A Cryptographic Key Generation Scheme for Multilevel Data Security

Lein Harn and Hung-Yu Lin

Computer Science Telecommunications Program, University of Missouri-Kansas City, Kansas City, MO, U.S.A.

In 1982, Akl and Taylor proposed an elegant solution to the partially ordered multilevel key distribution problem, using a cryptographic approach. Since then, continuing research has been conducted to try to realize and simplify their scheme. Generally speaking, there are two problems associated with their scheme. First, a large value associated with each security class needs to be made public. Secondly, new security classes are not permitted to be added into the system once all the security keys have been issued. Our paper presents a very similar approach. But, instead of using the top-down design approach as in their scheme, our scheme is using a bottom-up key generating procedure. The result is that the published values for most security classes can be much smaller than in their scheme. This property becomes more obvious for a broad and shallow hierarchical graph. In addition, our scheme can accommodate the changes of adding new security classes into the system.

Keywords: Cryptographic scheme, Multilevel data security, Key distribution, Partially ordered hierarchy, RSA scheme.

1. Introduction

The multilevel data security problem originally exists in military and government departments as well as some private corporations where classified data management is necessary. Now, because of the increase in computing resources, it is more frequently found in applications such as database management [1-3], computer networks [4-7], and operating systems [8, 9].

The multilevel security problem exists in many organizations where a hierarchical structure of data sensitivity and user privilege coexists. Government and military organizations are the classic examples of such hierarchies [10]. There are also examples in commercial environments. For instance, a corporate hierarchy may be organized in a tree structure, with top management at the root and security classes corresponding to divisions, departments, and projects at successive levels of the tree. A manager of a division has clearance for the security class of that division and, thereby, is authorized to access information in all departments and projects within that division. Members of a project team, on the other hand, are cleared only for that project and will be unable to access information concerning other projects, including those within the same department. A totally different application environment would be a computer running in a mul-
tlevel secure mode having users with different access rights as well as objects of various sensitivity levels.

To make the rest of our paper more precise in describing these relationships, we define the following basic terms:

- Let \( \mathcal{U} \) be the set of users in the computing environment. We define the set of security classes \( \mathcal{T} = \{ C_i \} \) of the computing environment to be a partition of \( \mathcal{U} \), i.e. \( \mathcal{T} = \{ C_i \} \)

for \( 1 \leq i \leq m \), such that \( \bigcup_{i=1}^{m} C_i = \mathcal{U} \) and \( C_i \cap C_j = \emptyset \)

for \( i \neq j \).

- We assume that the set of security classes \( \mathcal{T} \) is ordered in a hierarchy by the relation \( \leq \), where \( C_j \leq C_i \), means that \( C_j \) is subordinate to \( C_i \).

- We define a bijection \( \text{key}: \mathcal{T} \to \mathcal{K} \) for \( \mathcal{K} = \{ K_i \} \), a set of keys, such that \( \text{key}(C_i) = K_i \), for \( 1 \leq i \leq m \).

Thus, for all \( u_1, u_2 \in C_i \), \( u_1 \) and \( u_2 \) share \( K_i \) and all data secured under \( K_i \).

- Let \( \mathcal{S}(C_i) \) be the set of all subordinate classes of a given security class \( C_i \). That is, \( C_j \in \mathcal{S}(C_i) \) iff \( C_j \leq C_i \).

- Also, we define \( \mathcal{S}(C_i) = \mathcal{T} - \mathcal{S}(C_i) \).

- Let \( L_i \in \mathbb{Z} \) represent the level of the security class \( C_i \) in \( \mathcal{T} \).

With this basic terminology, the multilevel security problem is now formally defined to be the problem of securing the data of all security classes in a partially ordered set \( \mathcal{T} \) such that \( C_i \) has access to data accessible to users in \( C_j \) iff \( C_j \leq C_i \). This problem can be solved by the encryption of the data. The set of keys \( \mathcal{K} \) is used to secure the classes \( \mathcal{T} \), such that each user \( u \in C_i \) holds a subset of keys \( \mathcal{K} \) so as to be able to access his/her own data and that of subordinate users.

![Fig. 1. A multilevel security hierarchy.](image)

**TABLE 1** The set of keys required to be held by each class in hierarchy given in Fig. 1.

<table>
<thead>
<tr>
<th>Security class</th>
<th>Keys held</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>( K_1, K_2, K_3 )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>( K_1, K_2, K_3, K_4 )</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>( K_1, K_2, K_3 )</td>
</tr>
<tr>
<td>( C_4 )</td>
<td>( K_1 )</td>
</tr>
<tr>
<td>( C_5 )</td>
<td>( K_1, K_2 )</td>
</tr>
<tr>
<td>( C_6 )</td>
<td>( K_1 )</td>
</tr>
</tbody>
</table>

This quantity of keys is awkward to administer and may itself become a security hazard. This is called the key management problem of multilevel security. For example, given the special tree multilevel hierarchy shown in Fig. 1, in order to retrieve the data encrypted under each user's own key or subordinates' keys, that user has to hold a set of keys as shown in Table 1.

It is shown in Table 1 that higher privileged users, entitled to retrieve more secret information, are required to hold more keys. The large number of keys held by users at the higher level is a disadvantage, especially in systems with large numbers of security classes. Many security issues are involved when the information (or key) needs to be stored in secret. Storage for the keys is one concern because the secret key is sizable because of the security requirement. The fact is that the more keys need to be held, the more risks are involved; keys can be lost or stolen. Therefore the goal is to find a mechanism such that each user needs to hold only one key and that user is able to use that key to retrieve all the information to which he/she is entitled, while retaining a secure system.
2. Review of the Akl/Taylor Scheme

The first attempt to solve the key management problem employing cryptographic techniques was proposed in 1982 by Akl and Taylor [111]. They assume a communication system where every user belongs to one of a set of disjoint security classes and periodically receives data from an authority. The set of classes is partially ordered by the relation ≤, where Cj ≤ Ci means that any user u ∈ Cj can have access to information destined to any user u' ∈ Ci.

The problem is to design a scheme such that an object 𝑥 broadcast by u ∈ Cj and addressed to u' ∈ Cj is accessible to u' ∈ Cj if and only if Cj < Cj.

Akl and Taylor assume the existence of a key center (KC), which is responsible for key generation and key distribution. Each user in the network receives a single secret key generated by KC and uses this key to derive the secret keys of the user's subordinates. When u ∈ Cj wishes to broadcast a message 𝑥 to u' ∈ Cj, he/she first enciphers it under

Then publish 𝑡.

Step 3: The KC chooses a secret pair of large prime numbers 𝑝 and 𝑞, with product 𝑀 = 𝑝𝑞, and a random secret key 𝐾, 2 ≤ 𝐾 ≤ 𝑀 - 1, gcd(𝐾, 𝑀) = 1. 𝑀 is made public.

Step 4: The secret keys for all security classes 𝐶 ∈ 𝐶 can now be computed by KC as follows:

\[ K_j = K_0 \mod M. \]

2.2 Key Derivation Procedures

User u ∈ Cj can derive the key 𝐾j by the formula

\[ K_j = K_0 \mod M, \]

iff Cj ≤ Cj.

According to the rules of this scheme of assigning primes, the public integers 𝑡, have a special feature which makes the key derivation possible, namely

\[ t_j \text{ is divisible by } t_j \text{ if and only if } C_j \leq C_j. \]

This statement is true because 𝒟(𝐶) ⊆ 𝒟(𝐶) if Cj ≤ C, thus t, is divisible by t, Conversely, 𝒟(𝐶) ⊄ 𝒟(𝐶), if Cj ≤ C, hence t, is not divisible by t,.

This point is extremely important, since the security of the scheme relies on it. This is to say that if t/t, is an integer, the key derivation works successfully and Cj has privilege over Cj; otherwise it fails, which implies Cj has no privilege over Cj.

However, two problems arise in the practical implementation of this scheme. First, since each security class is assigned a distinct prime 𝑃, and the corresponding integer 𝑡, of each class is a product of all primes which are not subordinate to this security class in the hierarchy, it must be evaluated for each sensitivity level in advance. The size of these integers is proportional to the number of users in the network; therefore, when the number of users in the network becomes large this method becomes impractical, since 𝑡 grows.
dramatically with the increasing number of security classes. Secondly, it will violate the security requirement whenever a new security class is added into the system and it becomes the subordinate of any existing security class. In other words, it is impossible to expand the system whenever all the security keys have been issued.

Akl and coworkers have consistently attempted to solve the sizable $t$ problem. First, in 1983, MacKinnon and Akl [12] proposed two new algorithms to reduce the value of $t$; however, the result was still not satisfactory. Later, in 1985, MacKinnon et al. finally solved the mystery and found an optimal algorithm for assigning $t$ [13, 14]. Unfortunately, $t$ is still sizable. Therefore there is no other way to reduce the value of $t$, and the sizable $t$ problem is thus left unsolved.

While Akl and Taylor have solved the multilevel security problem for the general case of a partially ordered hierarchy, there are other researchers looking for solutions for the special case of a tree hierarchy. In 1987, Sandhu [15] proposed an ID-based scheme for solving the key generation problem in a tree-structured multilevel data security environment. Based on his scheme, each user's secret key is calculated from his/her own ID and his/her supervisor's secret key through a one-way function. In this way, no extra public information is needed for the key derivation. Another important advantage is that the insertion of new security classes can be easily handled. However, it requires computational overhead in deriving keys.

In this paper, we propose a key generation scheme by eliminating the sizable $t$ problem for certain hierarchies in comparison to Akl and Taylor's scheme. Generally speaking, instead of using a top-down design approach, as in all the existing algorithms, we present a bottom-up key generating scheme. The result is that the published $t$ values for most security classes are much smaller than Akl, Taylor and MacKinnon's scheme. In addition, our scheme can accommodate the changes of adding new security classes into the system.

3. Our Scheme for a Totally Ordered Hierarchy

This is the simplest multilevel hierarchy, such that for any two security classes $C_i$ and $C_j$, where $C_i \neq C_j$, either $C_i \leq C_j$ or $C_j \leq C_i$. Based on the difficulty of solving "factoring a product of two large primes," the following algorithm to assign each security class a secret key is proposed.

3.1 Algorithm

3.1.1 Key Generation Procedures
Step 1: The KC chooses and keeps three parameters for the key generation. These three parameters are two large primes $p$ and $q$, which define the publicly known parameter $n = pq$, and $\alpha \in [2, n-1]$, such that $\alpha$ and $n$ are relatively prime, i.e. gcd$(\alpha, n) = 1$. The KC also calculates a secret value, $d$, such that $d \cdot 3 \mod \phi(n) = 1$, where $\phi(n)$ is the Euler's totient function of $n$. In other words, $d = 3^{-1} \mod \phi(n)$.

Step 2: The KC generates a set of keys $K = \{K_i\}$ for all the security classes such that

$$K_i = \alpha^{\frac{\mu_i}{3}} \mod n,$$

where $L_i$ is the level of $C_i$ and which is counted from the lowest level.

Thus, each user $u \in C_i$ holds only one key $K_i$.

3.1.2 Key Derivation Procedures
If user $u_j \in C_j$ wishes to access the data of a user $u_i \in C_i$ and $C_i \leq C_j$, then $u_j$ can derive $key(C_i) = K_j$ as

$$K_j = K_i^{L_i} \mod n$$

where $L_i$ is the level of $C_i$, and $L_j$ is the level of $C_j$.

Now, Theorem 1 below, proves that the key derivation is correct.

[Theorem 1] Given $C_i$ and $C_j$ such that $C_i \leq C_j$ and given the keys $K_i$ and $K_j$ generated by the key
generation procedures, so that key \( C_i = K_i \) and key \( C_j = K_j \), and given that the key \( K_j \) is generated by the key derivation procedure, then \( K_j = K_j \).

Proof: \( K_j = K_j^{\mu - 1} \mod n \)

where \( C_j \leq C_i \) and hence \( L_i > L_j \)

\[
-(\alpha^{d_i \cdot 3^{i-1}} \mod n)^{\mu - 1} \mod n
\]

\[
= \alpha^{d_i \cdot 3^{i-1}} \mod n
\]

\[
\alpha^{(d_i \cdot 3^{i-1}) \cdot 3^{i-1}} \mod n
\]

where \( d_i \cdot 3 \mod \phi(n) = 1 \)

\[
= \alpha^{3(-1)L_i} \mod n
\]

where \( 3(-1) \mod \phi(n) = d \)

\[
= \alpha^{d_i \mod n}
\]

according to the key generation procedures

\[
= K_i
\]

QED

3.2 Example
Given the totally ordered hierarchy in Fig. 2, where \( C_i \leq C_j \leq C_k \leq C_l \leq C_m \), Figure 3 shows the keys assigned to each security class associating to our scheme.

![Fig. 3. Secret keys assigned for the security hierarchy given in Fig. 2.](image)

3.3 Security Analysis
The security of this proposed algorithm is equivalent to that of the RSA scheme [16] whose security at best is based on the computational difficulty of factoring a product of two large primes. As in the RSA scheme, the key center needs to pick \( 3 \) relatively prime to \( \phi(n) \) in the interval \([1, n - 1]\). This is assured by choosing \( p \) and \( q \) to be safe primes. The corresponding inverses \( d = 3^{-1} \mod \phi(n) \), and \( \phi(n) \), are kept secret. It is computationally infeasible for any user (or outsider) to determine the secret key \( d \) either by solving the equation \( 3 \cdot d \mod \phi(n) = 1 \) or by solving the equation \( K_i = \alpha^{d \cdot d_i \cdot 3^{i-1}} \mod n \) (which is equivalent to breaking the RSA scheme).

4. Partially Ordered Hierarchy

4.1 Algorithm

4.1.1 Key Generation Procedures
Step 1: The KC chooses and keeps three parameters for the key generation. These three parameters are two large primes \( p \) and \( q \), which define the publicly
known parameter \( n = p'q \), and \( a \in [2, n-1] \), such that \( a \) and \( n \) are relatively prime, i.e. \( \gcd (a, n) = 1 \).

Step 2: The KC assigns each security class, \( C_0 \), a distinct prime, \( e_0 \), and makes these primes publicly available. It can start to assign these primes to security classes from the bottom of the graph and select the smallest odd prime available.

Step 3: The KC calculates the multiplicative inverse, \( d_0 \), for each class. That is \( d_0 = e_0^{-1} \mod \phi(n) \).

Step 4: The KC calculates a set of public values \( t = \{ t_i \} \) and security keys \( K = \{ K_i \} \) for all security classes in the hierarchy such that
\[
t_i = \Pi e_i \quad \text{and} \quad K_i = a^{t_i \mod \phi(n)} \mod n, \quad \text{for all } j \text{ such that } C_j \in \mathcal{S}(C_i).
\]

4.1.2 Key Derivation Procedures

If user \( u_i \in C_i \) wishes to access data of user \( u_j \in C_j \) and \( C_j < C_i \), then \( u_i \) can derive \( key(C_j) = K_j \) as
\[
K_j = K_i^{t_j} \mod n.
\]

Now, Theorem 2 below, proves that the key derivation is correct.

[Theorem 2] Given \( C_i \) and \( C_j \) such that \( C_j < C_i \) and given the keys \( K_j \) and \( K_i \) generated by the key generation procedures, so that \( key(C_j) = K_j \) and \( key(C_i) = K_i \), and given that the key \( K_j \) is generated by the key derivation procedure, then \( K_j = K^t_j \).

Proof: \( K_j = K_i^{t_j} \mod n \)

where \( C_j < C_i \), and hence \( t_i > t_j \)

\[
= (a^{(t_i/t_j)^{t_i/t_j}}) \mod n
\]

for all \( k \) and \( m \) such that \( C_k \in \mathcal{S}(C_i) \) and \( C_m \in \mathcal{S}(C_j) \)

\[
= (a^{(t_i/t_j)^{t_i/t_j}}) \mod n
\]

for all \( s \) such that \( C_s \in \mathcal{S}(C_i) - \mathcal{S}(C_j) \)

\[
= (a^{(t_i/t_j)^{t_i/t_j}}) \mod n
\]

where \( (t_i/t_j)^{t_i/t_j} \mod \phi(n) = 1 \)

\[\text{QED}\]

4.2 Security Analysis

The relation between the personal secret keys belonging to two unrelated distinct users \( u_i \) and \( u_j \) is very simple. For example, user \( u_i \)'s key cannot be calculated from user \( u_j \)'s key \( K_i \) since user \( u_i \) cannot discover \( u_j \)'s system secret key \( d_j \). Similarly, the conspiracy of all children keys cannot discover the ancestors' keys since each ancestor's key includes his/her own system secret key \( d_j \). All of these attacks run up against the problem detailed in the previous section, and so discovering some key to which one or more users is not entitled would be equivalent in difficulty to breaking the RSA scheme. In summary, as long as all the system secret keys \( \{d_i\}, p \) and \( q \), are kept secret, conspiracy attacks cannot succeed, and individual users of the system are bound to only what they are supposed to know.

5. Comparisons Between Akl and Taylor's Scheme and Our Scheme

Let us use the following example taken from Akl and Taylor's original paper [11] to illustrate the difference between these two schemes.

5.1. Example

Given the partially ordered graph has six security classes, \( C_i, i = 1, 2, \ldots, 6 \), as shown in Fig. 4, Table 2 shows the keys assigned to each security class associating with Akl and Taylor's scheme and our scheme.

There are several differences between Akl and Taylor's scheme and our scheme.

Fig. 4. A partially ordered security hierarchy.
TABLE 2  The set of keys required to be held by each class in hierarchy given in Fig. 4.

<table>
<thead>
<tr>
<th>Akl and Taylor's scheme</th>
<th>Our scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$ $a{t=e}$</td>
<td>$a^{i_1d_1d_1a_1} {t=e}$</td>
</tr>
<tr>
<td>$K_2$ $a^{d_2d_2a_2} (e,e_2e_3e_4)$</td>
<td>$a^{i_2d_2a_2} (e,e_2e_3e_4)$</td>
</tr>
<tr>
<td>$K_3$ $a^{i_3r_3i_3r_3} (e,e_2e_3e_4)$</td>
<td>$a^{i_3d_3r_3} (e,e_2e_3e_4)$</td>
</tr>
<tr>
<td>$K_4$ $a^{d_4d_4a_4} (e,e_2e_3e_4)$</td>
<td>$a^{i_4d_4} (e,e_2e_3e_4)$</td>
</tr>
<tr>
<td>$K_5$ $a^{d_5d_5a_5} (e,e_2e_3e_4)$</td>
<td>$a^{i_5} (e)$</td>
</tr>
<tr>
<td>$K_6$ $a^{d_6d_6a_6} (e,e_2e_3e_4)$</td>
<td>$a^{i_6} (e)$</td>
</tr>
</tbody>
</table>

6. Conclusion

We have proposed a new systematic solution to the multilevel key generation problem. The results show that our scheme (a) is more efficient in the memory utilization since it needs less space to keep public information and (b) can handle new user's insertion without changing all keys.

References


Lein Harn was born in Taipei, Taiwan, in 1954. He received the B.S. degree from the National Taiwan University in 1977, the M.S. degree from the State University of New York, Stony Brook, NY, in 1980, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, in 1981, all in electrical engineering.

From 1981 to 1984 he was a Research/Teaching Assistant and was involved in research on signal detection and digital filtering in the Department of Electrical Engineering at the University of Minnesota. Since 1984 he has been an Assistant Professor in the Department of Electrical and Computer Engineering, University of Missouri at Columbia, and the Department of Computer Science, University of Missouri at Kansas City, MO. From 1980 to 1987 he was a Visiting Associate Professor at the National Cheng Kung University, Taiwan, R.O.C. Currently he is an Associate Professor at the Computer Science Telecommunications Program, University of Missouri at Kansas City, MO. His research interests include digital filter design, multidimensional digital signal processing, data security, cryptography, and VLSI design.

Hung-Yu Lin was born in Yunling, Taiwan, in 1963. He received the B.S. degree from the Department of Computer Science, Tunghi University, Taiwan. From 1987 to 1988 he was a system engineer at the Nanyang Auto Inc., Taiwan. Currently he is a graduate student at Computer Science Telecommunications Program, University of Missouri at Kansas City, MO. His research interests include data security and cryptography.