A Multilevel Encryption Scheme for Database Security

Thomas Hardjono and Jennifer Seberry

Department of Computer Science
University College
The University of New South Wales
Australian Defence Force Academy
Canberra, A.C.T., 2600

Abstract

An encryption scheme is proposed which allows a hierarchical organization of keys used to encrypt and decrypt data stored in databases. The scheme uses the same method as the RSA cryptosystem [RISH78] to encrypt and decrypt data, with additional restrictions on the availability of the encryption information to the public. Its security is based on the Discrete Logarithm problem [DIHE76] which is known to be NPI. Each user is issued a key which is used for decryption and user clearance verification.

Keywords: Data encryption, Security, Databases, Distributed Databases

CR Categories: E.3, H.2.0, H.3

1. Introduction

Databases are designed to be shared by a number of users, and so a method is needed to ensure that only authorized users can access data stored in the database. The aim of this paper is to propose a scheme which allows a hierarchical organization of encryption and decryption keys associated with the levels of security in the database system. The scheme uses the same method as the RSA cryptosystem [RISH78] to encrypt and decrypt data and provides security based on the difficulty of solving the Discrete Logarithm problem [DIHE76] which is known to be NPI [GAJO79]. Each user is issued a key which is used for decryption and used for user clearance verification. Users having higher clearance levels can read data of lower clearance level, while users with lower clearance levels can request data to be stored as being of higher level than the

user's clearance. In our discussion we consider three levels of security for data, the highest level is the A-level, second is the B-level, and the lowest level is the C-level. Correspondingly, there are three groups of users: the A-clearance, B-clearance and C-clearance groups. The ideas in this paper can be easily extended to more than three security and clearance levels. This paper does not address the problem of inference control (such as that in statistical databases) and the problem of database operations on encrypted data.

The idea of securing databases using cryptographic methods is not new and a considerable amount of work has been done in this area. Gudes, Koch and Stahl [GUKO76] has put forward methods to perform cryptographic transformations in databases which preserves data structures. This includes the substitution, transposition, reduction and expansion of data items. Davida [DAWE81] has proposed an encryption scheme based on the Chinese Remainder Theorem which allows some algebraic operations to be performed on the encrypted data [DAYE82]. The Chinese Remainder Theorem is useful in that given n large primes $d_1,..., d_n$ and a particular value C_i such that $C_i = a_j \ mod \ d_j \ \ (j = 1,..,n)$ all a_j values can be retrieved from C_i by simply calculating $C_i \ mod \ d_j$. In this way a database can be encrypted, where record i of the database would correspond to C_i, the field values being encrypted would correpond to a_j, and the decryption keys for field j would correspond to d_j. Denning [DENN83] has used the Data Encryption Standard (DES) [DES77] to perform encryption and authentication at the field level. For each data element a distinct cryptographic key is used. The keys are constructed by combining the record and field identifiers together with a secret database key via a key generating function. Authentication is performed using checksums of the data elements and the key used to compute the checksums are made out of the identifiers and the database key. Another proposition which allows operations on encrypted data is that of *privacy homomorphism* put forward by Rivest, Adleman and Dertouzos [RIAD78]. The idea is that instead of decrypting data before operations are performed, the operations are done on the encrypted data. Specialized encryption functions or privacy homomorphisms are used to encrypt data and hence allows operations to be performed. This presents some applications in securing databases.

2. Security requirements and assumptions

Most secure systems classify users into one of a number of security classes or groups which corresponds to the security level attached to the data. Furthermore, in these systems it is commonly agreed that a "write-up-read-down" rule is observed. This means

that any user can read and write data to the database if the data is of the same security classification as the user's security clearance. In addition, a user with lower security clearance can only write data to a higher level security area or the same security level as the user in the database (ie. write-up). A user with a higher security clearance can only read data at the same security or from a lower security level (ie. read-down). This prevents low clearance users from reading sensitive data of higher security classification, and prevents high clearance users from giving higher clearance data to lower clearance users.

In our discussion we assume that the database system used is an untrusted one, one taken directly "off the shelf" without any modification. We further assume there is a *Security System* or *Filter* (similar to [DENN84] and [GRAU84]) which behaves as an interface between the user and the database system, and that there is to some extent a secure operating system underlying the database. This is shown in fig. 1. All queries to the database system pass through the security system which holds all cryptographic information necessary to ensure security of data stored in the untrusted database. The security system also ensures that only permitted data is passed to a user. Data items which are classified higher in the database than the user's security clearance are filtered out. Using the security system users can request the retrieved to be delivered either in clear form or in an encrypted form. In the first case the security system must perform all decryption of the retrieved data and filter out higher level data. In the later case the system must reencrypt the filtered data and the user must decrypt user his or her key. This is particularly useful for data which need to be transmitted over a public computer network.

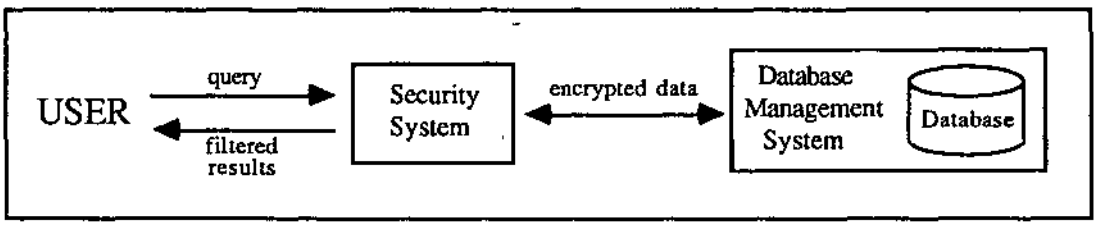

Figure 1. Security system between the user and the database system.

Other assumptions include the possibility of a Trojan Horse in the database system which could either run on its own or run with the aid of a user. To allow operations on encrypted data some form of field and record identifiers are used in the database, and that they are included in the database encryption. Finally, given that the database is shared it is necessary to store the identification of a user who performed the last update to the database. This may be part of a log mechanism which is supported in most database

systems. A more comprehensive discussion on the requirements and characteristics of database encryption can be found in [DAWE81]. Readers wishing to know more on other encryption methods are directed to [SEPI88].

3. Encryption and decryption in the RSA cryptosystem

The security of the proposed scheme is achieved by relying on the difficulty of solving the Discrete Logarithm Problem [DIHE76]. The scheme follows the RSA cryptosystem [RISH78] in the encryption and decryption of messages which is based on both the Discrete Logarithm and Factorization problems. Given a message M, encryption key K_e and decryption key K_d, M is decrypted and encrypted as follows:

$$\textit{encrypt:} \qquad C = E_{K_e}(M) = M^{K_e} \ (\mathit{mod}\ N) \tag{1}$$

$$\textit{decrypt:} \qquad M = D_{K_d}(M) = C^{K_d} = (M^{K_e})^{K_d} = M \ (\mathit{mod}\ N) \tag{2}$$

where C in this case is the resulting cryptogram. Here K_e and K_d are chosen such that:

$$K_e \,.\, K_d \equiv 1 \ (\mathit{mod}\ \phi(N)) \tag{3}$$

where $N = p \,.\, q$ for prime p and q, and $\phi(N) = lcm\,(p-1, q-1)$. $\phi(N)$ is the Euler Totient function. Equation (3) is based on Euler's generalization theorem which requires:

$$M^{\phi(N)} \equiv 1 \ (\mathit{mod}\ N),\ gcd(M, N) = 1 \tag{4}$$

Here *lcm* means least common multiple and *gcd* means greatest common divisor.

In the RSA cryptosystem N is the product of two very large primes p and q. In the proposed scheme N does not necessarily have to be the product of only two primes, though it is desirable that N is composed out of primes, each to the exponent of one only. The RSA cryptosystem makes public the values of N, K_e and C, while in our scheme we keep N, $\phi(N)$ and K_e secret. Hence the security of the proposed scheme relies only on the Discrete Logarithm problem which is known to be NPI. As will be shown later we issue decrypting keys to the user in the form of a modification to the actual decrypting key K_d.

4. The proposed scheme

The proposed scheme uses encryption of data in the database to prevent access by illegal users. Each data element is encrypted and then stored in the database, and only users with the appropriate decryption keys can decrypt any required data. The keys issued to users are chosen in a multilevel fashion, giving a number of higher clearance users more "powerful" keys and low clearance users "weaker" keys. Hence, for example, an A-clearance user will have more power than a B-clearance user and thus can access more data than the B-clearance user. The A-clearance user can read A-level data, while the B-clearance user can request data be stored as A-level.

Part of the proposed scheme is to find suitable keys K_e and K_d which satisfy equation (3). One direct way to do this is to let the product of K_e and K_d to be equal to $D(N)$ where:

$$D(N) = (c \,.\, \phi(N)) + 1 \qquad (5)$$

where c is any constant greater than zero. Note that $D(N)$ will always be equal to 1 modulo $\phi(N)$. From equation (5) we can find any two numbers whose product equals $D(N)$, and they can be used as K_e and K_d in equation (3), and hence equation (1) and (2). In other words any two factors of $D(N)$ whose product equals $D(N)$ can be used as keys in (3). We call keys which are taken directly from the set of divisors of $D(N)$ *parent* keys.

The scheme gets its multilevel property from the fact that given a pair of parent keys generated using $D(N)$ for a suitable N and c, these keys can decrypt data encrypted using some of the other parent keys also generated from the same $D(N)$. Let us consider that $D(N)$ has the following divisors $(d_1, d_2, d_3, \ldots d_n)$ in an increasing order and that the product of each of (d_1, d_n), (d_2, d_{n-1}), (d_3, d_{n-2}), . . ., (d_k, d_{n-k+1}) equals $D(N)$. Note that these pairs are parent keys, and to each of the security levels A-level, B-level and C-level we assign a pair of these divisors. If we encrypt data using d_{n-j} and $d_{j+1} | d_k$ then we can decrypt using d_k because $d_k = \alpha \,.\, d_{j+1}$, for some integer $\alpha > 0$ and positive $j < n$. And thus we have

$$d_{n-j} \,.\, d_k \equiv d_{n-j} \,.\, (\alpha \,.\, d_{j+1}) \equiv \alpha \quad (mod\ \phi(N)) \qquad (6)$$

and

$$K_{actual} = d_k / \alpha = d_{j+1} \qquad (7)$$

where K_{actual} is the actual decryption key used by the security system. So for our data M,

$$M^{d_{n-j}} \cdot K_{actual} \equiv M^{d_{n-j}} \cdot d_{j+1} \pmod{\phi(N)} \equiv M \pmod{N} \qquad (8)$$

For example take (d_1, d_n), (d_2, d_{n-1}) and (d_3, d_{n-2}). Assume that $d_1 | d_3$, $d_1 | d_2$ and $d_2 | d_3$ and we encrypt all A-level data using d_{n-2}, all B-level data using d_{n-1} and all C-level data using d_n. A-clearance users holding key d_3 can not only decrypt A-level data, but also B-level and C-level data. B-clearance users can decrypt B-level and C-level data, while C-clearance users can only decrypt C-level data. C-clearance users cannot decrypt A-level data because d_3 does not divide d_1, hence equation (7) will give $K_{actual} < 1$. On seeing this the security system will deny access to the requested data.

5. Example

To illustrate the discussion in the previous section, we present an example using only small numbers. In an actual implementation it is necessary to use numbers in the order of three hundred to four hundred digits long for N and c to ensure security.

- First, let N = 10062 , ø(N) = 3024 and c = 26.
  Then D(N) = 26.3024 +1 = 78625 ,
  and D(N) has divisors:
  5, 17, 25, 37, 85, 125, 185, 425, 629, 925, 2125, 3145, 4625, 15725
  or in pairwise order:
  (5, 15725), (17, 4625), (25, 3145), (37, 2125), (85, 925), (125, 629), (185, 425).
  To get the parent keys we reduce modulo ø(N) :
  (5, 605), (17, 1601), (25, 121), (37, 2125), (85, 925), (125, 629), (185, 425).
  and the product of each pair equals 1 modulo ø(N).

- Next, we assign the group of A-clearance users the key $K_{dA} = 125$, the B-clearance users key $K_{dB} = 25$, and the C-clearance users key $K_{dC} = 5$.
  Here note that $K_{dC} \mid K_{dA}$, $K_{dC} \mid K_{dB}$, and $K_{dB} \mid K_{dA}$.
  An A-clearance user can decrypt any data encrypted using one of the following K_e:
  629 or ((const . ø(N)) + 629) for const > 0
  121 or ((const . ø(N)) + 121) for const > 0
  605 or ((const . ø(N))+ 605) for const > 0

while a B-clearance user user can decrypt any data encrypted using one of the following K_e :

121 or ((const . ø(N)) + 121) for const > 0

605 or ((const . ø(N)) + 605) for const > 0

and a C-clearance user can only decrypt any data encrypted using K_e:

605 or ((const . ø(N)) + 605) for const > 0

- Consider an A-clearance user with requesting access to a data item M encrypted as M^{3145}. To get the actual key used internally by the security system, it calculates

$$\alpha = K_{dA} \cdot K_e \pmod{\phi(N)} \equiv 125 \cdot 3145 \equiv 5 \pmod{\phi(N)} .$$

This can be done because the system holds all parent keys securely. It then calculates:

$$K_{actual} = K_{dA} / \alpha$$
$$= 125 / 5 = 25$$

and uses K_{actual} to perform decryption of M^{3145}, and thus

$$(M^{3145})^{25} \equiv M^1 \equiv M \pmod{N}$$

- Note that if a B-clearance user with parent decryption key $K_{dB} = 25$ tries to read A-level data M^{629}:

$$\alpha = K_{dB} \cdot K_e \pmod{\phi(N)} \equiv 25 \cdot 629 \equiv 605 \pmod{\phi(N)}$$

and

$$K_{actual} = K_{dB} / \alpha$$
$$= 25 / 605 < 1$$

and hence the system will not decrypt the A-level data.

6. User decryption keys

Previously we found an encryption and decryption key for each security level. We can simply issue these decryption keys to all users of a security level, hence all users

of a level will have exactly the same key. However, it is desirable for each user to have a key which is different (or at least looks different) from the other user keys. In this section we present two ways to generate user keys from one parent key. We call keys which are generated from the parent keys *child* or children keys. These are the keys issued to the users.

6.1 User key generation based on multiplication by another parent key

First we choose a subsidiary parent key (d_x, d_{n-x+1}) such that for a given parent key (K_e, K_d) taken from the previous three parent keys all the products:

$$K_e \,.\, d_x\,,\ K_e \,.\, d_{n-x+1}\,,\ K_d \,.\, d_x\,,\ \text{and}\ K_d \,.\, d_{n-x+1}$$

does not equal one modulo $\phi(N)$. Next we issue children keys K_{d^*} derived from parent decrypting key K_d in the following way:

$$\begin{aligned} K_{d^*} &= ((\ \delta \,.\, \phi(N)\ ) + K_d\ ) \,.\, d_{n-x+1} \\ &= (\ \delta \,.\, \phi(N) \,.\, d_{n-x+1}) + (\ K_d \,.\, d_{n-x+1}\ ) \end{aligned} \qquad (9)$$

where δ is any positive integer greater than zero. Thus we can see from (9) that the following is true for data element M:

$$\begin{aligned} (M^{K_e})^{d_x \cdot K_{d^*}} &\equiv (M^{K_e \cdot d_x})^{K_d \cdot d_{n-x+1}} \equiv \\ (M^{K_e \cdot K_d})^{d_x \cdot d_{n-x+1}} &\equiv (M^1)^1 \equiv M \pmod{N} \end{aligned} \qquad (10)$$

and hence what we have given the user is a child key derived not from one parent key, but from two parent keys. Two or more users can try to find $\phi(N)$ by reducing their keys modulo some integer i until a common number $\beta = (\ K_d \,.\, d_{n-x+1}\ )$ is found. Hence $\phi(N)$ equals i. But note that β is not a parent key belonging to any of the three security levels. As before the security system must keep the subsidiary parent keys (d_x, d_{n-x+1}) secret. On receiving a key K_{d^*} from the user the system calculates:

$$\begin{aligned} (\ K_{d^*} \,.\, d_x\ ) &\equiv ((\ K_d \,.\, d_{n-x+1}\ ) \,.\, d_x\ ) \\ &\equiv (\ K_d \,.\, 1) \equiv K_d \pmod{\phi(N)} \end{aligned}$$

and use K_d to determine the user security clearance and to decrypt the requested data as in the example before. The immediate problem here is that the method relies on the dificulty

of factoring β and hence insecure. Since both K_d and d_{n-x+1} are not necessarily prime, factoring β is not as difficult as the Factorization problem [RISH78].

6.2 Generation of user keys based on exponentiation of parent keys

The second way we propose to generate user keys is to use certain powers of the parent encryption and decryption keys. It is desirable for a given parent encryption key K_e and a parent decryption key K_d to find integers X and Y such that the equation

$$K_e^X \,.\, K_d^Y \equiv 1 \ (\mathit{mod}\ \phi(N))$$

or

$$K_e^Y \,.\, K_d^X \equiv 1 \ (\mathit{mod}\ \phi(N)) \qquad (11)$$

are true. To find such X and Y we must proceed with the following calculations:

from (3) $\quad K_e \,.\, K_d \equiv 1 \ (\mathit{mod}\ \phi(N))$

thus $\quad K_d \equiv K_e^{-1} \ (\mathit{mod}\ \phi(N))$

and hence $\quad K_d^X \equiv (K_e^{-1})^X \equiv K_e^{-X} \ (\mathit{mod}\ \phi(N))$.

Solving equation (11) we have:

$$K_e^Y \,.\, K_d^X \equiv K_e^Y \,.\, K_e^{-X} \equiv K_e^{Y-X} \ (\mathit{mod}\ \phi(N)) \ .$$

and thus

$$K_e^{Y-X} \equiv 1 \ (\mathit{mod}\ \phi(N)) \qquad (12)$$

must be true.

Using Euler's theorem in equation (4) we get:

$$K_e^{\phi(\phi(N))} \equiv 1 \ (\mathit{mod}\ \phi(N)) \qquad (\mathit{where}\ gcd(K_e, \phi(N)) = 1)$$

which means

$$Y - X = \phi(\phi(N)). \qquad (13)$$

Note That in equation (11) we can interchange the use of X and Y and still produce the same results. An example is shown below.

Example:

For N = 10062 , ø(N) = 3024 and ø(ø(N)) = ø(3024) = 864 .
We take K_d = 125 and K_e = 629, and take X = 65 and Y = 929.

Now,

$K_d^X \,.\, K_e^Y$	$\equiv \; K_d^Y \,.\, K_e^X$
$\equiv \; 125^{65} \,.\, 629^{929}$	$\equiv \; 125^{929} \,.\, 629^{65}$
$\equiv \; 2141 \;.\; 1637$	$\equiv \; 2141 \;.\; 1637$
$\equiv \; 1$	$\equiv \; 1 \qquad (\text{mod } \phi(N))$

As before, an A-clearance user with K_{d^*} = 2141 requests access to a data item M encrypted as M^{3145}. On receiving the user's key, the security system checks:

$$K_{d^*} \;.\; K_e^Y \;(\text{mod } \phi(N)) \equiv 1$$

and finds that the user is an A-clearance user. Thus K_{dA} = 125. To get the actual key used internally by the security system, it calculates α as before and then finds K_{actual} to perform decryption of the required data.

Here K_d^X and K_e^Y children keys of K_d and K_e respectively. From this we see that we can give K_d^X or K_d^Y as the user child decryption key, and encrypt all data elements using just K_e . An advantage of using power X or Y of K_d is that we can choose a unique X for each user, creating a form of user identification. On receiving a user key K_d^X the security system checks the user security level by solving equation (11) using a pre-stored K_e^Y. If the user request data M to be returned in clear form, the system simply decrypts cryptogram $C = M^{K_e}$ using the secret parent decrypting key K_d

as in equation (1) and (2). If the user requests data M to be returned in an encrypted format the system returns cryptogram C' to the user where:

$$C' \equiv C^{K_d \cdot K_e^Y} \equiv (M^{K_e})^{K_d \cdot K_e^Y} \equiv M^{K_e^Y} \qquad (mod\ N)$$

and the user decrypts C' as:

$$(C')^{K_d^X} \equiv (M^{K_e^Y})^{K_d^X} \equiv M^1 \equiv M \qquad (mod\ N).$$

Note that a user knowing $K_{d^*} = K_d^X$ or $K_{d^*} = K_d^Y$ cannot decrypt data element M stored as M^{K_e} because $K_d^X \cdot K_e \not\equiv 1 \ mod\ \phi(N)$ and similarly $K_d^Y \cdot K_e \not\equiv 1 \ mod\ \phi(N)$.

Compared with the first method the second method provides the most secure way of generating unique user decryption keys. There are a number of points which must be taken into consideration when we use this solution. First of all, the number of possible values of X for K_d^X is bounded by $v_{\phi(N)}(K_d)$ where $v_m(a)$ denotes the *order of a modulo m*. This is because K_d generates only a subset of the reduced set of residues modulo $\phi(N)$, and since we do not require $\phi(N)$ to be prime it is clear that K_d cannot be a primitive root modulo $\phi(N)$. It is of course necesary that K_d and $\phi(N)$ be coprime. Given a key K_d^X and $\phi(N)$ the maximum number of user keys that can be generated equals $v_{\phi(N)}(K_d)$. Furthermore, out these keys only a total of $v_{\phi(N)}(K_d) - 4$ can be useful. This is because

$X = \lambda + 1$ gives $K_d^X \equiv K_d \ (mod\ \phi(N))$,
$X = \lambda - 1$ gives $K_d^X \equiv K_e \ (mod\ \phi(N))$,
$X = \lambda$ gives $K_d^X \equiv 1 \ (mod\ \phi(N))$, and
$X = \lambda + \pi/2$ gives $K_d^X \ (mod\ \phi(N) = K_e^Y \ (mod\ \phi(N)$

$(\ where\ \pi = v_{\phi(N)}(K_d)\ \ and\ \ \lambda = (\ constant \cdot v_{\phi(N)}(K_d)\ )\ )$

which does not give the desirable security. In addition, out of the remaining keys, there are only $(v_{\phi(N)}(K_d) - 4)/2$ keys which are safe. This is because half of the remaining keys are in fact the inverse of the other remaining half. This is shown in the following:

$$K_d^{(\lambda + i)} \cdot K_d^{(\lambda + \pi - i)} \equiv 1 \ (mod\ \phi(N)) \quad (for\ i = 1,..., \pi/2 - 1) \qquad (14)$$

and thus to ensure security we only choose $X = \lambda + i$, for $i = 2,..., \pi/2 - 1$.

7. A variation of the proposed scheme

There is a very interesting variation of the proposed scheme. It is based on the fact that given a particular $\phi(N)$ there are more than one N which produces that $\phi(N)$. That is, there is a one-to-many mapping between $\phi(N)$ and N. Recall that equation (4) requires that $gcd(M, N) = 1$ for any data M. It is clear that if N is prime equation (4) will hold true regardless of M, and that given a non-prime N it is desirable to choose N such that it consists of only a small number of prime factors. Thus using (4) we can assign a different N to each of the security classification levels in the database and perform all calculations in each level over a different modulo N. For example we can assign N_1 to the A-level, N_2 to the B-level and N_3 to the C-level, and hence all encryption and decryption of A-level data will be performed over modulo N_1, B-level data over modulo N_2, and C-level data over modulo N_3. As before N_1, N_2 and N_3 are kept secret in the security system. In our example before we had that $N = 10062$ and $\phi(N) = 3024$, but in actual fact there more than one N value for a given $\phi(N)$. Some of the other N values for this $\phi(N)$ are: *10584*, *10668*, *10836* and *10962*. Note that a different N used in each security level will not affect the number of user keys available in that level. In fact, reencryption of all data in a particular security level using a new N can be performed without having to generate new user keys for all user of that clearance level.

8. Security of the scheme for databases

The above encryption and decryption scheme presents a very attractive method to secure data stored in untrusted databases. The unit of encryption can either be whole records, fields or individual data elements in a record. Denning has suggested that field encryption may present advantages over record encryption in terms of the amount of computation required to perform database operations such as projections and selections [DENN83]. Our scheme assumes an encryption of individual data elements together with record and field identifiers, however it can be extended to encryption of records and fields. We also assume that the identifiers are kept in clear form for database operations purposes.

Unlike the RSA cryptosystem, in which N, cryptogram C and encrypting key K_e are made public, the proposed scheme only allows the children keys $K_{d^*} = K_d{}^X$ and cryptogram C public. The scheme assumes C to be public because it assumes that the database system is an untrusted one with "back-doors" to the stored data. Given that N, $\phi(N)$, $\phi(\phi(N))$, K_e, K_d, X and Y are all secret, and given that each user has a different

key generated using a different X in $K_d{}^X$, the malicious user or the attacker is faced with a problem more difficult than Discrete Logarithm problem. This is because in the RSA system N is public, whereas in this scheme N is secret. Without knowing N the attacker cannot even attempt to solve the Discrete Logarithm problem. If two or more malicious users worked together, their decryption keys cannot be reduced modulo some integer i to a common number β because each of the keys are already reduced modulo $\phi(N)$. Not knowing $\phi(N)$ and X the attacker cannot find K_d from $K_{d^*} = K_d{}^X$.

9. Remarks

In the above scheme we have taken the approach of giving as little as possible cryptographic information to the user. All the user knows is that he or she has a (decrypting) key.

Another advantage of the proposed scheme is that at any time we can re-encrypt the whole database using a different modulo N. Although it may require a large amount of computation reencryption will thwart the efforts of a penetrator or trojan horse in performing ciphertext searching of the database. The reencryption can be done on a copy of the database in the background, and brought foward when all entries are reencrypted. Otherwise, the secure system can keep track of the old and new N values and anytime a data element is accessed it is decrypted over the old N and encrypted over the new N value. This way the database is converted in a slow manner. This is because there are different values N with the same $\phi(N)$.

The proposed scheme presents a method to secure distributed database systems. In a distributed database system a number of autonomous database systems at remote sites cooperate and are linked up via computer networks. Using the proposed scheme we can assign a different N to each site and perform all encryption and decryption of data at a site based on modulo of the N at that site.

Most distributed database systems use system-wide naming for objects in the system, including user names or identification. In this manner a user is recognized at every site. Thus depending on how each site organizes it parent and children keys a particular user can be designated as having A-clearance at one site and B-clearance at another. In this manner the clearance of a user depends on the site which the user wants to access, hence preserving autonomy of that site. Note that we can either have a user using one key for all sites or one key for each site.

10. Summary

A scheme for multilevel encryption with a variation has been proposed which allows users with higher security clearance to read data of lower security classification, and which allows users with lower security clearance to store data with higher security classification. The scheme is based on the Discrete Logarithm problem to provide its security and uses the same encryption and decryption method as the RSA cryptosystem. The system holds parent keys which are used to generate children keys. All cryptographic information with the exception of the user keys is held securely internally in the security system and the system has the ability to deny access to the database. The scheme present some ways to encrypt data in a distributed database system and while still retaining its multilevel property.

Acknowledgements

The authors would like to thank Dr. Joseph Pieprzyk for many comments and suggestions regarding the issues discussed in this paper.

References

[DAWE81] G. I. Davida, D.L. Wells and J.B. Kam, "*A database encryption system with subkeys*", ACM Trans. on Database System, Vol. 6, No. 2, June 1981, pp. 312-328.

[DAYE82] G. I. Davida and Y. Yeh, "*Cryptographic relational algebra*", Proc. IEEE Symposium on Security and Privacy 19842, IEEE Soc.,pp. 111-116.

[DENN83] D.E. Denning, "*Field encryption and authentication*", Proc. CRYPTO 83, D. Chaum (ed), Plenum Press 1983, pp.231-247.

[DENN84] D.E. Denning, "*Cryptographic checksums for multilevel database security*", Proc. IEEE Symposium on Security and Privacy 1984, IEEE Soc.,pp. 52-61.

[DES77] "*Data Encryption Standard*", US National Bureau of Standards, Washington, DC, FIPS PUB 46, Jan 1977.

[DIHE76] W. Diffie and M. Hellman, "*New directions in cryptography*", IEEE Trans. on Information Theory, Vol. IT-22, No. 6, Nov 1976, pp. 644-654.

[GAJO79] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.

[GUKO78] E. Gudes, H.S. Koch and F.A. Stahl, "*The application of cryptography for data base security*", Proceedings of AFIPS National Computer Conference 1976, pp. 97-107.

[GRAU84] R. Graubart, "*The integrity-lock approach to secure database management*", Proc. IEEE Symposium on Security and Privacy 1984, IEEE Soc.,pp. 62-74.

[RIAD78] R.L. Rivest, L. Adleman and M.L. Dertouzos, "*On data banks and privacy homomorphisms*", Foundations of Secure Computation, R.A. DeMillo, D.P. Dobkin, A.K. Jones and R.J. Lipton (eds.), Academic Press, New York, 1978, pp. 169-177.

[RISH78] R.L. Rivest, A. Shamir and L. Adleman, "*A method for obtaining digital signatures and public-key cryptosystems*", Comm. of ACM, Vol. 21, No. 2, Feb 1978, pp. 120-126.

[SEPI88] J. Seberry and J. Pieprzyk, Cryptography: An Introduction to Computer Security, Prentice-Hall, Sydney, 1988.