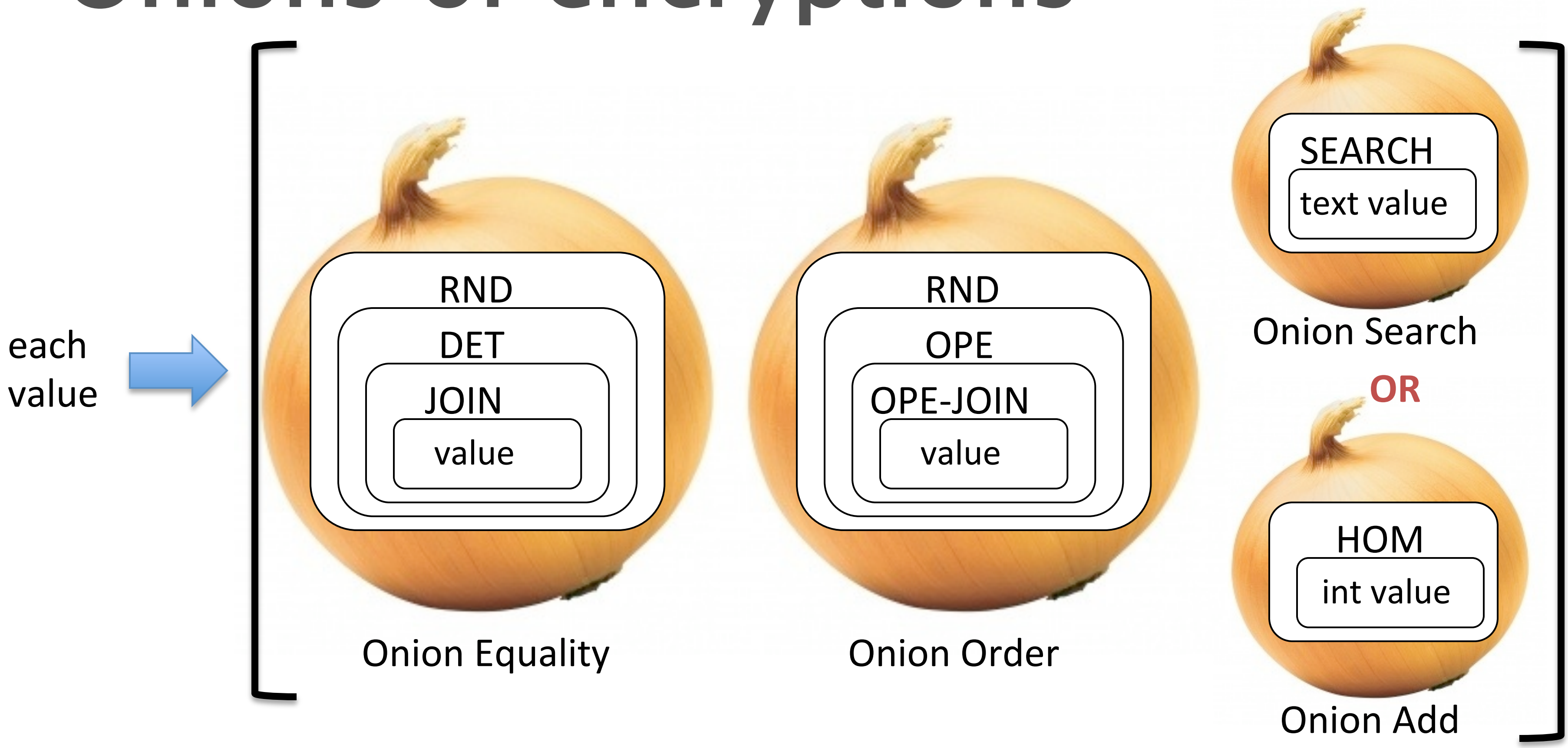
first implementation

How to encrypt each data item?

- Encryption schemes needed depend on queries
- May not know queries ahead of time



Onions of encryptions

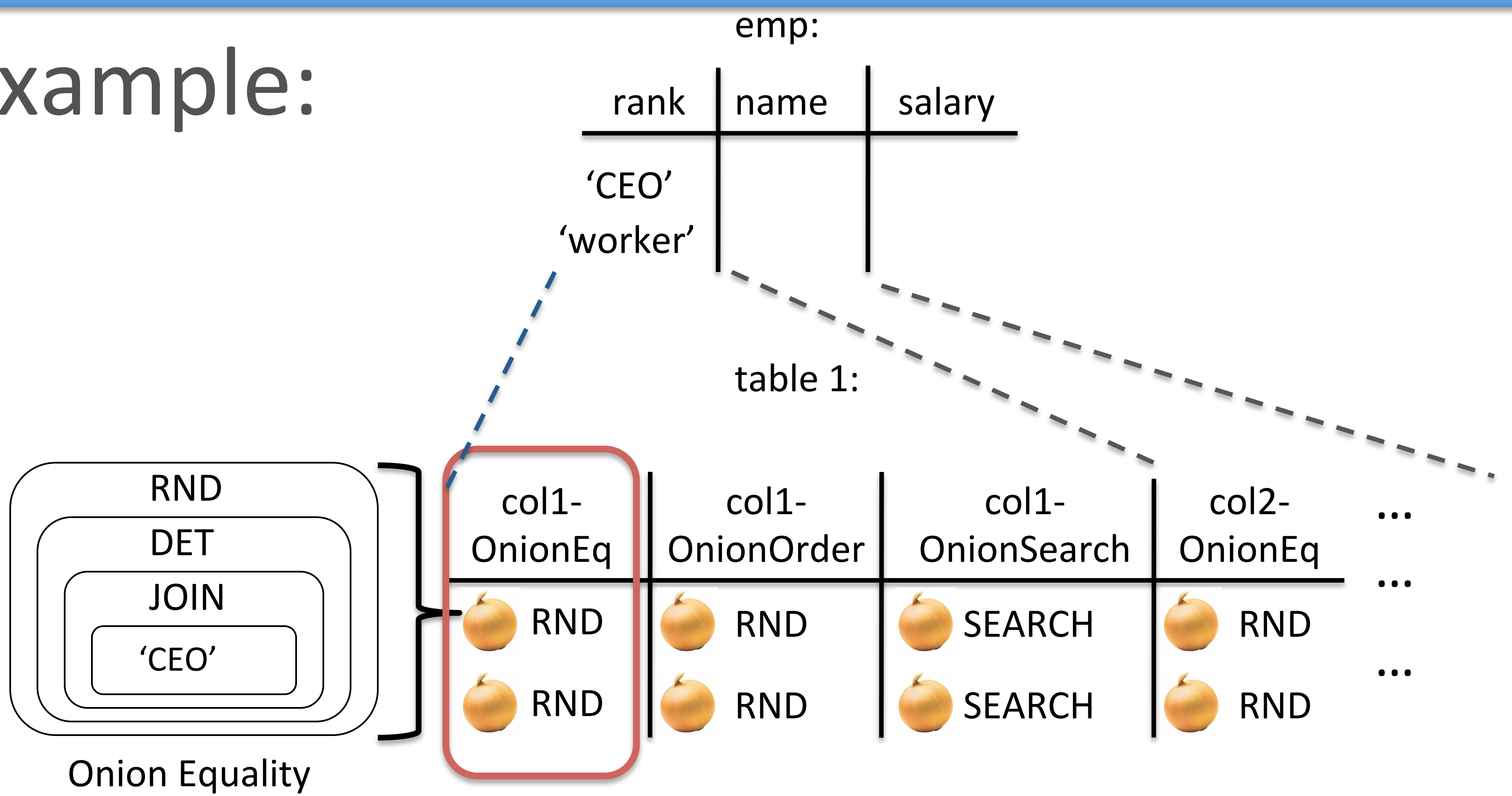


- Same key for all items in a column for same onion layer
- Start out the database with the most secure encryption scheme

Adjust encryption

- Strip off layers of the onions
  - Proxy gives keys to server using a SQL UDF ("user-defined function")
  - Proxy remembers onion layer for columns
- Do not put back onion layer

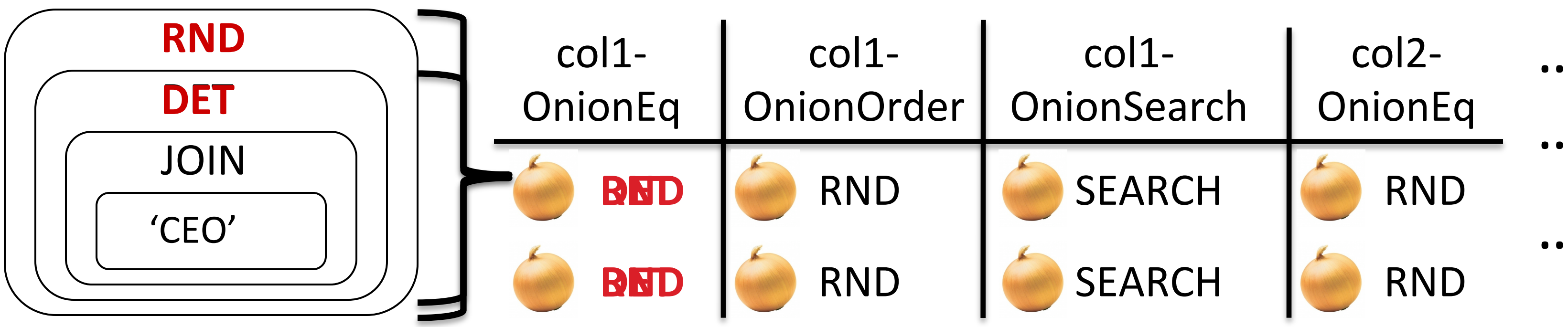
Example:



SELECT * FROM emp WHERE rank = ‘CEO’;

Example (cont’d)

table 1



Onion Equality

SELECT * FROM emp WHERE rank = ‘CEO’;

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

SELECT * FROM table1 WHERE col1-OnionEq = xda5c0407;

Confidentiality level

Queries → Encryption scheme exposed → amount of leakage

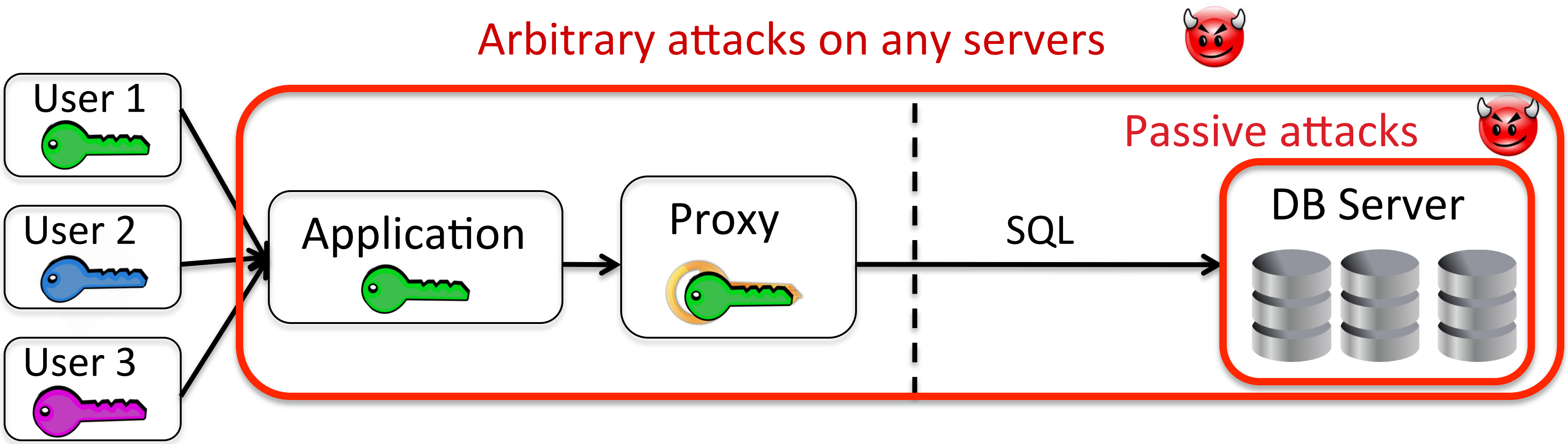
➢ Encryption schemes exposed for each column are the most secure enabling queries

- equality predicate on a column → DET → repeats
- aggregation on a column → HOM → nothing
- no filter on a column → RND → nothing

common in practice

→ Never reveals plaintext

Application protection



- User password gives access to data allowed to user by access control policy
- Protects data of logged out users during attack

Challenge: data sharing

	SPEAKS_FOR msg_id	SPEAKS_FOR msg_id
msg_id	sender	receiver
5	Alice	Bob
	[Km5] Alice-pass	[Km5] Bob-pass

	ENC_FOR msg_id
msg_id	message
5	[“secret message”] Km5

1. How to enforce access control cryptographically?

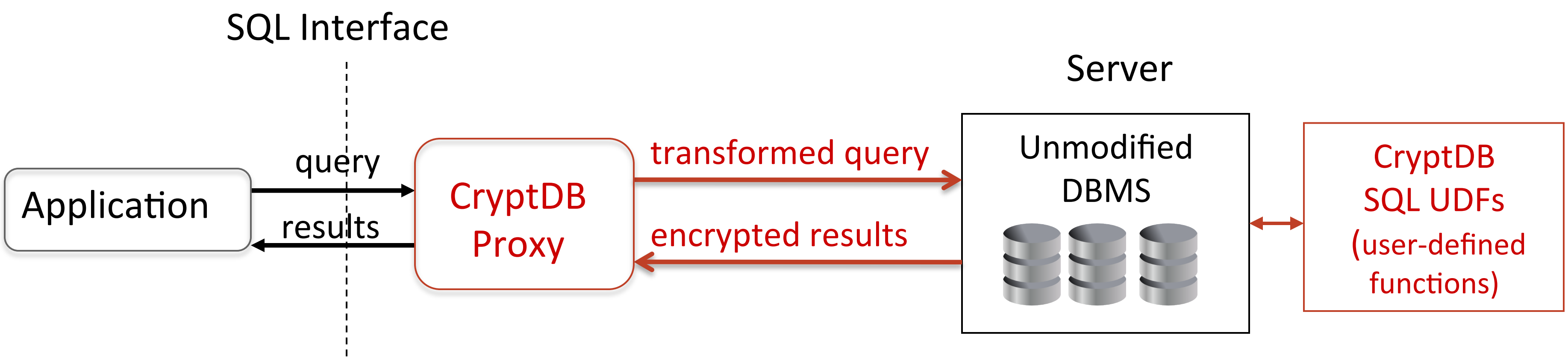
   → Key chains from user passwords

2. Capture read access policy of application at SQL level?

   → Annotations

3. Process queries on encrypted data

Implementation



- No change to the DBMS
- Portable: from Postgres to MySQL with 86 lines
- One-key: no change to applications
- Multi-user keys: annotations and login/logout

Evaluation

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

Queries not supported

- More complex operators, e.g., trigonometry
- Operations that require combining incompatible encryption schemes
  - e.g., T1.a + T1.b > T2.c

⇨ Extensions: split queries, precompute columns, or add new encryption schemes

Real queries/applications

	Application	Total columns	Encrypted columns	# cols not supported	Annotations + lines of code changed
Multi-user keys	phpBB	563	23	0	38
	HotCRP	204	22	0	31
	grad-apply	706	103	0	113
One-key	TPC-C	92	92	0	0
	sql.mit.edu	128,840	128,840	1,094	0

SELECT 1/log(series_no+1.2) …

… WHERE sin(latitude + PI()) …

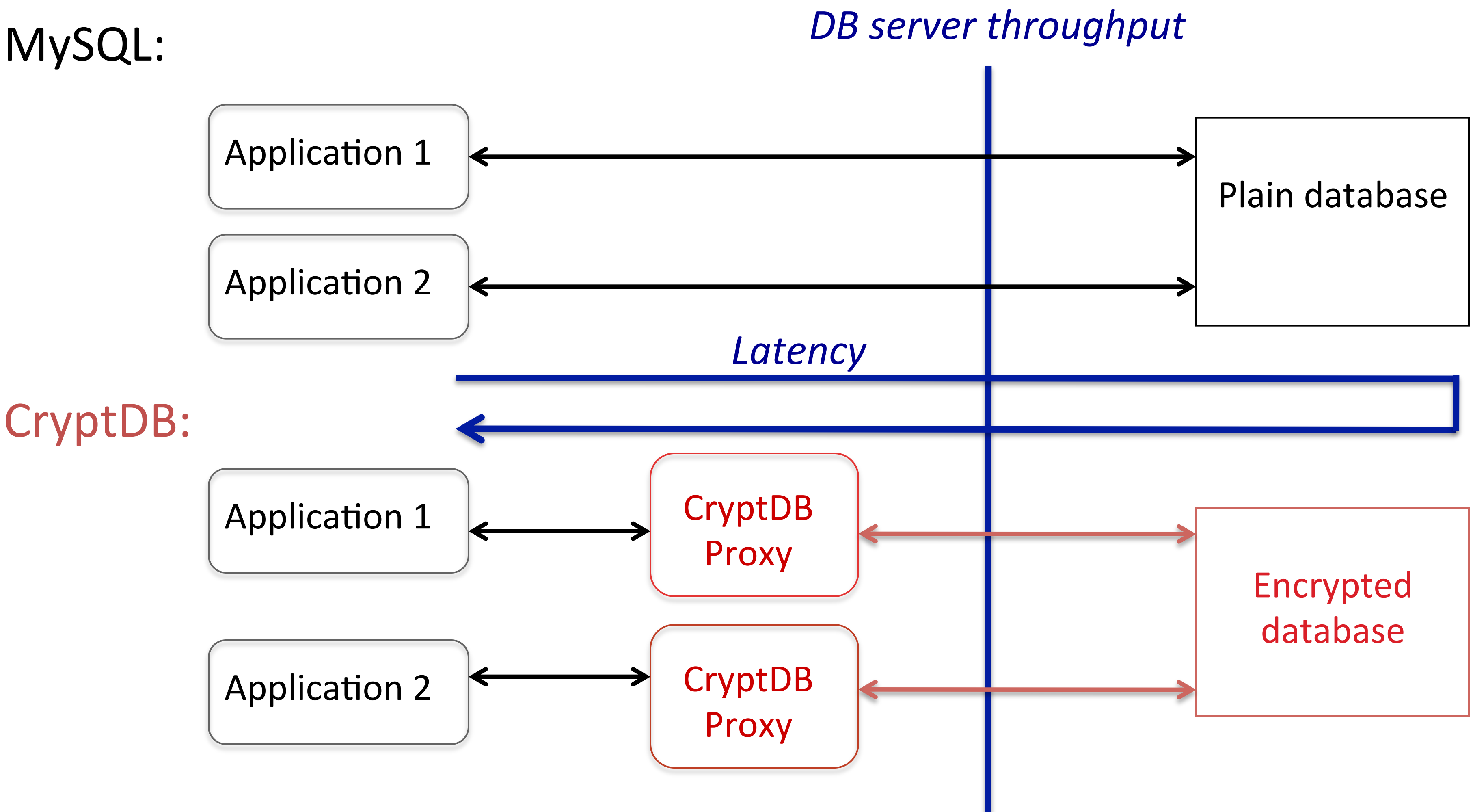
Resulting confidentiality

	Application	Total columns	Encrypted columns	Min level is RND	Min level is DET	Min level is OPE
Multi-user keys	phpBB	563	23	21	1	1
	HotCRP	204	22	18	1	2
	grad-apply	706	103	95	6	2
One-key	TPC-C	92	92	65	19	8
	sql.mit.edu	128,840	128,840	80,053	34,212	13,131

Most columns at RND

Most columns at OPE analyzed were less sensitive

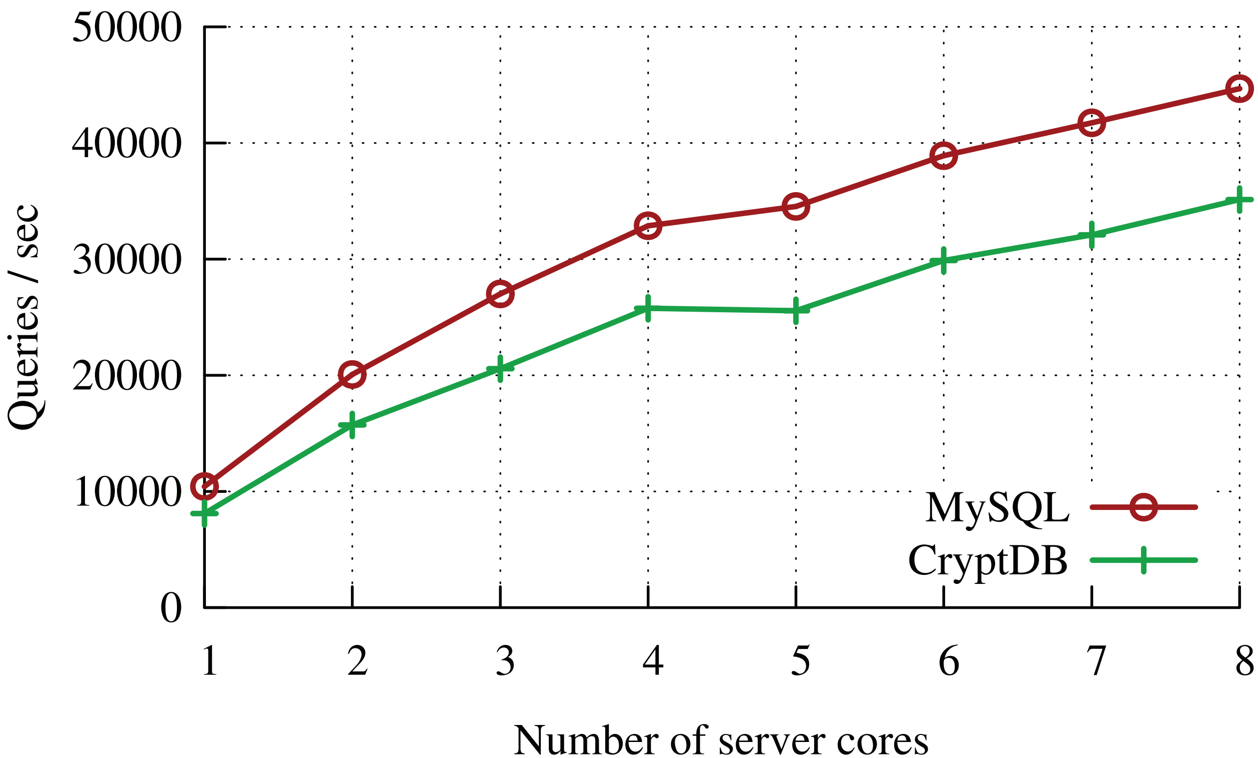
Performance



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

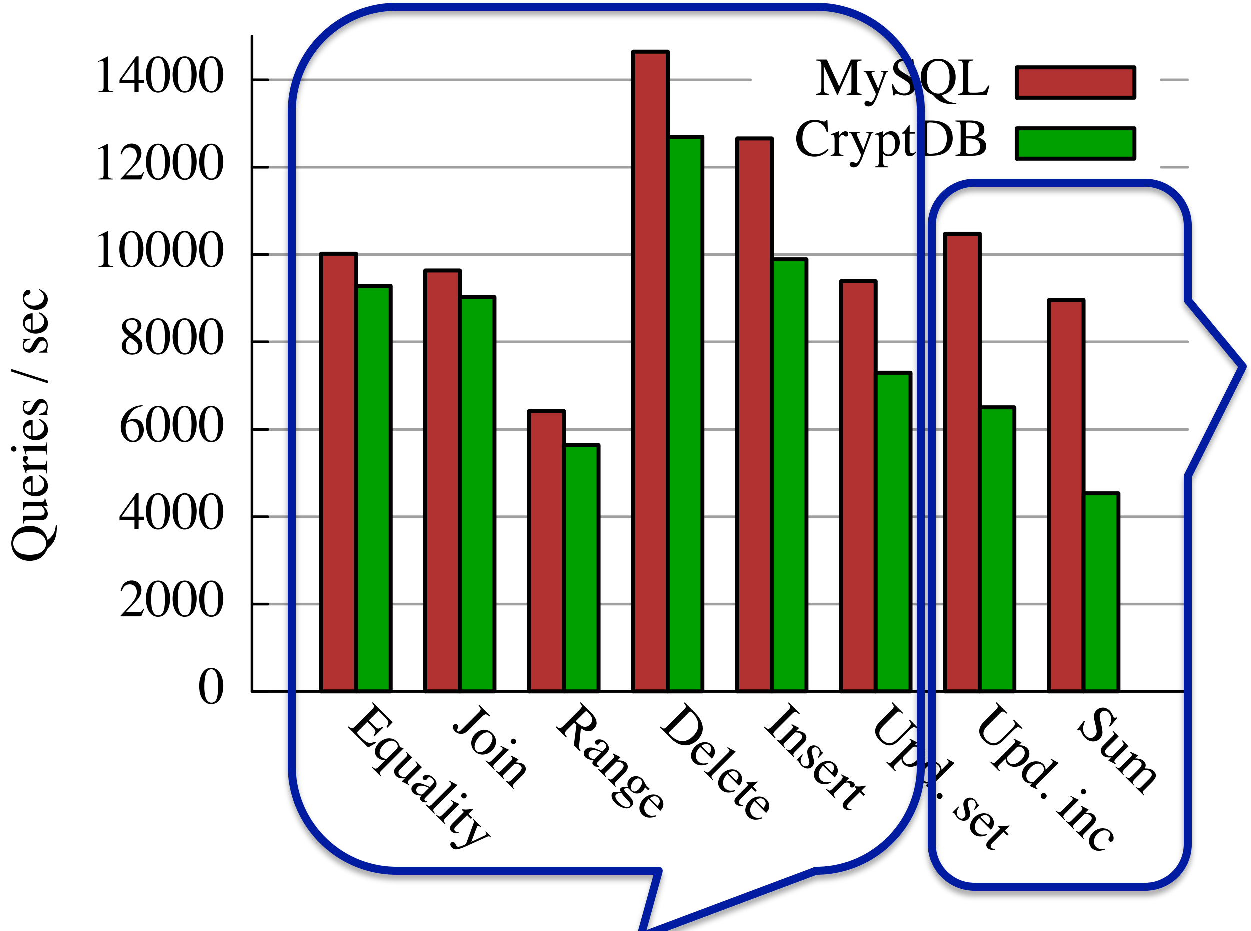
TPC-C performance

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



Throughput loss 26%

TPC-C microbenchmarks



Homomorphic addition

No cryptography at the DB server in the steady state!

Encrypted DBMS is practical

Related work (See the bibliography in the paper)

- Cryptography proposals
  - Fully homomorphic encryption
  - Search on encrypted data
  - Systems proposals
  - Lower degree of security, rewrite the DBMS, client-side processing

- Query integrity

Conclusions

CryptDB:

1. The first practical DBMS for running most standard queries on encrypted data
2. Protects data of users logged out during attack even when all servers are compromised
3. Modest overhead and no changes to DBMS

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

Demo at poster session!

Thanks!

Database security problems (initially stated), not covered by CryptDB

- NO – Attacks by possible (malicious) SQL injection
- NO – Inband SQLi attacks
- NO – Inferential attacks
- NO – Out-of-Band attacks
- Ok – Data-confidentiality and Privacy Concerns
- Ok – Outsourced databases or Cloud-Based DBaaS environments
  - Access Control Services
  - Confidentiality, Honest-but-curious adversary models
- NO
  - Other issues: integrity, user-authentication, … resistance against malicious intruders, DoS, …

Remarks/Discussion/Open-Research

- Onion Model:
  - Considerable space overhead to store multiple copes of all columns (one per onion) in general, for different Database EDR Models ?
  - Some onions can be larger that the original data ?
- Performance/Latency
  - Authors argue that 26% Overhead (TPC-C) is reasonable: is it ?
- For Logged On Users in operation, the system cannot have the same guarantees comparing to the case when users are not logged. During the operation, the security conditions are "smoothly" decreased. Can we have a better solution ?
- What if we lose the proxy: single point of failure… Ex., what if we lose the keys (and onion-compositions) … ?

Remarks/Discussion/Open-Research

- More open Research Issues from the CryptDB Design
  - Adversary Model: Only Data Confidentiality Attacks
  - No Protection for Inference Attacks
  - Complete Protection: Only protected when Users are Logged-Off
  - User Authentication Services: User/Pwd, Pwd-Based Encryption translated to Onion-Keys
    - DAC-Model can be mapped with the scheme
  - No Provision for more exigent Access-Control Services
    - RBACs, ABACs, Context_Aware Access Control
  - How to address multi-user environments with Data-Owners/Data Subjects, Different User Roles,…
    - Authentication vs. Key-Distribution Schemes

Remarks/Discussion/Open-Research

- Re-evaluation of the solution in a Cloud-Based SaaS / DBaaS solution
  - New tradeoffs ?
  - New Adversarial Model Considerations ?
- No Full-Fledged SQL Solution
  - How to support other SQL queries ?
    - Ex., Manipulation of Dates, Temporal information, queries on sub-strings, ...
      - In the proposed design: sub-strings in differet columns …
    - Statistical Databases: more queries involving arithmetic compjutations
    - How to protect issues related to other dimensions in the DB: number of rows, number of columns, table structure, approximate sizes od values in cells, … ??
    - What if we have in some columns values that have few-values (ex., nale/female), etc …

Remarks/Discussion/Open-Research

- Other Crypto-Schemes
  - More Homomorphic Schemes (New Crypto Schemes ? :- (((
  - Symmetric Searchable Encryption Techniques ?
  - How to support Multimodal Searches ?
    - Supporting complementarily other Search Operations for Information Retrieval
    - Hybrid Repositories ? DBMS + KVSs ?
      - » Search By Proximity, Similarities
      - » Search by “Ranking” Scores

Remarks/Discussion/Open-Research

- The Client and Proxy in the CryptDB Architecture are "trusted" components.
  - Can we redesign a solution if the Proxy is attacked when there are users logon performing DB queries ? The Proxy can be a particular target of attacks in the system architecture… particularly for multi-iser environmemts
    - The approach… probably just transit the attraction of attackers from the database server to proxy server
  - How to address a "end-to-end" encryption behaviour ? Example: Interception between client and proxy ?

Other Open Issues …

(See referred Drawbacks, Weaknesses, Tradeofs)

- http://web.eecs.umich.edu/~mozafari/fall2015/eecs584/reviews/summaries/summary40.html

- Another approach
  - The Cipherbase approach
  - http://research.microsoft.com/en-us/projects/cipherbase/
  - http://research.microsoft.com/pubs/179425/cipherbase.pdf