Multilevel secure database encryption with subkeys *

Min-Shiang Hwang †
Email: m24@ms.tl.gov.tw

Wei-Pang Yang ‡
Fax: 886-35-721490

Department of Information Management †
Chao Yang Institute of Technology
Wufeng, Taiwan, R.O.C.

Department of Computer and Information Science ‡
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.

October 31, 2012
Abstract

In this paper, we propose a multilevel database encryption system with subkeys. This new system is called the record-oriented cryptosystem which encrypts each record with different field-subkeys according to a security class of the data element. Each field is decrypted individually by the field-subkeys of which security class is higher than or equal to that of the encrypted field-subkeys. This system is based on the Chinese Remainder Theorem. Our scheme can protect the finest level of granularity such as relation level, attribute level, tuple level, or data element level in the relational database model.

KeyWords: Chinese remainder theorem; Cryptography; Multilevel database; Data security; Subkeys

1 Introduction

Some of the advantages of using a database are the following [10, 31]: (1) shared access; (2) minimal redundancy; (3) data consistency; (4) data integrity, so that data values are protected against accidental or malicious unauthorized changes; and (5) controlled access, so that only authorized users are allowed to access data values. A database management system (DBMS) with security facility is designed to provide all of these advantages efficiently.

*This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC-85-2213-E009-029.
In general, there are four methods of enforcing database security [15]: First, physical security, such as storage medium safekeeping and fire protection [9]; second, operating system security, such as the use of an access control matrix, capability-list, and accessor-list [8, 17, 21]; third, DBMS security, such as protection mechanisms and query modification [16, 28, 36]; and fourth, data encryption, such as the data encryption standard (DES) [29, 34] and RSA scheme [32]. The first three methods, however, are not totally satisfactory in solving the database security problems, for three reasons. First, it is difficult to control the disclosure of raw data, because the raw data exists in readable form inside a database [11]. Second, it is invalid to prevent the disclosure of sensitive data, because the sensitive data must be backed up frequently in storage media in case of system failure or disk crash. Third, it is difficult to control the disclosure of confidential data in a distributed database system. A practical solution to the above problems is to use encryption methods to enforce database security [2, 3, 11, 14, 18, 19, 20, 38, 40]. An encryption database security can solve the above problems in the following manner: Data are encrypted into ciphertext, which only can be decrypted with the proper decryption key, thus eliminating the problem of data disclosure.

Database security methods based on encryption include database encryption systems with a single key [18] and database encryption systems with subkeys [11]. The first type of method needs a trustworthy centralized access control scheme with which to control all access to data stored in the database system (DBS). All encryption and decryption are executed by the trusted access control scheme with private keys. In the second type of method, however, decryption
is executed by users themselves with their own subkeys.

A database system with subkeys has the following advantages over conventional systems. First, each encrypted record is a single encrypted value which is a function of all fields, so the system is record-oriented. Obviously, a small change in the encrypted value will cause a significant change in the decrypted value. Therefore, unauthorized modification of data can be prevented. Second, the system’s properties can withstand pattern matching attacks. Third, the possibility of substitution attacks is eliminated because the system encrypts all fields together. Finally, a user can read only some of the field data objects, depending on the reading field-subkey he has. Not all fields need to be available to everyone.

A single-level database encryption/decryption system with subkeys has been proposed by Davida et al. [11]. This system is called the record-oriented cryptosystem which encrypts each record with field-subkeys and decrypts individually each field by these single-level field-subkeys. In this paper, we propose a multilevel database encryption/decryption system with subkeys.

This multilevel databases system is a partially-ordered hierarchy as shown in Figure 1. Each subject (e.g., user, program, processor, etc.) is given a distinct clearance and each object (e.g., a file, a message, data, etc.) is assigned a security level. Subjects and objects are classified into a number of distinct security classes $S_1, S_2, \cdots, S_m$ [24, 35]. In such a hierarchy, an object with a particular security class can be accessed only by subjects in the same or a higher security class [1, 5, 33].

This new system encrypts each record with different field-subkeys according
to the security class of the data element and each field is decrypted individually by the field-subkeys of which the security class is higher than or equal to that of the encrypted field-subkeys. Our system is based on the Chinese Remainder Theorem (CRT). Our scheme can protect the finest level of granularity such as relation level, attribute level, tuple level, or data element level in the relational database model.

Using the CRT, the subkey scheme has the following merit: The raw field data can be easily recovered within only one operation. The CRT has been used widely in security control, such as in access control schemes [23], in secure broadcasting schemes [6], in identification and authentication schemes [4], in database encryptions [11], and in public-key cryptosystems [27].

The paper is organized as follows. In Section 2, we review and develop a single-level database system with subkeys. In Section 3, we propose an encryption scheme for multilevel database security. We analyze the security and computational complexity of our scheme in Section 4 and Section 5, respectively. Section 6 and 7, we propose several algorithms for relational algebra and dynamic ability. Section 8 is the conclusion of this paper.
2 Single-level database encryption schemes with subkeys

A single-level database encryption/decryption system with subkeys was proposed by Davida, et al. in 1981. Their system was based on the Chinese remainder theorem [30]. Let \( C \) be the ciphertext of an encrypted record, \( m_i \) be the value of the \( i \)th field of a record, \( r_i \) be the random number generated for field \( i \), \( e_i \) be the encryption key for field \( i \) and there be \( n \) fields in each record of the database. The encryption procedure is done by forming

\[
C = \sum_{i=1}^{n} e_i (r_i \| m_i) \mod N
\]

where \( N = \prod_{i=1}^{n} k_i \); \( k_i \) is the decryption key for field \( i \); \( \| \) indicates a concatenation; \( (r_i \| m_i) \leq k_i \); and \( e_i = (N/k_i)b_i \) where \( b_i \) is the multiplicative inverse of \( N/k_i \) modulo \( k_i \). Decryption can be done as follows

\[
r_i \| m_i = C \mod k_i, \quad i = 1, \cdots, n.
\]

By discarding the random bit \( r_i \), one can get the \( i \)-th field value \( m_i \).

In order to prevent known-plaintext attacks, Davida, Wells, and Kam [11] concatenate a random redundancy value \( r_i \) in each field (the length of the redundancy value \( r_i \) is at least 32 bits, which leads to better security.). Therefore their scheme needs extra spaces to store the raw data. We proposed a two-phase encryption scheme in [22] for enhancing database security. Phase 1 encrypts the data in each field with one-way function. Phase 2 encrypts the encrypted data based on the Chinese remainder theorem.

We briefly describe the two-phase encryption algorithm as follows. To illustrate the scheme, we assume that there are \( n \) fields in each record of a database.
Let \( m_1, m_2, \ldots, m_n \) be the \( n \) raw data of fields of a record.

**Phase E1:** Encrypt \( m_i \), for \( i = 1, \ldots, n \). Let \( f \) be the encryption algorithm and \( d_i \) be a secret key of the algorithm of field \( i \). This encryption is done as \( f_{d_i}(m_i) \).

**Phase E2:** Encrypt \( f_{d_i}(m_i) \) with writing subkeys \( e_1, e_2, \ldots, e_n \). This encryption is done as

\[
C = E((f_{d_1}(m_1), e_1), (f_{d_2}(m_2), e_2), \ldots, (f_{d_n}(m_n), e_n)),
\]

where \( E \) is an encryption algorithm, \( e_i \) is a writing key for field \( i \), and \( C \) is the encrypted data of a record. With the Chinese remainder theorem, the encryption procedure is the following:

\[
C = \sum_{i=1}^{n} e_i f_{d_i}(m_i) \mod N.
\]

The decryption procedure is the reverse of the encryption procedure:

**Phase D1:** Decrypt ciphertext \( C \) with reading subkeys \( k_1, k_2, \ldots, k_n \). The decryption is done as

\[
f_{d_i}(m_i) = D(C, k_i),
\]

where \( D \) is a decryption algorithm which is based on the Chinese remainder theorem and \( k_i \) is a reading key for field \( i \). The decryption procedure is as follows.

\[
f_{d_i}(m_i) = C \mod k_i.
\]

**Phase D2:** Decrypt \( f_{d_i}(m_i) = m'_i \) with the secret key \( d_i \) as follows:

\[
m_i = f_{d_i}^{-1}(m'_i),
\]
We now propose a new encryption scheme for multilevel database security. To illustrate the scheme, we assume that there are \( n \) fields in each record of a database. Each field \( i \) has a security hierarchy \( H_i \). Each atomic has a security class. Let \( m_1, m_2, \ldots, m_n \) be the \( n \) raw data fields of a record associated with the security class \( s_{1x}, s_{2y}, \ldots, s_{nz} \) as shown in Figure 2. Here, \( s_{ij} \in H_i \) denotes the \( j \)th security class in \( H_i \). \( A_i \) is attribute name and \( L_i \) is type of security class which corresponding to \( A_i \).

Let \( k_{ij} \) be the decryption key for the security class \( s_{ij} \). All \( k_{ij} \) are pairwise relatively prime integers. Essentially, the encryption process is to convert the field values of a record into a ciphertext form, say \( C \), and later we can recover it to the original raw field values by using the decryption key. This encryption is done by the following equation

\[
C = \sum_{i=1}^{n} e_i m_i \mod N,
\]

where \( N = k_{1x} \cdot k_{2y} \cdots k_{nz} \). Each field value \( m_i \) thus can be decrypted by the equation

\[
C \mod k_{ij}
\]
for a modulus $k_{il}$ of a security class $s_{il} \geq s_{ij}$.

We employ the following two theorems to show that Equation (8) and (9) are correct.

**Theorem 3.1 (Chinese Remainder Theorem) [12]**

Let $k_{1x}, k_{2y}, \ldots, k_{nz}$ be pairwise relatively prime integers and let $N = k_{1x}k_{2y}\cdots k_{nz}$, then there exists

$$C = \sum_{i=1}^{n} e_i m_i \mod N.$$  \hfill (10)

$C$ is the smallest constant such that

$$C \mod k_{ij} = m_i, \quad i = 1, \ldots, n; j = x, y, \ldots, z.$$  \hfill (11)

**Theorem 3.2** If Equation (11) holds and $k_{ij}$ can be divided by $k$ then $C \mod k = m_i$ when $m_i < k$.

**Proof.** Since $C \mod k_{ij} = m_i$, $C = ak_{ij} + m_i$, where $a$ is an integer. We have $C \mod k = m_i$. \hfill $\Box$

According to Theorem 3.1 and 3.2, we can construct two cryptographic key generation schemes for access control in a totally-ordered hierarchy and a partially-ordered hierarchy, respectively. The algorithm for generating the secret key of security class for each hierarchy $H_i$ is stated as follows.

*Algorithm Key-Generation for Totally-Ordered Hierarchy.*
Step 1: Get a node $s_{ij}$ from the hierarchy $H_i$ by preorder traversal.

Step 2: Assigns $s_{ij}$ a large randomly prime $p_{ij}$.

Step 3: Computes the secure key $k_{ij}$ for $s_{ij}$ as follows.

$$k_{ij} = \prod_{s_{il} \geq s_{ij}} p_{il}. \quad (12)$$

Step 4: Repeat from Step 1 until all nodes of the hierarchy $H_i$ are completely examined.

Algorithm Key-Generation for Partially-Ordered Hierarchy.

Step 1: Get a node $s_{ij}$ from the hierarchy $H_i$ by preorder traversal.

Step 2: Assigns $s_{ij}$ two large randomly primes $p_{ij}$ and $p'_{ij}$.

Step 3: Computes the secure key $k_{ij}$ for $s_{ij}$ as follows.

Step 3.1: If $s_{ij}$ is a root node, then $k_{ij} = p_{ij}$.

Step 3.2: If $s_{ij}$ is not a root node, then

$$k_{ij} = \prod_{s_{il} \geq s_{ij}} p_{il} \prod_{s_{il} > s_{ij}} p'_{il}. \quad (13)$$

Step 4: Repeat from Step 1 until all nodes of the hierarchy $H_i$ are completely examined.

An illustrative example for generating secret key $k_{ij}$ for each security class by the Algorithm Key-Generation is shown in Figure 3.

The following example illustrates the encryption and decryption of the proposed scheme.

Example 3.1 Assume that there are 3 fields in each record of a database and 2 security levels (top-secret and secret). Let $(4, s_{12}), (10, s_{21}), (15, s_{32})$ be three atomics of a record $R$. Here $s_{ij}$ is the $j$th security level of the $i$th field. Let
Figure 3: An example of generating secret key for each security class in (a) totally-ordered hierarchy $H_i$, (b) partially-ordered hierarchy $H_i$.

$(p_{11}, p_{12}) = (5, 7), (p_{21}, p_{22}) = (11, 13), (p_{31}, p_{32}) = (17, 19)$. By the algorithm Key-Generation for Totally-Ordered Hierarchy, we can compute the secret keys $(k_{ij})$ of the three fields as follows.

\[
\begin{align*}
    k_{s_{12}} &= p_{11}p_{12} = 5 \times 7 \\
    k_{s_{21}} &= p_{21} = 11 \\
    k_{s_{32}} &= p_{31}p_{32} = 17 \times 19
\end{align*}
\]

By the Theorem 3.1 and 3.2, we obtain $N = k_{s_{12}}k_{s_{21}}k_{s_{32}} = 124355$. The writing key $e_i, e_i = (N/k_{ij})b_i$ where $b_i$ is the multiplicative inverse of $(N/k_{ij} \mod k_{ij})$, can be computed ($e_1 = 7106, e_2 = 79135, e_3 = 38115$). Finally, we compute the ciphertext of the record as follows:

\[
\begin{align*}
    C &= (7106 \times 4 + 79135 \times 10 + 38115 \times 15) \mod 124355 \\
    &= 23594.
\end{align*}
\]
When a user wants to read the message of the ith field, the user decrypts the ciphertext using the corresponding decryption key of the ith field.

To read field 1:
\[ 23594 \mod 5 \times 7 \]
\[ = 23594 \mod 5 \]
\[ = 4 \]

To read field 2:
\[ 23594 \mod 11 \]
\[ = 10 \]

To read field 3:
\[ 23594 \mod 17 \times 19 \]
\[ = 23594 \mod 17 \]
\[ = 15 \]

4 Cryptanalysis

There are some ways to challenge the security of the scheme using Chinese remainder theorem [39].

1. It cannot withstand known-plaintext attacks. Let \( C \) and \( C' \) be the ciphertext of two different records \( R \) and \( R' \), respectively. If \( m_i \) and \( m'_i \) are the raw data of field \( i \) in \( R \) and \( R' \), respectively, and both are known to a
cryptanalyst, then from Equation (9) we have

\[ C - m_i = a_1 k_{ij}, \]
\[ C' - m'_i = a_2 k_{ij}, \]

where \( a_1 \) and \( a_2 \) are an integer. The subkey \( k_{ij} \) thus can be derived from the above two equations using the greatest common divisor. \( a_1 \) and \( a_2 \) may have a common divisor. In this case, we derive \( k_{ij} \) using more raw data such that all \( a_i \) are pairwise relatively prime.

2. The following strategy can also be used to attack the scheme. Let \( C_r \) be the \( r \)th encrypted record and \( m_i \) be the \( i \)th field raw data of the \( r \)th record. Thus, there exists an integer \( a_3 \) in the system such that

\[ C_r = a_3 k_{ij} + m_i. \]

Assume that a field other than \( i \) is updated, then

\[ C'_r = a_4 k_{ij} + m_i. \]

Since \( m_i \) is not changed, then

\[ C_r - C'_r = C''_r = (a_3 - a_4) k_{ij}. \]

If a similar operation is performed on another encrypted record \( C_h \), then

\[ C_h - C'_h = C''_h = (a'_3 - a'_4) k_{ij}. \]

The subkey \( k_{ij} \) can then be computed by finding the gcd\((C''_r, C''_h)\).

3. The scheme cannot withstand collusion attacks. All users who have read capability only can, together, compute the writing key \( e_i \), which is known only by the system, if they have all of the reading keys \( k_{ij} \).
Now let us see the two-phase encryption scheme can withstand the known-plaintext attack. From Equation (6) we have

\begin{align}
C - f_{kj}(m_i) &= a_1k_{ij} \\
C' - f_{kj}(m'_i) &= a_2k_{ij}.
\end{align}

The above simultaneous equations have three unknown variables, \( f_{kj}(m_i) \), \( f_{kj}(m'_i) \), and \( k_{ij} \). Hence, there are infinite possible solutions for \( k_{ij} \). In general, if \( t \) corresponding fields of \( t \) records are known, there are \( t + 1 \) unknown variables to be determined with \( t \) simultaneous equations. Hence it will be much more difficult to mount a known-plaintext attack against our scheme.

The security of our scheme depends on the one-way function in addition to the subkey scheme. Illegal users cannot read the raw data of a tuple unless they know both the reading subkey and the secret key of the encryption algorithm. Thus security is guaranteed in our scheme to eliminate the second weakness.

In order to eliminate the third weakness in a read-only environment, we add a dummy field in relation tables. Since the writing key for field \( i \), \( e_i \), is equal to \((N/k_{ij})b_i\), \( e_i \) can be obtained if we know all the \( k_{ij} \)'s. However, any user does not know the secret key of the dummy field. Therefore, our scheme can withstand the collusion attacks.

The other security issue to consider is that cryptosystems can withstand timing attacks [26]. Since ciphertext is an encrypted record with many field-messages in our scheme, attackers need many timing measurements to cryptanalyze our scheme using timing attacks.

Another security issue to consider is that a security class \( s_{ij} \) should not be able to derive the secret key of the other security classes \( s_{il} \), using its own
cryptographic key for $s_{ij} \leq s_{il}$. The scheme must also provide security against two or more security classes collaborating to derive a higher level key. In the following, we prove that our method is secure against such derivation.

**Theorem 4.1** The security of the Algorithm Key-Generation for a totally-ordered hierarchy is equivalent to factoring a large composed prime.

**Proof.** We divide the proof into the following two cases. 

**Case 1:** it is trivial to show that if a large composed prime can be factored, the secret key $k_{il}$ can be derived by $s_{ij}$ where $s_{ij} < s_{il}$.

**Case 2:** if the secret key $k_{ih}$ can be derived by $s_{ij}$ where $s_{ij} < s_{ih}$, a large composed prime can be factored. From step 3 of algorithm Key-Generation for Totally-ordered Hierarchy in Section 3 we know that

$$k_{ij} = \prod_{s_{il} \geq s_{ij}} p_{il}.$$ 

Since $k_{ij}/k_{ih} = \prod_{s_{ih} > s_{il} \geq s_{ij}} p_{il}$, this case thus holds. \qed

The security of the Algorithm Key-Generation for Partially-Ordered Hierarchy is also equivalent to factoring a large composed prime. The proof is similar to that for Theorem 4.1.

Next, we show that our scheme is correct in the following.

**Theorem 4.2** The proposed scheme satisfies $s_{ij} \leq s_{il}$ if and only if the encrypted data $C$ under $k_{ij}$ can be decrypted under $k_{il}$, where $k_{il}$ and $k_{ij}$ are the secret keys of $s_{il}$ and $s_{ij}$, respectively.

**Proof.** We divide the proof into the following two cases.

**Case 1:** if $s_{ij} \leq s_{il}$ then $C$ under $k_{ij}$ can be decrypted under $k_{il}$. This case
holds by the Theorem 3.2.

**Case 2:** if $C$ under $k_{ij}$ can be decrypted under $k_{il}$ then $s_{ij} \leq s_{il}$. This case is equivalent to stating that if $s_{ij} \not< s_{il}$ then $C \mod k_{ij} = m_i$ and $C \mod k_{il} \neq m_i$. If $C \mod k_{il} = m_i$, implies $k_{ij}|k_{il}$. However, From step 3 of algorithm Key-Generation for Totally-ordered Hierarchy and step 3.2 of algorithm Key-Generation for Partially-Ordered Hierarchy in Section 3 we know that

$$k_{ij} = k_{il}p', \text{ for } s_{ij} \leq s_{il}$$

where $p'$ is relatively prime with $k_{il}$. By the Theorem 4.1, this case thus holds. □

5 Computational and storage space complexity

In this section, we examine storage space and computational complexity of enciphering and deciphering each field. Assume that each record contains $n$ fields; the number of bits of each field is $b$ on the average; there are total $l$ security classes in a relation table. The computation time needed for each record in Section 3 is as follows.

Encryption Equation (8) requires a total of $2n$ multiplications, $(n-1)$ additions, $n$ divisions, and one module operation. Let $t_{op}(a, b)$ denote the time cost of an "op" operation (i.e., multiplication, division, addition, or module) with two bits $a$ and $b$.

$$t_{encryption} = 2nt_{multiplication(nbl, b)} + (n-1)t_{addition(nbl, nbl)} +$$

$$nt_{division(nbl, b)} + t_{module(nbl, nbl)},$$

$$= 2n^2t_{multiplication(b, b)} + n(n-1)t_{addition(b, b)} +$$

15
\[ nt_{\text{division}}(nbl, b) + t_{\text{module}}(nbl, nbl). \]

Decryption Equation (9) requires only one module operation:

\[ t_{\text{decryption}} = t_{\text{module}}(nbl, b). \]

Some efficient implementations of the Chinese remainder theorem have been developed [13, 25, 37]. Dirr and Taylor [13] have designed a fast and efficient hardware implementation of the Chinese remainder theorem in residue arithmetic. Their method incurs a time cost of \( 70\lceil \log_2 L \rceil \) ns for computing the equation \( C = m_i \mod k_{ij}, \) for \( i = 1, 2, \cdots, L. \) It only needs 3.5 ms to encipher a large database with 32 fields, 1000 records, and 10 security levels. Thus, our subkey scheme is practical to implement.

Next we discuss the storage space of the scheme. Our scheme encodes each field \( m_i \) of a record as a number modulo a number \( k_{ij} \) of the form described in Equation (10). Assume that there are \( n \) fields in relation table, an average of \( b \) bits in each field, and \( l \) security classes for a hierarchy. The total number of bits in each record is \( nbl. \) Although the scheme does some data expansion, it is somehow demanded for enhancing database security.

6 Cryptographic relational algebra

In this section, we show how to perform the relational operations in our scheme. Codd defined a very specific set of eight operations: restrict, project, Cartesian product, union, intersection, difference, natural join, and division. Basically, only the first five primitive operations are needed; the other operations can be derived from these five [?]. For example, natural join is a projection of
Table 1: Algorithm of projecting $i$th field.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Ciphertext $C_g$, $g = 1, \ldots, h$, where $h$ is the number of records in the database. Read subkey $k_{ij}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>The raw data $m_{ig}$ in the $i$th field of the $g$th record.</td>
</tr>
</tbody>
</table>

1. for $g = 1, \ldots, h$ do
2. $m_{ig} = C_g \mod k_{ij}$;

a restriction of a product, intersection is a difference twice, and division is the difference of a product of a difference. Thus, we shall treat only the five primitive operations.

Since our scheme is a so-balled record-oriented (tuple-oriented) subkey scheme, it is easy to see that the restrict, union, intersection, and difference are the same as in a traditional database. By the Chinese remainder theorem, we develop two algorithms for projection and production, as shown in Table 1 and Table 2, respectively. In Table 1, we only project the $i$th field. By iteration, other fields can also be projected.

View is an important mechanism in relational database model. A view is a table that does not have any existence in its own right, but is instead derived from one or more underlying base tables [10]. We develop an algorithm for view mechanism, as shown in Table 3. Step 3 in Table 3, $C'_g = C_g \mod N'$, can be proved to be correct as follows:

$$
C'_g \mod k_{ij},
= (C_g \mod N') \mod k_{ij},
= C_g \mod k_{ij},
$$
Table 2: Algorithm for Cartesian production.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Ciphertext $C'_g$, $g = 1, \cdots, h'$, where $h'$ is the number of records in a relation table $R'$.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ciphertext $C''_g$, $g' = 1, \cdots, h''$, where $h''$ is the number of records in a relation table $R''$.</td>
</tr>
<tr>
<td></td>
<td>Read field subkeys $k'_{ij}$, $i = 1, \cdots, n'$, where $n'$ is the number of fields in a relation table $R'$.</td>
</tr>
<tr>
<td></td>
<td>Read field subkeys $k''<em>{ij}$, $i = 1, \cdots, n''$ in a relation table $R''$, where $k''</em>{ij} \neq k'_{ij}$ for all $i$ and $j$.</td>
</tr>
<tr>
<td>Output:</td>
<td>New relation table $R$.</td>
</tr>
</tbody>
</table>

1. Compute $N_1 = \prod_{i=1}^{n'} k'_{ij}$
2. Compute $N_2 = \prod_{i=1}^{n''} k''_{ij}$ */ Computing the ciphertext by the Chinese remainder theorem */
3. Compute $N = N_1 \times N_2$
4. for $g = 1, 2$ do
5. begin
6. Compute $G_g = N/N_g$;
7. Find $G'_g$ such that $G_g G'_g \mod N_g = 1$;
8. end;
*/ Computes new ciphertext record */
9. for $g = 1, \cdots, h'$ do
10. for $g' = 1, \cdots, h''$ do
11. $C_{(g-1)h''+g'} \leftarrow (C'_g G_1 G'_1 + C''_g G_2 G'_2) \mod N$;

Table 3: Algorithm for view mechanism.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Ciphertext $C'_g$, $g = 1, \cdots, h$, where $h$ is the number of records in the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read subkeys $k_{ij}$ for some fields $i$.</td>
</tr>
<tr>
<td>Output:</td>
<td>New encrypted record data $C''_g$ in the view</td>
</tr>
</tbody>
</table>

1. Compute $N' = \prod_{i} k_{ij}$;
2. for $g = 1, \cdots, h$ do
3. $C''_g = C'_g \mod N'$;


7 Dynamic ability

In the following subsections we give algorithms for inserting a new field, updating a data element, and removing a field in the relation table.

7.1 Inserting a new field

When inserting a new field to the relation table, we compute the encrypted data of records of the form described in Equation (10). The algorithm for inserting a new field to the relation table is given in Table 4.

Step 8 in Table 4, \( C'_g = (C_g G_1 G'_1 + m'_{i'g} G_2 G'_2) \mod N' \), can be proved to be correct as follows:

\[
C'_g \mod k_{i'j}, \text{ for the new field } i',
\]
\[
= ((C_g G_1 G'_1 + m'_{i'g} G_2 G'_2) \mod N') \mod k_{i'j},
\]
\[
= m'_{i'g} G_2 G'_2 \mod k_{i'j},
\]
\[
= m'_{i'g}.
\]

And

\[
C'_g \mod k_{ij}, \text{ for some existing field } i,
\]
\[
= ((C_g G_1 G'_1 + m'_{i'g} G_2 G'_2) \mod N') \mod k_{ij},
\]
\[
= C_g G_1 G'_1 \mod k_{ij},
\]
\[
= m_{ig}.
\]

7.2 Updating a data element

When the \( i \)th field raw data of the \( g \)th record \( (m_{ig}) \) is updated into \( m'_{ig} \), we compute the new encrypted data of records from the old \( C_g \) according to the
Table 4: Algorithm for inserting a new field to the relation table.

<table>
<thead>
<tr>
<th>Input:</th>
<th>Ciphertext $C_g$, $g = 1, \cdots, h$, where $h$ is the number of records in a relation table $R$.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing read subkeys $k_{ij}$, $i = 1, \cdots, n$.</td>
</tr>
<tr>
<td></td>
<td>New read subkey $k_{i'j}$, for a new field.</td>
</tr>
<tr>
<td></td>
<td>New raw data $m_{i'g}$, $g = 1, \cdots, h$, for the new field $i'$ of the $g$th record.</td>
</tr>
<tr>
<td>Output:</td>
<td>Ciphertext $C'_g$, $g = 1, \cdots, h$.</td>
</tr>
</tbody>
</table>

1. Compute $N = \prod_{i=1}^{n} k_{ij}$
2. Compute $N' = N \times k_{i'j}$
3. Compute $G_1 = N'/N$;
4. Find $G'_1$ such that $G_1 G'_1 \mod N = 1$;
5. Compute $G_2 = N'/k_{i'j}$;
6. Find $G'_2$ such that $G_2 G'_2 \mod k_{i'j} = 1$;
7. for $g = 1, \cdots, h$ do
8. $C'_g \leftarrow (C_g G'_1 G_1 + m_{i'g} G_2 G'_2) \mod N'$;

following equation.

$$C'_g = [C_g + (m'_{ig} - m_{ig}) G_i G'_i] \mod N'.$$ (15)

The algorithm for updating a data element in the relation table is given in Table 5.

7.3 Removing a field

By the property of Chinese remainder theorem, a field can be arbitrarily deleted from the relation table. The removal will not affect the previously discussed actions.

8 Conclusions

We have proposed a multilevel database encryption system with subkeys. The system has the following four important advantages.
Table 5: Algorithm for updating a data element.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Compute $N = \prod_{i=1}^{n} k_{ij}$</td>
</tr>
<tr>
<td>2.</td>
<td>Compute $G_i = N/k_{ij}$;</td>
</tr>
<tr>
<td>3.</td>
<td>Find $G'_i$ such that $G_iG'_i \mod N = 1$;</td>
</tr>
<tr>
<td>4.</td>
<td>Computes new ciphertext record */</td>
</tr>
<tr>
<td></td>
<td>$C'<em>g \leftarrow (C_g + (m'</em>{ig} - m_{ig})G_iG'_i) \mod N.$;</td>
</tr>
</tbody>
</table>

1. It allows the finest level of granularity to be protected such as relation level, attribute level, tuple level, or data element level in the relational database model.

2. It allows the encryption of fields with different security class, but the decryption is permitted only in the security class higher than or equal to that of the encrypted field-subkeys.

3. It allows the encryption/decryption of fields within a record.

4. The security of our scheme is equivalent to factoring a large composed prime.

**Acknowledgements**

The authors wish to thank many anonymous referees for their suggestions to improve this paper. Part of this research was supported by the National Science Council, Taiwan, R.O.C., under contract no. NSC85-2213-E-009-029.
References


