Comments on a cryptographic key assignment scheme

Qiang Tang and Chris J. Mitchell
Information Security Group
Royal Holloway, University of London

8th July 2004

Abstract

In this paper we analyse the security of a cryptographic key assignment scheme, recently proposed by Huang and Chang, that is designed to provide time-constrained hierarchical access control. We show that the new scheme has potential security vulnerabilities, which enable malicious users and attackers to violate the privacy of other users.

keywords: Key assignment, Access control.

1 Introduction

In [1], Huang and Chang propose a new key assignment scheme designed to provide time-constrained hierarchical access control. The authors claim that this scheme satisfies the following four security requirements:

1. The key generation and key derivation algorithms are quite simple.
2. Dynamic access control is easily implemented because most keys and other information items of the system do not need to be immediately changed when a class insertion or deletion occurs.
3. The bit-length of the secret information $I(i, z)$ held by users in class i is independent of the number of classes in the hierarchy and the number of time periods in the scheme.

4. After the expiry of time period z, the holder of keying information $I(i, z)$ should not be able to use this to obtain any keys for any time periods later than z.

The authors of [1] claim that their system is not only secure but also practical and sufficiently flexible to be used in an open network environment. However, we show below that the new scheme has significant potential security vulnerabilities, which enable malicious users and attackers to violate the privacy of other users.

The rest of this paper is organised as follows. In Section 2, we give a concise description of the key assignment scheme. In Section 3, we give our comments on this scheme. In Section 4, a brief conclusion is provided.

2 Description of the key assignment scheme

There are two kinds of entity involved in the key assignment scheme, namely the Central Authority (CA), and users, who belong to a collection of classes arranged in a partially ordered hierarchy. The CA generates and assigns a key to each class. All secret parameters are managed by the CA. We assume that the partially ordered hierarchy has m classes c_i $(1 \leq i \leq m)$, which are partially ordered by the binary relation $\leq$. The time during which the scheme operates is divided into N periods, numbered 1, 2, $\cdots$, N.

The work flow of the scheme can be divided into the following five stages.

- Initial computation

    1. The CA chooses two large primes p and q, where $p = 2p_1 + 1$ and $q = 2p_2 + 1$, and p_1, p_2 are two large primes. Let $n = pq$.
    2. For each class c_i in the partially ordered hierarchy, the CA randomly chooses the public value e_i $(1 \leq i \leq m)$ to be small so as to ensure that $\prod_{i=1}^{m} e_i \leq n$, and so that each e_i is relatively prime to $\phi(n)$. The CA then computes d_i $(1 \leq i \leq m)$ such that $e_i d_i \equiv 1 \pmod{\phi(n)}$ $(1 \leq i \leq m)$.
    3. The CA randomly selects a secret value a and computes $k_i = a^{\prod_{c_j \leq c_i} d_j} \bmod n$ $(1 \leq i \leq m)$.
    4. The CA selects a function $f(t)$, $1 \leq t \leq N$, where each $f(t)$ is chosen to be sufficiently small that $3 \leq f(t) \ll p_1, f(t) \ll q_1$, and

$\gcd(\phi(n), f(t)) = 1$ $(1 \leq t \leq N)$, where $x \ll y$ means that x is much less than y. The CA ensures that $\prod_{t=1}^{N} f(t) \prod_{i=1}^{m} e_i \leq n$.

5. The CA publishes the parameters e_i $(1 \leq i \leq m)$, n, and the function $f(\cdot)$. Suppose that all authorized users know the current time, and that the total lifetime of the scheme is divided into N time periods. The other parameters are kept by the CA securely so that no users can access them.

- Key assignment and distribution

  The CA constructs $s_t = \prod_{k=t}^{N} f(k)$ $(1 \leq t \leq N)$. The key associated with class c_j at time period t is $k_{j,t} = a^{d_j s_t} \bmod n$.

  When a user is assigned to class c_i, and is permitted to access the keys for this class up to time period z, he is given the information $I(i,z) = k_i^{s_z} \bmod n$.

- Key derivation

  From the given information $I(i,z)$ and the public parameters, a user can derive the secret key $k_{j,t}$ if and only if the class $c_j \leq c_i$ and the time $t \leq z$. He computes $k_{j,t}$ as: $k_{j,t} = I(i,z)^{h \prod_{c_k \leq c_i, c_k \neq c_j} e_k} \bmod n$, where $h = \prod_{v=t}^{z-1} f(v)$.

- Adding a class

  Suppose that a new class c_{m+1} is added to the existing system. The CA only updates the information $I(i,z)$ of those security classes c_i which satisfy $c_{m+1} < c_i$. All other information in the system stays the same.

  First, the CA randomly chooses a small integer e_{m+1} which is relatively prime to $\phi(n)$, and derives d_{m+1} such that $e_{m+1} d_{m+1} \equiv 1 \pmod{\phi(n)}$. The CA then computes the new information $I'(i,z) = (I(i,z))^{d_{m+1}} \bmod n$ for the classes c_i which satisfy $c_{m+1} < c_i$, and securely distributes this secret information to the members of class c_i.

- Deleting a class

  To delete a class c_k from an existing system, the CA discards the secret data d_k and the public parameter e_k of the class c_k. If class c_i satisfies $c_k < c_i$, then the users in this class can compute the new information as $I'(i,z) = [I(i,z)]^{e_k} \bmod n$. The other users in the system are not affected.

3 Comments on the key assignment scheme

It is claimed in [1] that the proposed scheme is secure and appropriate for key distribution in an open network. However, we now show that the key assignment scheme has the following two significant security flaws.

1. Firstly, users may collude to obtain keys which they are not individually entitled to. Suppose gcd($f(t)$, e_i)=1 for some t and i. Then, given knowledge of $k_{i,t}$, the key for class c_i at time t, and given the value $a^{s_{t+1}}$, available to any user who is part of the system at time $t+1$, then the secret key $k_{i,t+1}$ can be computed, as follows.

   First note that, since gcd($f(t)$, e_i)=1, it is straightforward to find integers u and v such that $f(t)u + e_i v = 1$. Now observe that:

$$\begin{aligned}(a^{s_{t+1}})^v.(k_{i,t})^u &= (a^{s_{t+1}e_i d_i})^v.(a^{d_i s_t})^u \\ &= (k_{i,t+1})^{e_i v}.(k_{i,t+1})^{f(t)u} \\ &= (k_{i,t+1})^{e_i v + f(t)u} \\ &= k_{i,t+1}\end{aligned}$$

   and the result follows.

2. Secondly, when a class c_k is deleted from the hierarchy at time period t, the authors in [1] claim that it is only necessary to update the key information of c_i, where $c_k \leq c_i$. However it is obvious that at time period t the deleted users in the deleted class c_k can still obtain the key $k_{j,t}$ for all classes c_j for which $c_j \leq c_k$, using the existing key information. This property makes this scheme unsuitable for many applications.

In addition, the proposed scheme is also subject to security vulnerabilities in some specific situations as described below.

1. The scheme has the property that any person, either a valid user or an attacker, who gets the key material for class c_i at time period t can use it to obtain all the keys for any time period $t' \leq t$ and any class c_j ($c_j \leq c_i$). Suppose Alice gets the key material $I(i,t)$ of time period t for class c_i. Then she can compute key $k_{j,t'} = I(i,t)^{h \prod_{c_k \leq c_i, c_k \neq c_j} e_k} \bmod n$ for class c_j ($c_j \leq c_i$) at time period t' ($t' \leq t$), where $h = \prod_{v=t'}^{t-1} f(v)$.

   This means that, for example, a newly admitted user can obtain secret keys in use prior to the time of his or her admission to the system.

This is clearly a property that would be undesirable in many possible application scenarios.

2. The description of the scheme in [1] does not include a specification of how the parameters e_i and $f(t)$ should be chosen, despite the fact that inappropriate choices for these parameters compromise the security of the scheme. To our knowledge, the choice of these parameters should at least satisfy the following two criteria:

   (a) Firstly, it is essential that all the elements e_i $(1 \leq i \leq m)$ are distinct. More generally it should also be the case that $e_i \neq ce_j \pmod{\phi(n)}$ for any integer c and for any distinct values i, j. Otherwise, the users in class i can always compute the key $k_{j,t} = (k_{i,t})^c \bmod n$, whether $c_i \geq c_j$ is satisfied or not.

   To see why this holds, suppose $e_i \equiv ce_j \pmod{\phi(n)}$ for some $i \neq j$. Hence $d_j \equiv cd_i \pmod{\phi(n)}$. Hence

   $$k_{j,t} = a^{d_j s_t} \equiv a^{cd_i s_t} \equiv (k_{i,t})^c \pmod{n}$$

   as required.

   Not only are the requirements on the values e_i missing, but the authors do not seem to be aware of the necessary restrictions, since **Example 1** from [1] contains insecure choices of values. Suppose the CA chooses the parameters as suggested in **Example 1** of [1], i.e. $e_i = 4i + 1$ $(1 \leq i \leq m)$, and suppose also that $m \geq 6$. Note that we immediately have that $e_1 = 5$ and $e_6 = 25$, i.e. $e_6 = 5e_1$. Then a user in class c_6 can always compute the keys for class c_1 at any time period t, by computing $k_{6,t}^5 = (a^{d_6 s_t})^5 \equiv a^{d_1 s_t} \equiv k_{1,t} \pmod{n}$, since $5e_1 = e_6$ implies $d_1 \equiv 5d_6 \pmod{\phi(n)}$. So if $c_6 \not\geq c_1$, then the security of the scheme will be compromised because the users in class c_6 can access the data of class c_1.

   (b) Secondly, the values $f(t)$ $(1 \leq t \leq N)$ and e_i $(1 \leq i \leq m)$ should also be pairwise distinct. In fact, analogously to above, it should never be the case that $f(t) = ce_i$ for any integer c and any values i and t.

   To see why this constraint is necessary, suppose $f(t) = ce_i$ for some i and t. Then

   $$k_{i,t} = a^{d_i s_t} \equiv a^{d_i f(t) s_{t+1}} \equiv a^{cd_i e_i s_{t+1}} \equiv (a^{s_{t+1}})^c \pmod{n}.$$

   However, a user who holds any key for time period $t + 1$ ($k_{j,t+1}$ say) can compute $(k_{j,t+1})^{e_j} = a^{s_{t+1}} \bmod n$. Hence, every user who

is part of the system at time $t+1$, or any later time, can compute $k_{i,t}$ (and also $k_{i,s}$ for all s for which $c_s \leq c_t$).

Again, it would appear that the authors of [1] are unaware of these requirements because of the choices for parameters in the example given. Suppose the CA chooses the parameters as suggested in **Example 1** of [1], i.e. $e_i = 4i + 1$ $(1 \leq i \leq m)$, $f(t) = 2t + 15$ $(1 \leq t \leq N)$. Suppose also that $m \geq 4$ and $N \geq 2$. First observe that $e_4 = f(1) = 17$. The key for class 4 at time period 1 is $k_{4,1} = a^{\prod_{k=2}^{N} f(k)} \bmod n$. However, any user in any class c_i $(1 \leq i \leq m)$, after getting his key at time period t $(t \geq 2)$, can also compute the key for class 4 in time period 1 as $k_{4,1} = k_{i,t}^{e_i \prod_{k=2}^{t-1} f(k)} \bmod n$.

4 Conclusion

In this paper, we have analysed a new key assignment scheme with time-constrained hierarchical access control, and demonstrated the presence of significant design weaknesses.

5 Acknowledgements

The authors would like to express deep appreciation to the reviewers for their valuable comments.

References

[1] H.-F. Huang and C.-C. Chang. A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Computer Standards & Interfaces*, 26:159–166, 2004.