A Cryptographic Key Assignment Scheme with Adaptable Time-token Constraint in a Hierarchy

Hsing-Chung Chen[*]
Department of Computer Science and Engineering Asia University, Taichung County, Taiwan 41345
E-Mail: shin8409@ms6.hinet.net

Shiuh-Jeng Wang
Department of Information Managemei
Central Police University, Taoyuan County, Taiwan 333
E-Mail: sjwang@mail.cpu.edu.tw

Jyh-Horng Wen
Department of Electrical Engineerin
Thonghai University, Taichung County, Taiwan 40704
E-Mail: jhwen@thu.edu.tw

Abstract

A scheme, which uses flexible cryptographic key management upon adaptable time-token constraint for a user hierarchy access control (UHAC) scheme, is proposed in this paper. For adapting the changeability in a UHAC system, we propose a technique of assigning independent time-token which is distributed by a trusted agency server to reply an authorized user for once secure access request. The key feature of the technique is to adapt some secure parameters in distributed time-token for responding to each legal access request. Further, all class keys will be updated proactively by the concept of proactive key management, which makes the advantage that is proven and free from the scenario of the collusive attack example which proposed by Yi et al. in [18]. This cryptographic key assignment scheme based on the difficulty in solving a discrete logarithm, with adaptable time-token constraint, can achieve better security and more efficient management, than the conventional UHAC scheme. Besides, our scheme provides a flexible manner and dynamic key management to increase the availability of user access in UHAC structure.

Keywords: *key assignment, proactive key management, access control, adaptable time-token*

1. Introduction

Due to the continuing growth of computer networks and multi-user computer technologies, the sharing of network resources is becoming increasingly widespread. The administration of these resources, in a multi-user computer environment, has also grown more complex; access control, through the proper authorization procedures, is therefore becoming more and more important in the real world. The problem of access control arises in organizations having a hierarchical structure; such organizations can range from military or governmental departments to private corporations. Many applications exist some secure access problems in business or other areas of the private sectors; *e.g.*, some protected databases containing sensitive information or industrial secrets.

Since Akl & Taylor, in 1983 [1], first proposed a cryptographic key assignment for access control in a partial-order hierarchy (called key assignment, hereafter), based on the degree of difficulty of solving the discrete logarithm problem, many methods of hierarchical access control [4, 9, 10] have been proposed. A hierarchical structure (also called user hierarchical access control scheme, UHAC) is constructed by dividing users into a number of disjointed security classes, $C_1, C_2, ..., C_m$, which are partially ordered with a binary relationship ‘‘ $\underline{\pi}$ ’’. In a hierarchy, the statement $C_j \underline{\pi}\, C_i$ means that the security level of class C_j is lower than that of class C_i and $C_j \underline{\pi}\, C_i$ additionally denotes the possibility that the security level of

[*] Correspondence

class C_j can equal class C_i, where C_j is a successor of class C_i, and C_i is a predecessor of class C_j. The meaning of $C_j \underline{\pi} C_i$ is that users in class C_i can access any information held by users in class C_j, while the opposite is not allowed. To go through hierarchical access control, certain key assignment methods have common procedures, as stated below. The Central Authority (called CA, hereafter) agency calculates the cryptographic key for each class of each stratum in advance, where each class is according to his membership that comes from various assignments. The CA then distributes the cryptographic key to users who are members in this class. Thus, their key assignment assigns a cryptographic key k_i to all users in class C_i so that they can use k_i to derive k_j if, and only if, the class $C_j \underline{\pi} C_i$. In other words, the conventional UHAC is the meanings of those users are allowed to derive class key depending on their individual access right without any interaction with the CA. Hence, the user in class C_i can use k_i to decrypt the data which are encrypted with k_i, and the user in the class C_i uses k_i to derive the class key k_j which is then used to decrypt the encrypted data in class C_j. The advantage of Akl and Taylor's scheme is that the key generation and key derivation algorithms are simple. However, Akl and Taylor's scheme is not practical for implementing dynamic access control, because the whole system must be re-established, once a class insertion or deletion occurs. To allow a dynamic change of classes, most researchers of this topic have proposed schemes [4, 9, 10] that perform better in allowing insertion and deletion of classes within the hierarchy in a UHAC scheme.

Furthermore, these key assignment schemes [1, 4, 9, 10] have not been concerned with a practical situation: Users might be assigned to a class for only a period. If the user has resigned from her/his class C_i, but he premeditatedly eavesdrops on data transmissions, then he can also decrypt the data in class C_j if, and only if, the class $C_j \underline{\pi} C_i$. Thus, all messages are likely to be revealed during the system's period of change. Tzeng [16] first proposed a time-bound cryptographic key assignment scheme, a UHAC scheme with time constraint, for access control in a partial-order hierarchy, in which the cryptographic keys of a class were different for each period. A special feature of Tzeng's scheme lies in the fact that each class had some time-dependent keys, each of which could only be used, during a certain period. According to Tzeng's scheme, once the key information expired, its owner could not access any subsequent class keys. His method is more flexible for some applications, such as, broadcasting digital data to authorized users with multilevel-security, or constructing a flexible cryptographic key backup system. Although broadcasting data can save a lot of bandwidth over point-to-point transmission, the problem is still that anyone on the broadcast channel can eavesdrop. To prevent eavesdropping, Tzeng's scheme encrypted the data before broadcasting it, so that only authorized users with proper keys could decrypt the data and obtain meaningful messages.

However, Yi *et al.* [18] showed that the proposal given in Tzeng's scheme [16] was not secure against collusive attacks, whereby three non-authorized users may conspire to access some class keys, to which they had no access. Yi *et al.* [18] brought out the fact that Tzeng had not proposed a way to overcome this type of collusion. Following the Tzeng's scheme [16], Huang *et al.* [8] proposed a new cryptographic key assignment with time-constraint scheme for a partial-order user's class hierarchy applied to a UHAC scheme. Nevertheless, Tang *et al.* [13] showed that Huang *et al.*'s scheme had potential security vulnerabilities, which enable malicious participants and attackers to violate the system.

The other researches of this topic are shown below. Chien [5] proposed a solution based on the assumptions of low-cost and tamper-resistant equipment to the studies of time-bound hierarchical key assignments. Unfortunately, De Santis *et al.* [12] showed that Chien's scheme [5] still meets the collusion attacks. Wang *et al.* [17] proposed a technique called merging, and have used it to construct an efficient time bound hierarchical key assignment scheme where the communication load of transmitting multiple keys is one single aggregate key. The scheme has a good storage and computational performance.

These schemes [5, 8, 16, 17] aim to provide efficient access control methods with regard to user's classes which are organized hierarchically, and the period. In this way, they offer a worthy security level in UHAC scheme with the time constraint, respectively. Whatever, some situations have been happened when a user leaves a class, the keys of that class and all the descendent classes must be renewed and where the privileges of users change frequently or where there are many classes change frequently, the communication load for key redistributions is very large [17]. However, there is a classical idea, called proactive security, that may improve their class key management works for gaining more flexibility and security. In order to meet the desired demands for gaining more flexible class key management and reinforcing the further security in applications, we therefore proposed yet another more flexible key assignment in the UHAC scheme. The sophisticated solutions proposed in our scheme are based the ideas of proactive security and top-down key assignment in [1–4, 6, 7, 9, 10, 14]. Proactive security, first suggested by Ostrovsky and Yung [14] in 1991, refers to security and availability, in the presence of this mobile adversary attack which may corrupt all participants throughout the lifetime of the system in a non-monotonic fashion; this adversary attack is limited, however, to the number of participants it can corrupt during any distinct period. Frankel, *et al.* [6], proposed a proactive mechanism to provide an increased level of security and availability to an RSA public-key system via distribution of a private key and active communication between shareholders. They emphasized that proactive security concerns are vital in dealing with the increasing number of threats to communication network domains, and for securing long-lived cryptographic keys that cannot be easily replaced, *e.g.*, basic cryptographic infrastructure functions. Their proposed scheme was engaged in a "proactive maintenance" of key sharing, protecting them against an adversary attempting to uncover the secret or disrupt their operation. In addition to protection, proactive maintenance techniques provide flexible and dynamic key management.

Owing to its simple algorithm and fewer parameter requirements, the RSA public-key system [10] remains the most popular cryptosystem, with worldwide standards; thus, we adopted this system in our proposed scheme, in which the cryptographic key of class C_i at a period t_r is k_{i,t_r}. The period does not necessarily have to be a real time, however, it may be one week, one month or half a year; it completely depends on the management of proactive mechanism in the system. We actually divided the total time T into Z periods, initialing with t_1 and ending with t_z, *i.e.* $T = \{t_1, t_2, ..., t_z\}$. Suppose that a user u_i in class C_i has gotten the information $I(i, T_{u_i})$ with the authorized period set T_{u_i}, $T_{u_i} \subseteq T$, consisting of the periods which are from her/his beginning period t_s to her/his ending period t_e, *e.g.* $T_{u_i} = \{t_s, t_{s+1}, ..., t_r, t_{r+1}, ..., t_e\}$. Therefore, she/he can use the information $I(i, T_{u_i})$ together with a time-token τ_{u_i,t_r} to compute the class key k_{j,t_r} of class C_j at the period t_r if, and only if, the conditions $C_j \underline{\pi} C_i$ and $t_r \in T_{u_i}$ are satisfied, where the time-token τ_{u_i,t_r} is issued from a Central Authentication and Authority (CAA for short) server which acts for a trusted agency in our system. Once the authorized time duration T_{u_i} elapses, the information owner cannot access any subsequent class keys. Therefore, our key derivation is constrained, not only by the class relationship, but also by the time-token corresponding to user's authorized period set respectively. In this proposed method, the time-tokens are constructed using the distinct primes distributed from the CAA server, when the users activate the requests for accessing the system. This means that a the user u_i who is only assigned a the authorized time period set T_{u_i} cannot directly use her/his pre-assigned key information the key k_{i,t_r} to derive the secret key $k_{i,t_r'}$ where $t_{r'} \notin T_{u_i}$ and u_i could not obtain the authorized time-token $\tau_{u_i,t_r'}$ from CAA, in which the time period t_r' does not belong to the time duration, *i.e.*, $t_r' \notin T_{u_i}$. Moreover, this feature differs from Tzeng's scheme [16], Huang *et al.*'s scheme [8] and the other above-mentioned schemes [4, 5, 9, 10]. In their schemes, the key mechanisms are not offered to refresh all of class keys and protect future messages. Moreover, on when many classes' insertion or deletion frequently occurs, it can be seen that most of the previously proposed schemes have to be repeatedly recomputed and republish the related information, such as the users' information and corresponded public/private parameters. By our proposed method, these tasks are ingenious recomputed by CAA; on the contrary, the authorized users can avoid being re-updated their user information frequently. Furthermore, our proposed method has the following properties:

(i) Better security and easier management, as it provides a UHAC mechanism with time-token constraint for cryptographic key assignment in a hierarchy; it also offers greater security against attacks of collusion.

(ii) Dynamic and flexible access control can be easily implemented because the system's initial keys or information items do not require changing, once classes insertion or deletion have occurred.

(iii) Algorithms, for key generation and key derivation, using distinct time-tokens, are quite simple.

(iv) If the access time is out of the authorized period set, *e.g.* T_{u_i} including the periods before t_s and the remaining $(t_z - t_e)$ periods, the class key with time-token constraint owner cannot decrypt the protected data, or information.

From the above properties, it can be seen that the key management proposed, in this paper, is more flexible and more practical, since a person may be employed for only some authorized period set. For example, we may consider a secure application of a key assignment by legal authorities, such as an authorized user, in class C_i, who wants to read the time-dependent messages, encrypted with parameters corresponding to a period t_r in class C_j. If the key assignment scheme is updated by regular periodical period, corresponding to the time-token τ_{u_i,t_r}, the appropriate key is k_{j,t_r} for desired period t_r. The user $u_i \in C_i$ can decrypt the time-dependent messages if, and only if, $C_j \underline{\pi} C_i$ and her/his access period is valid. In the same manner, an authorized participant can broadcast data, so that only authorized users with appropriate keys in the valid time duration can decrypt the messages, and obtain meaningful information. Broadcasting messages can save a great deal of bandwidth over point-to-point transmission [16]. Hence, our scheme is suitable for some applications in today's computer network environment, *e.g.*, electronic paper subscription and digital TV broadcasting [16, 11], in which a user may be assigned to a certain class for only a period. Moreover, the hierarchical access control mechanisms [4, 8, 9, 10, 16] do not retain complete access records, as this would undermine security.

The remainder of this paper is organized as follows: our key assignment scheme for dynamic and flexible access control in a hierarchy is proposed in Section 2. The discussions are given in Section 3. A brief conclusion is offered in Section 4.

2. Our Scheme in Key Assignments

A new flexible cryptographic key assignment scheme is proposed in this section, in which the class keys are periodically updated by changing the periods for normal service time and the class keys could be refreshed in real time while dynamic change of classes or users happening. This provides a proactive policy to avoid the situation that the secret class key to be broken (cracked), it will be controlled rather than responding after the fact. A special feature of our scheme is that each class has some revised time-dependent keys (called class keys for simply), none of which can be used, except during certain periods. Once the class key and user information expires, its owner cannot access any subsequent class keys.

Our scheme is also based on an RSA assumption [10], using an idea of proactive mechanism [2, 3, 6, 7, 14] in which the class keys are periodically updated by changing the periods. Assume that a partial-order hierarchy has a number of disjoint classes $C_1, C_2, ..., C_m$ ordered by the binary relation "$\underline{\pi}$" and that the system time during which the desired hierarchy is valid is divided into periods, numbered period 1, period 2,…, period z, in which the periods are also denoted as period t_1, period t_2,…, period t_z. Each of periods is assigned a constant number such as $t_1 = 1$, $t_2 = 2$,…, $t_z = z$. Our scheme can be summarized as follows:

2.1 Preliminary

Proposition 1. *Given an output of y, $y \equiv (\alpha)^x (\bmod n)$, it is computationally infeasible to find α based on the assumption of RSA, where x is a positive integer restricted in RSA assumption and n is the product of two large secret primes.*

2.2 Initial computation

Step 1: The CAA chooses two large primes p and q, where $p = 2p_1 + 1$ and $q = 2p_2 + 1$, p_1 and p_2 are two large primes, and $n = p \times q$. Then the CAA chooses a secret key d_c, and finds the public key e_c, where the secret key satisfies $e_c \times d_c \equiv 1 (\bmod \phi(n))$.

Step 2: The CAA randomly chooses m distinct primes $e_1, e_2, ..., e_m$ for m classes at a period t_r, where each e_i is to be as small as possible so that $\prod_{i=1}^{m} e_i < \phi(n)$ and each e_i is asked to be a prime related to $\phi(n)$. Then the CAA computes the uniquely corresponding numbers of $d_1, d_2, ..., d_m$ such that $e_i \times d_i \equiv 1 (\bmod \phi(n))$, for $i = 1, 2, ..., m$ where $\phi(\cdot)$ is the well-known Euler's totient function and keeps all d_i's secret.Step 3: The CAA randomly chooses distinct prime numbers $a_{t_1}, a_{t_2}, ..., a_{t_r}, ..., a_{t_z}$ and keeps them secret, where each prime is a prime related to n. Each of the primes, a_{t_r}'s, will be then used to generate the distinct class keys for a valid period t_r where $t_r \in T$.

Step 4: The CAA computes the initial keys, $k'_{i,t_r} \equiv a_{t_r}{}^{\prod_{C_l \underline{\pi} C_i} d_l} \equiv a_{t_r}{}^{(d_1 \times d_2 \times .. \times d_m) \prod_{C_v \not\underline{\pi} C_i} e_v} (\bmod n)$ for $1 \le i \le m$ and $t_r \in T$, where the initial keys are employed for m class, timing from t_1 to t_z.

Step 5: The CAA randomly chooses relatively prime numbers set $\Gamma = \{a_{u_1}, a_{u_2}, ..., a_{u_k}\}$, where each $a_{u_i} \in \Gamma$ for all $i \in \{1, 2, ..., k\}$ satisfying gcd $(a_{u_i}, n) = 1$ and gcd$(a_{u_i}, a_{t_r}) = 1$. Then they will be assigned to k distinct users $u_1, u_2, ..., u_k$. Thereupon the CAA computes the inverse number for each $a_{u_i} \in \Gamma$, denoted as $(a_{u_i})^{-1}$ where $i \in \{1, 2, ..., k\}$, such that satisfying $a_{u_i} \times (a_{u_i})^{-1} \equiv 1 \ (\bmod n)$. The CAA keeps each pair of numbers, a_{u_i} and $(a_{u_i})^{-1}$, for all $i \in \{1, 2, ..., k\}$, secret.

Step 6: For each period t_r, the CAA randomly chooses one of the distinct primes $g_{t_1}, g_{t_2}, ..., g_{t_r}, ..., g_{t_z}$, where any $g_{t_r} \ne e_i$ for any t_r and i, and keeps them secret; That is, for each period t_r, entire classes should be randomly assigned the distinct primes $g_{t_1}, g_{t_2}, ..., g_{t_r}, ..., g_{t_z}$ and each of them should be as small as possible so that $3 \le g_{t_r} << p_1$, $3 \le g_{t_r} << p_2$, satisfying gcd$(\phi(n), g_{t_r}) = 1$, for all $t_r \in T$. The distinct primes are then used to generate distinct time-tokens. Finally, the distinct time-tokens are used to compute the corresponding class keys for each class at the period t_r.

2.3 Key assignment

In this key assignment phase, the CAA computes the class key for class C_i from the initial keys k'_{i,t_r}, where $i \neq 0$ and $i \geq 1$, at the period t_r as follows,

$$k_{i,t_r} \equiv \left(k'_{i,t_r}\right)^{g_{tr}} \equiv \left(a_{t_r}\right)^{g_{tr} \prod_{Cl \underline{\pi} Ci} dl} \equiv \left(\left(a_{t_r}\right)^{g_{tr} \times (d_1 \times d_2 \times \ldots \times d_m) \times \prod_{Cv \underline{\not\pi} Ci} e_v}\right) (\text{mod } n), \qquad (1)$$

where the CAA server keeps the prime number $g_{t_r} \in \{g_{t_1}, g_{t_2}, \ldots, g_{t_r}, \ldots, g_{t_z}\}$ secret.

2.4 Information assignment

The CAA then chooses a prime number a_{u_i} which has been randomly chosen from the prime set $\{a_{u_1}, a_{u_2}, \ldots, a_{u_i}, \ldots, a_{u_k}\}$ for the user u_i who is assigned a time-bound $T_{u_i} = \{t_s, t_{s+1}, \ldots, t_r, \ldots, t_{e-1}, t_e\}$ in the class C_i; the user's time intervals are from t_s to t_e, satisfying the condition of $t_r \in T_{u_i}$. That is to say, the user u_i is assigned to class C_i, and she/he is allowed to obtain her/his class keys with the only time duration T_{u_i} satisfying $T_{u_i} \subseteq T$. Following this, the CAA generates the information $I(i, T_{u_i})$, which is used to identify the user u_i, as follows,

$$I(i, T_{u_i}) = (i_{u_i} \| t_s \| t_e), \qquad (2)$$

where $i_{u_i} \equiv (a_{u_i})^{\prod_{Cl \underline{\pi} Ci} d_l} (\text{mod}\, n)$, and the symbol "$\|$" denotes a concatenation operator to the fixed operands of i_{u_i}, t_s, and t_e.
Finally, the CAA then sends the user's information $I(i, T_{u_i})$ back to the user u_i.

2.5 Key derivation

2.5.1 Authentication phase

Assume a user u_i, associated with C_i, wants to derive the class key k_{i,t_r} at a period t_r, where $t_r \in T_{u_i}$. The user u_i first chooses a random prime number x_1 and a large prime n_1, where $t_z << n_1 < n$ and $\gcd(n_1, \phi(n)) = 1$, for a new request time-token phase. Subsequently, she/he computes the inverse of the number x_1, $y_1 \equiv (x_1)^{-1} (\text{mod}\, n_1)$, satisfying $x_1 \times y_1 \equiv 1\ (\text{mod}\, n_1)$, and sends the request information $(i_{u_i} \| t_r \| y_1 \| n_1)^{e_c} (\text{mod}\, n)$ to the CAA server in order to obtain the time-token back. For dealing with the request, the CAA server will compute firstly $\left((i_{u_i} \| t_r \| y_1 \| n_1)^{e_c}\right)^{d_c} \equiv (i_{u_i} \| t_r \| y_1 \| n_1)\ (\text{mod } n)$, and authenticate this request via checking whether the user information is legal or not, *i.e.*, by checking whether the equation $(i_{u_i})^{\prod_{Cv \underline{\pi} Ci} e_v} (\text{mod}\, n)$ is equal to a_{u_i} or not and validating whether these conditions $t_1 \leq t_r \leq t_z$, $t_r \in T_{ui}$ and $n_1 >> g_{t_r}$ are satisfying or not. If the user's request passes the user authentication and time-bound validation, then the CAA server will make an access record to related log and continue the next phases as the following subsections.

2.5.2 Time-token generation phase

If the request of the user u_i passes the user authentication and access-timing validation, then the CAA server will compute the time-token for accessing the class $C_j \underline{\pi} C_i$ at the period t_r as follows:

$$\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r}), \qquad (3)$$

where $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{d_i \times g_{tr}} (\text{mod}\, n)$ and $s_{u_i,t_r} \equiv y_1 \times g_{t_r} (\text{mod}\, n_1)$. Subsequently, the time-token will send back to the user for the class accessing $C_j \underline{\pi} C_i$ at the period t_r.

2.5.3 Key derivation phase

The user u_i can derive the class keys for class C_i and all of the partially ordered classes C_j at the valid period t_r, if the relation $C_j \underline{\pi} C_i$ remains in the hierarchy scheme. Key derivation processes are shown in detail in the following two cases. In *Case 1*, we show that the user u_i can derive the class key for class C_i at the period t_r; In *Case 2*, we show that the user u_i can also derive the class key for all of the partially ordered classes C_j, if, and only if, $C_j \underline{\pi} C_i$ in the hierarchy scheme at the period t_r.

Case 1 From the given information $I(i,T_{u_i}) = (i_{u_i} \| t_s \| t_e)$, a random prime number x_1 and the public parameters associated with authentication procedure at the period t_r which satisfies $t_r \in T_{u_i}$, the user u_i, associated with C_i, then obtains the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$ back from the CAA server and can derive the class key k_{i,t_r}. Thus, she/he can obtain $\hat{x} \equiv s_{u_i,t_r} \times x_1 \equiv g_{t_r} (\bmod n_1)$ and derive the class key k_{i,t_r} by

$$(i_{u_i})^{\hat{x}} \times r_{u_i,t_r} \equiv k_{i,t_r} (\bmod n). \qquad (4)$$

The following *Theorem 1* ensures the correctness of the class key derivation of *Case 1*.

Theorem 1 *If $I(i,T_{u_i}) = (i_{u_i} \| t_s \| t_e)$ and the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$ are the key information produced by the proposed scheme in (2) and (3), then the user u_i who is in the class C_i can derive the k_{i,t_r} by (4), where k_{i,t_r} is the class key associated with C_i at the period t_r.*

The proof of *Theorem 1* is showed in the appendix.

Case 2 When the user u_i succeeds in passing the user authentication and time-bound validation, the user will receive the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$ from the CAA server. Using the given information $I(u_i,T_{u_i}) = (i_{u_i} \| t_s \| t_e)$ and the public parameters, user u_i, who belongs to class C_i, can derive the key k_{j,t_r}, where the key is a revised time-dependent key of class C_j at a period t_r, if the class relation $C_j \underline{\pi} C_i$ still remains in this hierarchy scheme and the period t_r satisfies the condition $t_r \in T_{u_i}$, computed as:

$$k_{j,t_r} \equiv \left((i_{u_i})^{\hat{x}} \times r_{u_i,t_r} \right)^{\prod_{C_k \underline{\pi} C_i, C_k \underline{\not\pi} C_j} e_k} (\bmod n). \qquad (5)$$

Again, the following *Theorem 2* ensures the correctness of the class key derivation of *Case 2*.

Theorem 2 *If $I(i,T_{u_i}) = (i_{u_i} \| t_s \| t_e)$ and the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$ are the key information produced by the proposed scheme in (2) and (3), then the user u_i who is in the class C_i can obtain the k_{j,t_r} by (5), where k_{j,t_r} is the class key associated with C_j satisfying $C_j \underline{\pi} C_i$ at the period t_r satisfying $t_r \in T_{u_i}$.*

The proof of Theorem 2 is also showed in the appendix.

2.5.4 Performance of class key derivation

In general, the computation time on a modulus n for a modular exponentiation operation is about $O(|n|)$ times, where $|n|$ denotes the bit length of n, *e.g.*, its length is usually taken from 512 bits to 1024 bits [15]. In this subsection, we separate two cases to analyze the class key derivation phase in *Subsection 2.5.3*.

For *Case 1*, given the public parameters, information $I(i,T_{u_i}) = (i_{u_i} \| t_s \| t_e)$, where $i_{u_i} \equiv (a_{u_i})^{\prod_{C_l \underline{\pi} C_i} d_l} (\bmod n)$, and the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$, to derive k_{i,t_r}, the user u_i who is the membership of class C_i must compute $k_{i,t_r} \equiv (i_{u_i})^{\hat{x}} \times r_{u_i,t_r} (\bmod n)$ where

$\hat{x} \equiv s_{u_i,t_r} \times x_1 \equiv g_{t_r} \pmod{n_1}$, $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_u \underline{\pi} C_i} d_u} \pmod{n}$ and $s_{u_i,t_r} \equiv y_1 \times g_{t_r} \pmod{n_1}$. Only two modular multiplications and one modular exponential operation are required at the user's computation efforts. It is therefore computation saving in this presentation. See the computation requirements in the part of CAA server. There are $\xi + 1$ modular exponentiation operations, one inverse computation and one modular multiplications for deriving $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \underline{\pi} C_i} d_l} \pmod{n}$ and one modular multiplications for deriving s_{u_i,t_r} required, where ξ denotes a number of classes in which each class satisfies $C_l \underline{\pi} C_i$.

For *Case 2*, given the public parameters, information $I(i, T_{u_i}) = (i_{u_i} \| t_s \| t_e)$ and the time-token $\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r})$, the user u_i who is the membership of class C_i wants to derive k_{j,t_r}. She/he must compute $k_{j,t_r} \equiv \left((i_{u_i})^{\hat{x}} \times r_{u_i,t_r}\right)^{\prod_{C_k \underline{\pi} C_i, C_k \not\underline{\pi} C_j} e_k} \pmod{n}$, where $\hat{x} \equiv s_{u_i,t_r} \times x_1 \equiv g_{t_r} \pmod{n_1}$, $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \underline{\pi} C_i} d_l} \pmod{n}$ and $s_{u_i,t_r} \equiv y_1 \times g_{t_r} \pmod{n_1}$. Consider the computations in CAA server. The computing costs of $\xi + 1$ modular exponentiation operations, one inverse computation and one modular multiplications for deriving $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \underline{\pi} C_i} d_l} \pmod{n}$ are required, where ξ is a number of classes satisfying $C_l \underline{\pi} C_i$; Besides, CAA needs to spend one extra modular multiplications for deriving s_{u_i,t_r}. On the other hand, in the way of user's efforts, two modular multiplications and $\eta + 1$ modular exponential operation are only required, where η is a number of classes satisfying $C_k \underline{\pi} C_i$ and $C_k \not\underline{\pi} C_j$. It can be seen that the computing efforts are rather efficient in our solution.

2.6 Dynamic change of classes during a valid period

2.6.1 Adding a class

Suppose that a new class C_{m+1} is added to the existing system within a valid period t_r. In response to this change event, the CAA server will randomly chooses a small integer e_{m+1} which is related to $\phi(n)$, and derives d_{m+1}, such that $e_{m+1} \times d_{m+1} \equiv 1 \pmod{\phi(n)}$. Further, the CAA server will remain the class key k_{i,t_r} computed by (1) for the class C_i if the class relation satisfies $C_{m+1} \not\pi C_i$ after the new class C_{m+1} is added, and update the class key for the class if the class relation satisfying $C_{m+1} \underline{\pi} C_i$, computed as

$$k_{i,t_r} \equiv \left[(a_{t_r})^{g_{t_r} \prod_{C_l \underline{\pi} C_i, C_l \neq C_{m+1}} d_l} \right]^{d_{m+1}} \pmod{n}, \tag{6}$$

by multiplying a new secret power parameter d_{m+1} and allot an adaptable time-token for any upcoming user's access request. That is, the adaptable time-token will serve to the user who remains in the classes satisfying $C_{m+1} \underline{\pi} C_i$ for the period t_r. After this change, if a user u_i, where she/he still remains in the class C_i satisfying $C_{m+1} \underline{\pi} C_i$ and her/his user information $I(u_i, T_{u_i})$ doesn't need to be upgraded, activates a new access request in the period t_r, the CAA server will generate and send back the adaptable time-token,

$$\tau_{u_i,t_r} = (r_{u_i,t_r} \| s_{u_i,t_r}), \tag{7}$$

where

$$r_{u_i,t_r} \equiv \left[\left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \underline{\pi} C_i, C_l \neq C_{m+1}} d_l} \right]^{d_{m+1}} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \underline{\pi} C_i} d_l} \pmod{n}, \text{and } s_{u_i,t_r} \equiv (y_1 \times g_{t_r}) \times d_{m+1} \equiv y_1 \times d_{m+1} \times g_{t_r} \pmod{n_1}.$$

Every remained user in the system will still keep her/his own user information, which had assigned after the class C_{m+1} is added. According to *Theorem 1* and *Theorem 2*, the user u_i can derive the keys k_{i,t_r} and k_{j,t_r}, by using her/his user information, and the adaptable time-token computed by (7), if the class relation $C_{m+1} \underline{\pi} C_i$ is satisfied and the period t_r satisfies the condition $t_r \in T_{u_i}$.

2.6.2 Deleting a class

Suppose that a class C_m is deleted from the existing system at the period t_r. The CAA server will remain the class key k_{i,t_r} computed by (1) for the original class C_i if the class relation satisfied $C_m \not\underline{\pi} C_i$ before the class C_m is deleted, and update the class key by (8) for the original class if the class relation satisfying $C_m \underline{\pi} C_i$ before the class C_m is deleted, computed as

$$k_{i,t_r} \equiv \left[\left(a_{t_r}\right)^{g_{tr} \prod_{Cl \underline{\pi} Ci} dl} \right]^{e_m} \equiv \left(a_{t_r}\right)^{g_{tr} \prod_{Cl \underline{\pi} Ci, Cl \neq Cm} dl} \pmod{n} \tag{8}$$

via multiplying the power number e_m which is the inverse number of the parameter d_m and allot an adaptable time-token for any upcoming user's access request. The CAA server only updates the time-tokens corresponding to the new related upcoming user's access request, which serve to the influenced classes; the initial class keys and all of the existing user

$$r_{u_i,t_r} \equiv \left[\left(\left(a_{u_i}\right)^{-1} \times a_{t_r} \right)^{g_{tr} \times \prod_{Cl \underline{\pi} Ci} d_l} \right]^{e_m} \equiv \left(\left(a_{u_i}\right)^{-1} \times a_{t_r} \right)^{g_{tr} \times \prod_{Cl \underline{\pi} Ci, Cl \neq Cm} d_l} \pmod{n},$$

information do not need to reconstruct anymore. Soon after that, the CAA server rejects all access requests for the class C_m by generating the new time-token in which the secret parameter d_m will be discarded. That is, the CAA will generate the new time-token $\tau_{u_i,t_r} = \left(r_{u_i,t_r} \,\|\, s_{u_i,t_r}\right)$, where
and $s_{u_i,t_r} \equiv y_1 \times e_m \times g_{t_r} \pmod{n_1}$, for all users who belong to the rest classes for satisfying $\forall C_l \underline{\pi} C_i$ and $C_l \neq C_m$.

2.6.3 User change

Suppose that a user u_i left unexpectedly from her/his responsibility in the system at the period t_r within her/his pre-assigned authorized period set. Although the user u_i activates an access request in the time belonging to those periods in pre-assigned authorized period set, the CAA server will reject to distribute any time-token if the user u_i has left.

2.7 Activate a proactive phase for a next new period

In this phase, the CAA will update all class keys for activating a new proactive phase for a periodical period t_{r+1}. The CAA computes the new class key $k_{i,t_{r+1}}$ by refreshing both the prime numbers: $g_{t_{r+1}} \in \{g_{t_1}, g_{t_2}, \ldots, g_{t_{r+1}}, \ldots, g_{t_z}\}$ instead of g_{t_r}, and $a_{t_{r+1}}$ instead of a_{t_r} for the class C_i from the corresponding initial key $k'_{i,t_{r+1}}$, where $i \neq 0$, $i \geq 1$, and $t_{r+1} > t_r$ as follows,

$$k_{i,t_{r+1}} \equiv \left(k'_{i,t_{r+1}}\right)^{g_{tr+1}} \equiv \left(a_{t_{r+1}}\right)^{g_{tr+1} \prod_{Cl \underline{\pi} Ci} dl} \equiv \left(a_{t_{r+1}}\right)^{g_{tr+1} \times (d_1 \times d_2 \times \ldots \times d_m) \times \prod_{Cv \not\underline{\pi} Ci} e_v} \pmod{n} \tag{9}$$

where the prime number $g_{t_{r+1}}$ is kept secret by the CAA server.

We assume that a user u_d who belongs to the class C_i satisfying $C_j \underline{\pi} C_i$ at the period t_{r+1} activates a secure access request and if the request passes the user authentication and access-timing validation, then the CAA server will compute the new time-token for further accessing the class $C_j \underline{\pi} C_i$ as follows. $\tau_{u_d,t_{r+1}} = \left(r_{u_d,t_{r+1}} \,\|\, s_{u_d,t_{r+1}}\right)$, where $r_{u_d,t_{r+1}} \equiv \left(\left(a_{u_d}\right)^{-1} \times a_{t_{r+1}} \right)^{\prod_{Cl \underline{\pi} Ci} d_l \times g_{tr+1}} \pmod{n}$ and $s_{u_{d+1},t_r} \equiv y_1 \times g_{t_{r+1}} \pmod{n_1}$. Subsequently, the adaptable time-token will send back to the user u_d. Then, the user u_d can derive (9) according to *Theorem 1*, and derive $k_{j,t_{r+1}}$ by using *Theorem 2.*

3. Discussions

In this section, we discuss the impacts of applying proactive mechanism to conventional UHAC scheme, and make some performance comparisons between the references [8, 16] and our scheme. The details are showed as follows.

3.1 Provision of proactive cryptographic key assignment scheme

The cryptographic key assignment in our proposed scheme is conditionally constrained upon the concept of period so that the class key is corresponding to a distinct time-token, which is secretly and frequently updated by the CAA. In other words, the cryptographic key assignment of revised time-dependent feature scheme provides the proactive management mechanism of the key assignment. That is to say, the proactive class keys are periodically updated through the change of associated time-tokens. In this way, the attempt of breaking the class key can be efficiently deterred. The CAA actively changes all of the class keys, by providing periodically refreshing time tokens for each period that have time-dependent properties, in which there are corresponding and distinct parameters, r_{u_i,t_r} and s_{u_i,t_r}, to send to users who have invoked access requests for the class keys. Assume that a legal user u_i in class C_i is assigned the information $I(i,T_{u_i})$, which has the initial identity i_{u_i} with an authorized time from t_s to t_e. After authentication, the user can obtain the time-token τ_{u_i,t_r}, comprised of the parameters r_{u_i,t_r} (with the secret product of $(a_{u_i})^{-1} \times a_{t_r} \bmod n$) and s_{u_i,t_r}. Then the user u_i can both derive the class key k_{i,t_r} by (4) and the class key k_{j,t_r} by (5) if, and only if $C_j \underline{\pi} C_i$ and $t_r \in T_{u_i}$. Thus, we can prove that all of the proactive class keys can be changed by periodically updating the time-tokens during a period, distributed by the CAA. In this way, our scheme can actually provide a proactive mechanism for a cryptographic key assignment scheme in a hierarchy.

3.2 Our scheme made a great impact on UHAC

As conventional UHAC is unable to efficiently reflect the user's expired employment time, or provide refreshment of secret class keys, the impact of a proactive mechanism in a conventional UHAC scheme is discussed below.

(i) Periodically refreshing class keys, by changing the time-tokens in a UHAC scheme, means that users have nothing to do with class keys, in the access control mechanism of the hierarchy.

(ii) Changing the time-tokens in a UHAC scheme allows more flexibility. Time-tokens with allocated rights and obligations can be controlled by the CAA. Our scheme need not change any initial parameters or published information when classes are either increased or decreased.

(iii) The impact of our scheme, on the security of conventional UHAC mechanisms in a hierarchy, is as follows:

a. Users cannot touch the primes of the time-dependent secret parameters, such as a_{u_i} and a_{t_r}, which effectively prevents the revealing of secret class key.

b. Users' access behaviors can be kept by the CAA server in our scheme, through request records. If any disturbance or leakage occurs, the records will assist in pinpointing the responsibility.

3.3 Comparisons

Next, we consider the comparisons of computation time for the class key derivation phase from a view of user-end among our scheme and the two schemes [8] and [16] respectively.

For convenience at comparisons, we assume that the comparisons are bounded to the period t_r for deriving a class key by a valid user. Therefore, the derivation of a class key k_{j,t_r} in computations is summarized in Table 1.

Table 1. Comparisons of a class key k_{j,t_r} derivation

Schemes / **Comparison Items**	Tzeng's scheme [16]	Huang et al.'s scheme [8]	Our scheme
Number of modular exponentiation computations	$t_e - t_s + \eta$	$z - t_r + \eta$	$\eta + 1$
Number of modular multiplication computations	0	0	2
Number of Lucas operations computations	$t_e - t_s$	0	0
Number of operations $f(\cdot)$ function computations	0	$z - t_r$	0
Number of hashing computations	1	0	0

Note: we assume that a time bound, which is from the start time t_s to the end time t_e, is represented as (t_s, t_e), and the period t_r satisfies the condition, $t_r \in T_{u_i}$.

In Tseng's scheme [16], according to the information $I(i,t_s,t_e)=(x,y)$ and the public parameters $g_1,g_2,f_1,f_2,e_1,e_2,\ldots,e_m,n_1,n_2$, the user must take the computations of $(x)^{g_1^{te-tr} g_2^{tr-ts} \prod_{C_k \underline{\pi} C_i, C_k \underline{\not\pi} C_j} e_k} \equiv \alpha \pmod{n_1}$ and $V_{f_1^{te-tr} f_2^{tr-ts}}(y) \equiv \beta \pmod{n_2}$. Upon the observation, the computations of α and β, $t_e - t_s + \eta$ modular exponentiations operations and $t_e - t_s$ Lucas operations are required, respectively, where η is the number of classes satisfying $C_k \underline{\pi}\ C_i$ and $C_k \underline{\not\pi}\ C_j$. Therefore, the computing cost of $k_{j,t_r} = H(\alpha,\beta)$ needs $t_e - t_s + \eta$ modular exponentiations, $t_e - t_s$ Lucas operations and a hash operation of $H(\cdot)$ function, respectively.

On the other hand, in Huang *et al.*'s scheme [8], the information

$$I(i, \mathrm{z}) \equiv \left(a_{t_r}^{\prod_{C_j \underline{\pi} \mathrm{C_i}} d_j} \right)^{\prod_{k=z}^{N} f(k)} (\mathrm{mod}\ \ n)$$

and some public parameters, where $f(\cdot)$ is simple function randomly provided by CAA *e.g.* $f(t_r) = 3t_r + 7$, in order to derive the class key $k_{j,t_r} \equiv (I(i,z))^{\prod_{C_k \underline{\pi} C_i, C_k \underline{\not\pi} C_j} e_r \times \prod_{tr=1}^{z-1} f(t_r)} (\mathrm{mod}\ n)$, the computation of $I(i,z)^{\prod_{C_k \underline{\pi} C_i, C_k \underline{\not\pi} C_j} e_k \times \prod_{t_r=t}^{z-1} f(t_r)} (\mathrm{mod}\ \ n)$ is required, where $C_j \underline{\pi}\ C_i$ and $t_r \le z$. In this expression, obviously, there are $z-t_r+\eta$ modular exponentiation operations and $z-t_r$ operations of function $f(\cdot)$ required, where η is a number of classes satisfying $C_k \underline{\pi}\ C_i$ and $C_k \underline{\not\pi}\, C_j$.

Finally, examine our scheme. In our scheme, when giving the public parameters, information of $I(i, T_{u_i}) = (i_{u_i} \parallel t_s \parallel t_e)$ and the time-token of $\tau_{u_i,t_r} = (r_{u_i,t_r} \parallel s_{u_i,t_r})$, the computing cost for a user in C_i deriving the key of

$$k_{j,t_r} \equiv \left((i_{u_i})^{(s_{u_i,t_r} \times x_1 \bmod n_1)} \times r_{u_i,t_r} \right)^{\prod_{C_k \underline{\pi} C_i, C_k \underline{\not\pi} C_j} e_k} (\bmod n),$$

where $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r} \right)^{g_{t_r} \times \prod_{c_u \underline{\pi} c_i} d_u} (\bmod n)$ and $s_{u_i,t_r} \equiv y_1 \times g_{t_r} (\bmod n_1)$, two modular multiplications and $\eta+1$ modular exponential operation are required, where η is a number of classes that satisfy $C_k \underline{\pi}\, C_i$ and $C_k \underline{\not\pi}\, C_j$.

4. Conclusions

In this paper, we have proposed an adaptable solution for cryptographic key assignment scheme with adaptable time-token constraint, which is more secure and effective than a UHAC scheme. A new technique is proposed in the paper that is used to adapt the changeability in dynamic scenarios of the UHAC scheme, *e.g.*, the scenarios of the dynamic change examples of class during a valid period shown in *Subsection 2.6.1* and *Subsection 2.6.2*, the scenarios of the user change shown in the *Subsection 2.6.3*. All class keys will be updated proactively by the concept of proactive key management, which we show it in *Subsection 2.7*; in this way, it can make the advantage that is proven and free from the scenario of the collusive attack example which proposed by Yi *et al.* in [18]. That is to say, our scheme can achieve more secure and more efficient dynamic management for UHAC scheme.

References

[1] S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM Transactions on Computer Systems*, vol. 1, no.3, pp. 239–248, 1983.

[2] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, "Cryptography: Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems," *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Nov. 2002.

[3] R. Canetti, A. Gennaro, Herzberg, and D. Naor, "Proactive Security: Long-Term Protection against Break-ins," *CryptoBytes*, vol. 3, no. 1, pp. 1–8, 1997.

[4] C. C. Chang, R. J. Hwang, and T. C. Wu, "Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," *Information Systems*, vol. 17, no.3, pp. 243–247, 1992.

[5] H. Y. Chien, "Efficient Time-Bound Hierarchical Key Assignment Scheme," *IEEE Transactions On Knowledge And Data Engineering*, vol. 16, no. 10, pp. 1301–1304, Oct. 2004.

[6] Y. Frankel, P. Gemmel, P. D. MacKenzie, and M. Yung, "Proactive RSA," *Advances in Cryptology Crypto`97 Proceedings on Lecture Notes in Computer Science*, Springer Verlag, pp. 1294–1308, 1997.

[7] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive Public Key and Signature Systems," *Proceedings of the 4th ACM Conference on Computers and Communication Security*, pp. 100–110, April 1997.

[8] H. F. Huang and C. C. Chang, "A New Cryptographic Key Assignment Scheme with Time-constraint Access Control in a Hierarchy," *Computer Standards & Interfaces*, vol. 26, issue 3, pp. 159–166, May 2004.

[9] F. H. Kuo, V. R. L. Shen, T. S. Chen, and F. Lai, “Cryptographic Key Assignment Scheme for Dynamic Access Control in a User Hierarchy,” *IEE Proceedings on Computers and Digital Techniques*, vol. 146, no. 5, pp. 235–240, Sept. 1999.

[10] S. J. Mackinnon, P. D. Taylor, H. Heijer, and S. G. Akl, “An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy,” *IEEE Transactions on Computers* , vol. 34, no. 9, pp. 797–802, 1985.

[11] B.M. Macq and J.-J. Quisquater, “Cryptology for Digital TV Broadcasting,” *Proc. IEEE*, vol. 83, no. 6, pp. 944–957, 1995.

[12] A. D. Santis, A. L. Ferrara and B. Masucci, “Enforcing the security of a time-bound hierarchical key assignment scheme,”*Information Sciences, in Press, Corrected Proof*, *Available Online*, 9 August 2005.

[13] Q. Tang and C. J. Mitchell, “Comments on a Cryptographic Key Assignment Scheme,” *Computer Standards & Interfaces*, vol. 27, pp. 323–326, 2005.

[14] R. Ostrovsky and M. Yung, “How to Withstand Mobile Virus Attacks,” *ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 51–61, 1991.

[15] G. J. Simmons, “Contemporary Cryptology: The Science of Information Integrity,” *IEEE Press*, N. Y., 1992.

[16] W. G. Tzeng, “A Time-bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 182–188, Feb 2002.

[17] S. Y. Wang, C. S. Laih, “A Time-bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,” *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 1, pp. 91 – 100, Jan.-Mar. 2006.

[18] X. Yi and Y. Ye, “Security of Tzeng’s Time-bound Key Assignment Scheme for Access Control in a Hierarchy,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 1054–1055, August 2003.

Appendixes

The proof of *Theorem 1*:

Since $i_{u_i} \equiv (a_{u_i})^{\prod_{C_l \preceq C_i} d_l} \pmod{n}$, $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l} \pmod{n}$, $s_{u_i,t_r} \equiv y_1 \times g_{t_r} \pmod{n_1}$ and $n_1 >> g_{t_r}$. Then we have

$$\begin{aligned}
(i_{u_i})^{\hat{x}} \times r_{u_i,t_r} &\equiv (i_{u_i})^{(s_{u_i,t_r} \times x_1 \bmod n_1)} \times r_{u_i,t_r} \pmod{n} \\
&\equiv \left((a_{u_i})^{\prod_{C_l \preceq C_i} d_l}\right)^{((g_{t_r} \times y_1 \times x_1) \bmod n_1)} \times \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l} \pmod{n} \\
&\equiv \left(a_{u_i} \times (a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l} \pmod{n} \\
&\equiv k_{i,t_r} \pmod{n}.
\end{aligned}$$

Thus the theorem.

The proof of *Theorem 2*:

Since $i_{u_i} \equiv (a_{u_i})^{\prod_{C_l \preceq C_i} d_l} \pmod{n}$, $r_{u_i,t_r} \equiv \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l} \pmod{n}$, $s_{u_i,t_r} \equiv y_1 \times g_{t_r} \pmod{n_1}$, $n_1 >> g_{t_r}$ and some public parameters. Then we have

$$\begin{aligned}
\left((i_{u_i})^{\hat{x}} \times r_{u_i,t_r}\right)^{\prod_{C_k \preceq C_i, C_k \npreceq C_j} e_k} &\equiv \left((i_{u_i})^{(s_{u_i,t_r} \times x_1 \bmod n_1)} \times r_{u_i,t_r}\right)^{\prod_{C_k \preceq C_i, C_k \neq C_j, C_k \npreceq C_j} e_k} \pmod{n} \\
&\equiv \left(\left(a_{u_i}^{\prod_{C_l \preceq C_i} d_l}\right)^{((g_{t_r} \times y_1 \times x_1) \bmod n_1)} \times \left((a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l}\right)^{\prod_{C_k \preceq C_i, C_k \neq C_j, C_k \npreceq C_j} e_k} \pmod{n} \\
&\equiv \left(\left(a_{u_i} \times (a_{u_i})^{-1} \times a_{t_r}\right)^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l}\right)^{\prod_{C_k \preceq C_i, C_k \neq C_j, C_k \npreceq C_j} e_k} \pmod{n} \\
&\equiv \left((a_{t_r})^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l}\right)^{\prod_{C_k \preceq C_i, C_k \neq C_j, C_k \npreceq C_j} e_k} \pmod{n} \\
&\equiv \left((a_{t_r})^{g_{t_r} \times \prod_{C_l \preceq C_i} d_l}\right)^{\prod_{C_k \preceq C_i, C_k \neq C_j, C_k \npreceq C_j} e_k} \pmod{n} \\
&\equiv (a_{t_r})^{g_{t_r} \times \prod_{C_u \preceq C_j} d_u} \pmod{n} \\
&\equiv k_{j,t_r} \pmod{n}.
\end{aligned}$$

Thus the theorem.

Authors

Hsing-Chung Chen was born in Taiwan, 1966. He received the B.S. degree in Electronic Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 1994, and the M.S. degree in Industrial Education from National Normal University, Taipei, Taiwan, in 1996, respectively. He received the Ph.D. degree in Electronic Engineering from National Chung Cheng University, Chia-Yi, Taiwan, in 2007. During the years 1991-2007, he had served as a System Engineer at the Department of Mobile Business Group, Chunghwa Telecom Co., Ltd. From February 2008–present, he has been the Assistant Professor of the Department of Computer Science and Information Engineering at Asia University, in Taichung County of Taiwan. Currently, he is interested in researching Multi-session Cryptography, Role-based Access Control, Fuzzy Control, Grey Theoretic, and Wireless Communications. He is a member of the Chinese Cryptology and Information Security Association (CCISA). He is also a member of the International Fuzzy System Association (IFSA), the member of the Chinese Grey Systems Association. He joins the international committee on International Conference on Convergence and Hybrid Information Technology (ICCIT) series. Now, he is also the reviewer of the *IET Communications* (formerly *IEE Proceedings Communications*).

Shiuh-Jeng Wang was born in Taiwan, 1967. He received the M.S. degree in Applied Mathematics from National Chung-Hsing University, Taichung, Taiwan, in 1991. He received his PhD degree in Electrical Engineering at National Taiwan University, Taipei, Taiwan in 1996. He is currently with Dept. of Information Management at Central Police University, Taoyuan, Taiwan, where he directs the Information Cryptology and Construction Laboratory (ICCL, http://hera.im.cpu.edu.tw). He was a recipient of the 5th Acer Long-Tung Master Thesis Award and the 10th Acer Long-Tung Ph.D Dissertation Award in 1991 and 1996, respectively. Dr. Wang was a visiting scholar of Computer Science Dept. at Florida State University (FSU), USA in 2002 and 2004. He also was a visiting scholar of Dept. of Computer and Information Science and Engineering at University of Florida (UF) from Aug. 2004 to Feb. 2005. He served the editor-in-chief of the *Communications of the CCISA* in Taiwan from 2000-2006. He has been elected as the Panel Director of Chinese Cryptology and Information Security Association (CCISA) since Sept. 2006. Dr. Wang academically toured the CyLab with School of Computer Science in Carnegie Mellon University, USA, in Jan. 2007 for international project collaboration inspection. He is also the author/co-author of six books (in Chinese versions): *Information Security*, *Cryptography and Network Security*, *State of the Art on Internet Security and Digital Forensics, Eyes of Privacy –Information Security and Computer Forensics, Information Multimedia Security,* and *Computer Forensics and Digital Evidence* published in 2003, 2004, 2006, and 2007, respectively. He is a full professor and a member of the IEEE, ACM. His current interests include information security, digital investigation and computer forensics, steganography, cryptography, data construction and engineering.

Jyh-Horng Wen received the B.S. degree in electronic engineering from the National Chiao Tung University, Hsing-Chu, Taiwan, in 1979 and the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, in 1990. From 1981 to 1983, he was a Research Assistant with the Telecommunication Laboratory, Ministry of Transportation and Communications, Chung-Li, Taiwan. From 1983 to 1991, he was a Research Assistant with the Institute of Nuclear Energy Research, Taoyun, Taiwan. From February 1991 to July 2007, he was with the Institute of Electrical Engineering, National Chung Cheng University, Chia-Yi, Taiwan, first as an Associate Professor and, since 2000, as a Professor. He was also the Managing Director of the Center for Telecommunication Research, National Chung Cheng University, from Aug. 2001 to July 2004 and the Dean of General Affairs, National Chi Nan University, from Aug. 2004 to July 2006. Since Aug. 2007, he has been the Department Head of Electrical Engineering, Tunghai University, Taichung, Taiwan. He is an Associate Editor of the *Journal of the Chinese Grey System Association.* His current research interests include computer communication networks, cellular mobile communications, personal communications, spread-spectrum techniques, wireless broadband systems, and gray theory. Prof. Wen is a member of the IEEE Communication Society, the IEEE Vehicular Technology Society, the IEEE Information Society, the IEEE Circuits and Systems Society, the Institute of Electronics, Information and Communication Engineers, the International Association of Science and Technology for Development, the Chinese Grey System Association, and the Chinese Institute of Electrical Engineering.