A Novel Framework for Database Security based on Mixed Cryptography

Hasan Kadhem¹, Toshiyuki Amagasa¹,², Hiroyuki Kitagawa¹,²
¹Department of Computer Science, Graduate School of Systems and Information Engineering
²Center for Computational Sciences, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
hsalleh@kde.cs.tsukuba.ac.jp, {amagasa, kitagawa}@cs.tsukuba.ac.jp

Abstract

Database security has become a vital issue in modern Web applications. Critical business data in databases is an evident target for attack. Therefore, ensuring the confidentiality, privacy and integrity of data is a major issue for the security of database systems. Recent high profile data thefts have shown that perimeter defenses are insufficient to secure sensitive data.

This paper studies security of the databases shared between many parties from a cryptographic perspective. We propose Mixed Cryptography Database (MCDB), a novel framework to encrypt databases over untrusted networks in a mixed form using many keys owned by different parties. The encryption process is based on a new data classification according to the data owner. The proposed framework is very useful in strengthening the protection of sensitive data even if the database server is attacked at multiple points from the inside or outside.

1 Introduction

The World Wide Web has experienced massive growth in recent years. Individuals, businesses and governments have intensively used web applications that can provide effective, efficient and reliable solutions to the challenges of communicating, managing and directing commerce in the current century. However, these web-based applications have numerous entry points that can put databases at risk.

Recently, the number of reported data breaches involving sensitive private information at governmental, organizational and company levels has grown at an alarming rate. In some extreme cases, sensitive information belonging to millions of individuals has been revealed. For example, in May 2008, researchers at security vendors uncovered a server containing the sensitive email and Web-based data of thousands of people, including healthcare information, credit card numbers and business personnel documents and other sensitive data (www.searchsecurity.techtarget.com/news).

The true cost of such security breaches is manifold in itself. As mentioned by Cavusoglu et al. [2], security lapses can lead to the loss of consumer confidence and trust, over and above lost business and exposure to third-party liability.

Ensuring a suitable level of protection to database content affects the overall security model. Traditional techniques rely on access control, user authentication, intrusion detection and policies on how data is used to prevent such thefts and intrusion. Unfortunately, existing techniques cannot ensure that a database is fully immune to intrusion and unauthorized access and these mechanisms are ineffective against most inside attacks. Encryption is a well-studied technique to protect sensitive data so that when a database is compromised by an intruder, data remains protected even when a database is successfully attacked or stolen [5].

Even though encrypting the data provides important protection, there are implementation decisions that affect the encryption process. Who will encrypt data? Where will the data encryption be done? How is the data transferred? How are the encryption keys managed and protected? The answers to these questions affect data confidentiality, privacy and integrity.

We can see that there are three approaches to database servers where encryption takes place: first, the trusted database server where the creator, or owner, of the data operates a database server, which processes queries and signs the results; second, the untrusted server where the owner’s database is stored at the service provider (Database As a Service). The third and final model we call the semi-trusted server where the database is shared between many parties. Here, part of the data is stored as trusted while other parts are considered untrusted.

In traditional client server based encryption, the data is encrypted using either a server key in a trusted database or a client key in an untrusted database. Simply, if the data is encrypted using server key(s) and the administrator has the authority to use this key(s), then the whole system becomes vulnerable. The Database Administrator (DBA) or
any other employee who has access to the key(s) can access all data in plaintext. On the other hand, if each client encrypts his data using his key, then many problems might appear, such as how can organizations use the data? In both cases, the most important issue is the loss of the key used in the encryption process. A lost key means all the data is lost.

Recognizing the importance of encryption techniques, several database vendors offer an integrated solution that provides encryption functionality in their product. Those functions are used mostly in trusted servers. The weakness of this approach is that a user who has access to both the key table and the data table, and who can derive the key transformation algorithm, can break the encryption scheme.

The goal of this paper is to outline a cryptographic framework that supports the design of semi-trusted databases and provides protection for sensitive data even if the database server is attacked at multiple points by an inside or outside attacker.

1.1 System Model

We view a semi-trusted database system as shown in Figure 1. This system consists of three sides: client, Trusted Third Party (TTP) and server. The server side contains the encrypted databases and is responsible for replying to query requests on the encrypted databases. The client side makes query requests to retrieve, update, insert and delete data. The role of the trusted third party is to organize query requests and replies between clients and server(s).

This system could be used in many applications over the Web such as e-government, e-commerce and e-banking, where databases hold critical and sensitive information transferred over untrusted networks like the Internet. In e-government services, for example, the client might make query requests to many servers in different ministries and organizations. In this case, ensuring an appropriate level of protection for database content and for data transmitted over an untrusted network is, therefore, a fundamental part of any comprehensive security program.

Using data encryption, we propose a novel framework to ensure database security in the semi-trusted server scenario where the data in databases is shared by many parties. In this scenario, the data is encrypted in a mixed form using many keys owned by different parties. Our solution, called Mixed Cryptography Database (MCDB), is based on a new data classification that differs from traditional security based data classification. In our framework, data is classified based on data ownership and other criteria. Formally, each relation \( R_i \) over a database schema \( S \) in a plaintext database, \( R_i^E \) is the encrypted relation where the encryption is performed at the attribute level based on data classification using encryption function \( E^k \) (\( k \) is the key owned by a party). Also, we propose the Query Management System (QMS) to manage query requests executed over a mixed encrypted database and their query results. MCDB keeps the confidential data secure from (internal or external) intruders even if one key used in the encryption process has been compromised. Beyond that, data transmitted over an untrusted network is secured by more than one layer of encryption using different keys.

This paper discusses a hospital database as an illustrative example based on our framework. A malicious intruder trying to compromise patient records will be able to collect large amounts of data easily if these records are available electronically as plaintext. And if this information is encrypted using only server keys, it is an easy target for inside attack. Therefore, protection of a patient’s privacy is a basic requirement for ethical and legal use of information technology in health information systems; protecting confidential information related to hospitals is also crucial. In this scenario, many hospitals databases could be stored on one or many servers. Also, it is possible to have other related databases such as health insurance databases. The client makes query requests to many hospital servers, coordinated by a trusted third party.

1.2 Organization of Paper

This paper is organized as follows: Section 2 discusses related work. Section 3 addresses the problem of confidentiality, privacy and integrity, and then characterizes the basic attacks that can be conducted against semi-trusted databases. Section 4 discusses the proposed data classification and encryption process. Section 5 addresses query management issues. Section 6 analyzes security of data storage and data transmission. Finally, Section 7 concludes the paper and sketches future research directions.

2 Related Work

Much previous work is reported in literature that deals with database cryptography \([5, 13, 9, 3, 12]\). Previous relational database cryptography research studied the encryption and decryption process of fields within records using the Chinese Reminder Theorem (CRT) \([5]\), symmetric encryption \([9]\) or asymmetric encryption \([3]\). Other studies go farther theoretically to show how to integrate modern cryptography techniques into a Relational Database Management System (RDBMS) \([13, 12]\).
Davida et al [5] studied a cryptography system on the server side based on the Chinese Remainder Theorem (CRT). The system has sub-keys that allow the encryption and decryption of fields within records. Using symmetric encryption, Ge and Zdonik [9] propose a database encryption schema called FCE for column stores in data warehouses with a trusted server.

Recent years have seen extensive database cryptography research [1, 14, 16, 6, 4] conducted in the area of Database As a Service (DAS). The idea behind this research is to encrypt data so that it becomes accessible only to the client, because the data belongs to the client and is stored on an untrusted server. Bouganim and Pucheral [1] propose chipped secured data access solution C-SDA, which enforces data confidentiality and controls personal privileges with a client based security component acting as a mediator between the client and an encrypted database. This component is embedded in a smartcard to prevent tampering.

Another approach proposed by Mykletun and Tsudik [14] introduces a server coprocessor (SC), which is hardware with a processor, secure memory, an input device, a backup battery and a tamper proof container. An SC is installed on the server and supports cryptography operations. The client can communicate with the SC via server over a secure channel.

As discussed earlier, the database encryption operation in previous database cryptography research is done either by server or client key(s) using different algorithms. This paper introduces a new cryptography database framework that can deal with multi-tiered environments and can be used over the web. The new framework supports databases that are shared between many parties over untrusted networks by encrypting databases in a mixed form, so not only the client or server encrypts the databases.

3 The Problem

The increasing amount of sensitive data, especially personal data collected in databases, makes data confidentiality, privacy and integrity primary issues vital for governments, organizations and companies, and raises the risk of data theft.

Using server-based encryption or client-based encryption is not sufficient to encrypt semi-trusted databases where data is shared among many parties. Here, the problem is to ensure that confidentiality, privacy and integrity are achieved for a semi-trusted database where data is exchanged over an untrusted network such as the Internet. Consequently, we must protect databases from unauthorized disclosures and unauthorized modifications that may occur from inside and outside attacks. In our example, a hospital database, security of highly sensitive data must be ensured.

This section starts by introducing three security principles. It then illustrates the classification of attacks that are commonly directed against databases on the web.

3.1 Security Principles

3.1.1 Data Confidentiality

Data confidentiality is a major concern in database systems. There are many definitions of confidentiality in different organizations. According to ISO/IEC [11], to protect and preserve the confidentiality of information means to ensure that it is not made available or disclosed to unauthorized entities.

Hammer and Schneider [10] defined confidentiality as the concept of: 1) ensuring that information is accessible for reading, listening, recording or physical removal only to subjects entitled to it, and 2) that subjects only read or listen to information to the extent permitted.

Generally, data confidentiality is the protection of private information from surveillance or leaks when it is stored, or is transmitted across vulnerable networks such as the Internet. Actually, the confidentiality of data varies, depending on organization type and data classification. As a consequence, a particular piece of data could be confidential in one organization, but may not be confidential in another. In semi-trusted databases, each party needs to ensure confidentiality of its data even from other parties who share the same database.

3.1.2 Data Privacy

Society is experiencing exponential growth in the amount and variety of information collected on individuals. It is not surprising that organizations collect more personal-specific information than ever before, and often do so without any particular purpose [15].

From a user’s perspective, privacy is the right to secure sensitive personal data in digital form to protect it against fraud, identity theft or unauthorized use. Data privacy is the prevention of confidential or personal information from being viewed by parties and the control over its collection, use and distribution. Privacy differs from confidentiality in the sense that the data to be protected is personal.

A special case of data confidentiality is data privacy [1]. The importance of data privacy is to make a trusted relationship between the client and company or organization that collects and stores their personal information. In our hospital database example, privacy means to protect the patient’s personal information from being viewed or changed by unauthorized parties.

3.1.3 Data Integrity

Data integrity is the assurance that unauthorized parties are prevented from changing data. Integrity includes the protec-
tion of information during storage, transition, manipulation and data backup. Maintaining data integrity is essential to the privacy, security and reliability of data.

3.2 Class of Attackers

Before discussing the principles of the proposed Mixed Cryptography Database (MCDB), we first identify two basic classes of attackers that can break data confidentiality, privacy and integrity in semi-trusted databases:

- Outside attackers: An outside attacker is a person who may access the database directly or indirectly with the objective of achieving personal gain or causing harm to data. This is the commonly accepted ecracker attack from outside trying to commit fraud or interruption.

- Inside attackers: We classify inside attackers into two subclasses. Inside attackers in the first class are individuals with privileged access to the database; those who may misuse that position to create false transactions or interfere with legitimate transactions. This is the commonly accepted insider attack from other parts of the database server side not directly connected with the database. The attackers in the second class are individuals with privileged access to, or management responsibility for, the database, who may misuse those rights to interfere with or exploit the database. This is the generally accepted insider attack from operational staff responsible for the database system. An example is DBA use of privileged capabilities to alter records, create false accounting trails or create phantom users.

4 MCDB Baselines

The data exchange scenario comprising the setting for this paper is shown in Figure 1. It may include many database servers and many clients with an intermediate party. The database servers host primary data, and the intermediate party joins and coordinates between servers and clients. Database servers host what is owned by several parties, such as clients and intermediate parties, i.e., they contain data related to clients, servers and intermediate parties. The intermediate party could be a trusted third party or a public organization that coordinates many servers. Before discussing the encryption operation in our framework, we first illustrate data classification based on data owner. We then propose a mixed cryptography database based on that classification of data.

4.1 Data Classification

Data classification is the process of placing data categories that will reflect the security controls to protect that data against theft and inappropriate use. Traditional security-based data classification is the conscious decision to assign a level of sensitivity to data as it is being created, amended, enhanced, stored or transmitted.

We introduce new data classifications according to the data owner. Large databases contain many relations, and to ensure confidentiality, privacy and integrity, our data classification in the three tiered environment relies on who owns the data. Each side is responsible for encrypting the data owned. According to the framework, we classify data into three groups.

- Client data: data that belongs to the client. The client is responsible for encrypting this data to ensure privacy. This data directly reveals identity, such as name.

- Trusted third party data: data used to join many databases on one or many servers such as ID number. This data also includes data that may be used by servers without client permission. This data could indirectly reveal identity.

- Server data: data that belongs to an organization or company, which is responsible for collecting and maintaining it. This data could be used in statistical operations. The organization may apply traditional data classification on that data, so that some data will be encrypted and other data will be in plaintext.

The data classification of our hospital database example is shown in Figure 2. We chose an example of health information to show how our framework will satisfy data confidentiality, privacy and integrity over important and essential information sectors. While health information has come into use by many organizations and individuals, there is a need to have regulatory protections for this highly sensitive and deeply personal information. We applied our data classification method to the hospital database. The data is classified into three classes: patient data, hospital data and trusted third party data. In addition, there is unencrypted data that is considered as insensitive, such as date.

4.2 Database Encryption

In most databases used on the Web, data is stored in tables in the form they are loaded, mostly in plaintext, which does not satisfy high level security and privacy requirements. Several approaches exist to implement the encrypted operation in the database, as described in the related work section. Our proposed system, whose basic architecture is shown in Figure 1, adds encryption/decryption layers accumulatively while data moves from/to the database. The purpose of such design is to implement encrypted storage satisfying data confidentiality, privacy and integrity. In addition, it makes data usable for the server side, and makes it
The encryption operation in our system is based on data classification as described in the previous section. Such classification is needed to draw a full map of the encryption process. Following the convention, $E$ denotes the encryption function, and $D$ denotes the decryption function. The general approach to encryption is the field level; that is, sensitive fields need to be encrypted to protect the information from inside or outside attack. Data is encrypted using a symmetric encryption algorithm. After the data classification process, each side encrypts the data it owns.

**Definition 1:** For each database schema $S(R_1, ..., R_n)$, where $R_1$ to $R_n$ are relations created on the database server, $R_i$ name is encrypted using the server secret key $SE_S$.

**Definition 2:** For each relation schema $R(F_1, ..., F_i, F_{i+1}, ..., F_m, F_{m+1}, ..., F_n)$ in a relational database, where $F_l (1 \leq i < l)$ field is classified as trusted third party data, $F_j (l \leq j < m)$ field is classified as client data, and $F_k (m \leq k < n)$ field is classified as server data, we store an encrypted relation: $R_E^C = (F_1^C, ..., F_i^C, F_{i+1}^C, ..., F_m^C, F_{m+1}^C, ..., F_n^C)$, where, $F_i^C (1 \leq i < l)$ in the encrypted relation $R_E$ stores the encrypted value of $F_i$ in relation $R$ using trusted third party secret key $SE_T$, viz. $F_i^C = E^{SE_T}(F_i)$. $F_j^C (l \leq j < m)$ in the encrypted relation $R_E$ stores the encrypted value of $F_j$ in relation $R$ using trusted client secret key $SE_C$, viz. $F_j^C = E^{SE_C}(F_j)$. $F_k^C (m \leq k < n)$ in the encrypted relation $R_E$ stores the encrypted value of $F_k$ in relation $R$ using server secret key $SE_S$, viz. $F_k^C = E^{SE_S}(F_k)$. In a real database, the fields may be in different order, but we used this order to simplify the notation. Field names are encrypted as follows: $F_i$ and $F_j$ names are encrypted using trusted third party secret key $SE_T$. $F_k$ name is encrypted using server secret key $SE_S$. With many clients, it is impossible to encrypt field names of client data using a client secret key. Therefore, field names of all data are classified as client data encrypted using the same key, which is the TTP secret key.

**Definition 3:** Metadata rules updated in client, trusted third party and server whenever a change happens in the relation schema $R$ on the server side.

In the hospital example, the database is encrypted based on the data classification proposed in this paper. Figure 2(a) shows an unencrypted hospital database and Figure 2(b) shows the encrypted database. After the data classification process, each party encrypts existing unencrypted data using his key simply by calling the SQL update statement. We assume that a patient encrypts his personal information using a secret key stored on a secure smart card. The patient may allow other specialists to view his health information. In this case, the patients smart card could be used by other users to retrieve or update data integrated with user access control and authorization rules. For example, doctors may have the authority to read and update specific records while a nurse does not have that authority. In addition, the hospital wants to keep its information secure and ensure confidentiality, so it encrypts data using the server secret key. While a trusted third party coordinates between many hospital servers and other related servers, the data used to join many databases on one or many servers is encrypted using the trusted third party secret key.

**5 Query Processing**

To ensure data confidentiality, privacy and integrity, we should take care of two security issues: secure data storage and secure data transmission. We already discussed the first issue in database encryption. The second issue is related to client query requests and database server replies. The secure transmission of data is well studied and well supported in today’s e-business. Most web browsers and web servers support SSL (Secure Socket Layer) [8] or TLS (Transport Layer Security) [7]. In the proposed system, data is transferred via network using one of the secure transport protocols. In addition, transmitted data is already encrypted using either symmetric encryption or asymmetric encryption. This section proposes a query processing system that addresses the secure data transmission issue.

In our framework, the client might send a query request $(Q)$ to many databases servers. The trusted third party coordinates the query requests and distributes them $(Q_1, ..., Q_n)$ to the responsible servers, then combines the query results.
SELECT ESEs(patient).ESEt(p_id), ESEs(patient).ESEt(p_fname), ESEs(patient).ESEt(p_lname), ESEs(drug).ESEs(d_id) and ESEs(drug).ESEs(d_id)= ESEs(patient).ESEs(d_id) and ESEs(patient).ESEt(p_id) = ESEt(1234)

before submitting the final result to the client. This paper discusses an example of a query request over one server, which is the fundamental component of query processing in the MCDB framework. Two components deal with queries in our framework. The first is Query Management Agent (QMA), which deals with query requests; the second is Result Analysis (RA), which deals with query results. The two components use metadata to perform the query process.

Table 1. Trusted Third Party (TTP) metadata in the hospital example.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Field</th>
<th>Type</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient_drug</td>
<td>p_id</td>
<td>Big integer</td>
<td>TTP</td>
</tr>
<tr>
<td>patient</td>
<td>p_id</td>
<td>Big integer</td>
<td>TTP</td>
</tr>
<tr>
<td>patient</td>
<td>p_fname</td>
<td>text</td>
<td>client</td>
</tr>
<tr>
<td>patient</td>
<td>p_lname</td>
<td>text</td>
<td>client</td>
</tr>
</tbody>
</table>

5.1 Metadata

Metadata contains the minimum data required to rewrite and analyze the query. Each party has different metadata according to the data classification. For example, a client needs data only on fields he owns while the trusted third party needs data on fields owned by the client or TTP. The data consists of field name, type, owner and table name. Table 1 shows an example of the TTP metadata used to perform algorithms in QMA and RA. That data is also used by TTP to encrypt existing data and schema.

5.2 Query Management Agent (QMA)

Figure 3 shows the query management system in MCDB. The first component in the query management system is the Query Management Agent (QMA). The main objective of QMA is to encrypt, decrypt and rewrite the query request. The structure of QMA is similar for the client, the trusted third party and the server, but the procedure differs. QMA prepares the query request to be executed over the encrypted database on the server side.

In the hospital database example, the query we discuss is a client who wants to retrieve all drugs taken by a patient. This query over the hospital database includes patient id, name and the drug names. Figure 3 shows execution of this query over mixed encrypted databases. The query request follows the sequence of algorithms in Figure 5.

At the end of the SQL query rewriting process, the translated SQL statement is executed over the encrypted database and returns the result to a temporary query result in server result analyzer RA. Figure 4 shows the query executed on the server side. $SE_T$ is the server secret key and $SE_T$ is the trusted third party secret key.

5.3 Result Analyzer (RA)

The second component in the query management system is the result analyzer (RA). The objective of RA is to decrypt the query result to make it readable for the client. In addition, RA uses client public key $(PU_C)$ to add an additional asymmetric encryption layer to the result before it is transmitted to other parties; in this way only the client can decrypt it using the client private key $(PR_C)$. The query result follows the sequence of algorithms in Figure 6.
Secure data transmission is performed by the secure transport protocol. In addition, personal information is
transmitted with either asymmetric or symmetric encryption using the client private or secret key. That means two layers of encryption are used. Still, the ciphertext could be modified by inside or outside attackers without knowledge of the keys, resulting in meaningful modification of the data when it is decrypted by the recipient. We can add a hash function or digital signatures to ensure integrity.

Case 1: When an intruder tries to decrypt a query result transmitted over an untrusted network, the probability will be $\frac{1}{2^{SPR^C}} \frac{1}{2^C}$, where $SPR^C$ is the size of the client private key, and $SC$ is the size of the client secret key. This means an intruder should first decrypt the result using the client asymmetric algorithm key, then encrypt the result using the client symmetric algorithm key. This procedure provides privacy for a client’s personal information that is transmitted over an untrusted network to ensure data integrity.

7 Conclusion and Future Work

The number of web applications involving sensitive private information at the governmental, organizational and company levels is growing rapidly. Preserving data confidentiality, privacy and integrity in the semi-trusted database context, where the database is shared between many parties, is becoming one of the most challenging issues for the database community. This paper addresses this issue and contributes the following. First, it introduces new data classification based on who owns the data. Second, it proposes a mixed cryptography database based on data classification methods. Third, it illustrates a query management system over a mixed encryption database. Finally, it analyzes the security of data storage and data transmission in the new cryptography framework.

There are other important research issues related to our framework: first, the best encryption algorithm used in the mixed cryptography database on performance and security perspectives; second, access control methods used to control access for all parties using the database; and finally indexing and joining between different databases. We will discuss these in future papers.

References


