TDDB17
Information Security
(VT 2019)

Topic: Database Encryption

Olaf Hartig
olaf.hartig@liu.se
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

<table>
<thead>
<tr>
<th>ID#</th>
<th>SSN</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4945864</td>
<td>12</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>7000111</td>
<td>13</td>
<td>4000</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
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

<table>
<thead>
<tr>
<th>ID#</th>
<th>SSN</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4945864</td>
<td>12</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>7000111</td>
<td>13</td>
<td>4000</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6487539</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7299990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3452626</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9000013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8232333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5012128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 Bouganîm 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


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
“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
“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 Ropa et al. (2012).
CryptDB Encryption Scheme

- Multiple onion encryptions per column
- Example:

![Diagram showing CryptDB Encryption Scheme]

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

(1) SELECT name FROM Employees WHERE ID = 23;

CryptDB proxy

Need DET for C1-Eq, but it is at RND: adjust!

(2) UPDATE Table1 SET C1-Eq = DECRYPT_RND(K, C1-Eq, C1-IV);

(4) SELECT C2-IV, C2-Eq FROM Employees WHERE ID = xe243

DBMS

Table 1:

<table>
<thead>
<tr>
<th>C1-IV</th>
<th>C1-Eq</th>
<th>C1-Ord</th>
<th>C1-Add</th>
<th>C2-IV</th>
<th>C2-Eq</th>
</tr>
</thead>
<tbody>
<tr>
<td>x23bc</td>
<td>x72a1</td>
<td>x932b</td>
<td>xce10</td>
<td>x82d1</td>
<td>x52d8</td>
</tr>
<tr>
<td>x181e</td>
<td>x734a</td>
<td>x3e1b</td>
<td>x4210</td>
<td>xa130</td>
<td>xa163</td>
</tr>
</tbody>
</table>

(3) removing onion layer

Table 1:

<table>
<thead>
<tr>
<th>C1-IV</th>
<th>C1-Eq</th>
<th>C1-Ord</th>
<th>C1-Add</th>
<th>C2-IV</th>
<th>C2-Eq</th>
</tr>
</thead>
<tbody>
<tr>
<td>x23bc</td>
<td>xe243</td>
<td>x932b</td>
<td>xce10</td>
<td>x82d1</td>
<td>x52d8</td>
</tr>
<tr>
<td>x181e</td>
<td>x291a</td>
<td>x3e1b</td>
<td>x4210</td>
<td>xa130</td>
<td>xa163</td>
</tr>
</tbody>
</table>

(5) x82d1, x52d8

(6) results: 'Alice'

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?