Computers & Security Vol.20, No.7, pp.612-619, 2001

0167-4048/01$20.00

Hierarchical key assignment without public-key cryptography

Chu-Hsing Lin

Department of Computer Science and Information Engineering, Tunghai University, Taichung, 407 Taiwan, R.O.C.
E-mail: chlin@mail.thu.edu.tw

Abstract

VLSI chips make possible the hardware devices employed in today's computing environment for security functions. Controlling access in a hierarchy is an interesting research topic in computer security. Many investigations have been published in the literature with solutions involving assigning cryptographic keys to users at different access clearance levels. However, the existing schemes require a large number of costly arithmetic operations with large integers. This type of system is difficult to implement in a chip with lower computation ability. In this paper, we present a solution, suitable for a low cost chip, to the hierarchical control problem. The proposed scheme has promising characteristics such as high computational efficiency, little required memory in the chip and low cost implementation. This method possesses all of the dynamic properties that appear in existing methods.

Keywords: multilevel data security, partially ordered set, access control, key assignment, cryptographic key, user hierarchy, one-way hash functions.

1. Introduction

In a multi-user computing environment, the problem of access control to system resources is an important research topic. Much emphasis has been placed on the situation where users are classified into several privilege classes and organized into a hierarchical structure. In these applications, system resources, e.g., data, programs, files, require management and categorization into multiple security levels. A secure access control mechanism is designed such that users are authorized and classified in association with the various security levels for accessing resources. In other words, a user classified into a higher-privileged class will have access to the data or programs created or owned by a user in a lower-privileged class. Lower classified users do not have access to higher classified system resources.

Many real applications for access control schemes in a hierarchical structure can be found in the studies on multi-level databases security [1,2,3,6], applications to multilevel secure operating systems [15,16] and networks security policy [4,5,14].

We assumed that in a computing system with users classified into n disjoint privilege classes, $S=\{1, 2,\ldots, n\}$. S is a partially ordered set under a binary relation denoted by "$\prec$". As stated previously, $i \prec j$ means that j is a higher-privileged user than i. In other words, user j will have the right to access files, data, or programs owned or created by user i while the opposite is forbidden by the system access rules.

On the basis of a cryptographic function, an access control mechanism for security requirement is implemented. From this point of view, if each user is assigned a cryptographic key for encrypting and decrypting resources, then any access to the system can be controlled under a specific policy.

Further, for the purpose of multi-level data management, additional functions should be incorporated. Let k_i be the cryptographic key assigned to user i. The problem is how to assign the secret key k_i to each user i such that if $i \prec j$ then k_i should be easily derived from k_j, while it is infeasible to derive k_j given k_i.

Akl et. al [7,8] proposed several elegant schemes for assigning cryptographic keys to security classes for users organized in a hierarchical access structure. A trusted agent (TA, for short) is assumed to exist for generating and distributing keys. There are two common drawbacks with their methods. New keys must be recomputed for existing classes when a new user is added. The memory storage size needed to maintain the parameters grows dramatically as the number of user's increases. Other research results to improve the existing key-assignment schemes can be found in [9,10,11,12,19,22]. For related results involving access control problem in a hierarchical structure, one can consult [17,18,19,20-21,27].

When VLSI chips became available, hardware devices could be designed for security functions. Based on tamper-proof hardware, Leighton and Micali [24] and Zheng [25] proposed new key-agreement protocols. Their protocols were aimed at communication scenarios such as the one performed in the Clipper Chip. Only simple operations such as concatenations and one-way hash functions, executed efficiently in devices with lower computing ability, were required.

Based on the above concept and inspired by Leighton et. al and Zheng's idea, we solved the problem of controlling access in a hierarchy using tamper-proof chips. Though there are many results published in the literature to solve this problem using various level cryptographic keys for users with different access clearances. However, the existing schemes require a large number of costly arithmetic operations with large integers. This is difficult implement in a chip with lower computation ability. In this paper, we propose an efficient dynamic mechanism for access control in a hierarchical structure. The proposed scheme is based on very simple operations and can be implemented in a low cost chip. Dynamic operations such as changing a secret key, inserting a user, or deleting an existing user etc., can be fulfilled efficiently. Furthermore, it is easy for an ancestor to derive the secret key of his/her grandchild. In the next section, we briefly describe the problem of hierarchical control with keys. In Section 3, we present the new scheme. The cost measurement and the security analysis appear in Section 4. Our conclusions are presented in the last section.

2. Hierarchical Control with Keys

The problem of access control in a hierarchy for classified data management can be modeled as a directed acyclic graph. Let G=(V, E) represent the directed acyclic graph, where V indicates the set of users with various security clearances and E indicates the set of directed branches. The access control problem in a hierarchy can also be modeled as a partially ordered set S with a binary relation "$\prec$". We assumed that i and j are two elements in S. Let users correspond to vertices and their relationship specified using a directed branch. That is, there exists a directed edge from vertex j to vertex i if and only if $i \prec j$. Furthermore, a directed acyclic graph can be simplified by eliminating all of the edges that can be implied by the transitive closure property. The proposed solution is constructed according to the simplified directed acyclic graph.

Suppose that k_i is the secret key of user i. Essentially, a cryptographic transformation $H(a,b,x)$, with some parameters a and b, are constructed in such a way that $k_i = H(a,b,k_j)$ can easily be derived if $i \prec j$. However, it is computationally infeasible to compute k_j reversibly knowing a, b and k_i.

On the basis of this cryptographic transformation with elegant characteristics, the solution to the hierarchical access control problem is fulfilled. Some

assumptions are made in the model. A trusted third party, called the trusted agent (TA), is assumed to exist. The role of the TA is to create the cryptographic transformation H(*a,b,x*) for the keys of any two comparable users. In other words, initially the TA will take the responsibility to compute some relation parameters for all users.

In 1993, Leighton and Micali [24] proposed two secret-key agreement schemes without public-key cryptography. Later, Zheng [25] improved some weaknesses in their scheme. These schemes are based on very simple operations such as concatenation and one-way hash functions and implemented in tamper-proof VLSI chips. These operations can be done efficiently in low-computation-power devices. We shall construct our dynamic access control scheme based on simple operations that can be conducted in a tamper-proof chip environment.

3. The proposed Scheme

In this section, key generation and key derivation procedures for the proposed scheme will be described. Because one-way hash functions [26] play an important role in our scheme, the background of one-way hash functions will be briefly reviewed. A one-way hash function, h(M), operates on an arbitrary-length message M and returns a fixed-length hash value y. One-way hash functions have the following characteristics: (1) it is easy to compute y=h(M) by giving M; (2) it is difficult to obtain M by having the value y; and (3) given M, it is difficult to compute another message M' such that h(M)=h(M'). Note that provably-secure hashes with modular exponentiation are not included in the one-way hash functions we discuss in this paper.

With these in mind, we are going to introduce our scheme. This scheme is based on the following three assumptions (as employed in the Leighton-Micali and Zheng's schemes [24,25]):

(1) There exits a cryptographically strong one-way hash (or pseudo-random) function h;
(2) Tamper-resistant VLSI chips are available.
(3) There is a trusted agent in the system.

Initially, the agent picks up a m-bit random number X and keeps it secret. Note that in order to prevent the exhaustive search attack, m should be sufficiently large, e.g. m 100. Afterwards, the TA generates n personal secret keys, indicated as $k_1, k_2 \ldots, k_n$, each k-bit, for each individual user in the system. Typically, we can choose k 64, say k=64, 128 or 256, for security considerations.

At the registration stage, the agent customizes the tamper-proof chip for the user i by injecting the random number X and his/her identity ID_i into the chip. The agent then hands user i the chip and his/her personal secret key k_i, respectively. Note that the random number X is common for all users in the system while X should never be seen by anyone except the agent.

For dynamic key management, the agent uses Eq (1) to compute a relation parameter, indicated as (r_{ji}, n_{ji}), for each directed branch connecting from user j to user i with $i \prec j$. These relation parameters are then put onto a public board such that all users can read it or in a way that allows only legitimate users in the system to access it (e.g. by providing a password). However, when these relation parameters are accessed, some authentication process may be added to verify the values if necessary. Note that, as mentioned in Section 2, we need merely to consider the simplified graph. Only a single-edged branch appearing in the simplified graph is assigned a relation parameter, respectively.

$$\begin{cases} r_{ji} = h(X \| ID_j \| k_j) \oplus h(X \| ID_i \| k_i) \oplus k_j \oplus k_i \\ n_{ji} = h(X \| ID_i \| h(X \| ID_j \| k_j)) \oplus h(X \| ID_i \| k_i) \end{cases} \quad \text{Eq(1)}$$

We would like to point out that the user identities ID's in Eq(1) have implicit strength, as discussed in Zheng's paper [25]. User names (or identities) are more directly involved in the relation parameter. Based on the pseudo-randomness of the one-way hash function h, the probability that two different pairs of users are assigned an identical relation parameter is negligible. With this property, Zheng proposed a new key agreement protocol to remove the flaw in the previous work by Leighton-Micali [24]. Note that "⊕" indicates the bit-wise XOR operation, "||" is the

concatenation and h is a one-way hash function. There are several one-way hash functions, e.g. SHA, SHA-1, MD5, GOST Hash, etc. available in the literature [26]. Each of these one-way hash functions has variable-length input and fixed-length output. However, they have different output lengths. For instance, the output length of SHA, MD5, and GOST Hash is 160 bits, 128 bits and 256 bits, respectively.

Key derivation

To derive the secret key of a lower-privileged user i, user j, where $i \prec j$, merely needs to present i's identity ID_i, the relation parameter (r_{ji}, n_{ji}) and his/her personal secret key k_j to the chip. The chip then outputs user i's secret key k_i by computing the following equation.

$$k_i = h(X \| ID_j \| k_j) \oplus h(X \| ID_i \| h(X \| ID_j \| k_j)) \oplus n_{ji} \oplus k_j \oplus r_{ji} \quad \text{Eq(2)}$$

Proposition 1. If $i \prec j$ then the secret key of user i can be computed by applying user j's personal key and the relation parameter to Eq(2) employed in the chip.

With the derived secret key, the legitimate user will have the same access right to the resources as user i. It is very convenient for an ancestor to deduce the key of his/her grandchild. For illustration, examine the following example. Imagine that if j has a child namely l and l has a child indicated as i; that is, j is the grandfather of i in the hierarchical graph. Now user j is required to compute the secret key of his/her grandchild i. In this case, on getting the two relation parameters, namely (r_{jl},n_{jl}) and (r_{li},n_{li}), from the public board, user j provides them to the chip. The chip can then compute the required (r_{ji},n_{ji}) by employing the following equation. This equation can be computed

$$\begin{cases} r_{ji} = r_{jl} \oplus r_{li} \\ n_{ji} = h(X \| ID_i \| h(X \| ID_j \| k_j)) \oplus n_{li} \oplus h(X \| ID_i \| (n_{jl} \oplus h(X \| ID_l \| h(X \| ID_j \| k_j)))) \end{cases} \quad \text{Eq(3)}$$

After computing the value of (r_{ji},n_{ji}) and having the secret key k_j, the chip can derive and output the secret key k_i as discussed previously. With this characteristic, the cost to retain the relation parameters can be reduced dramatically. However, it is a trade-off between memory and computation time. Further, Eq(3) can be proven using the following theorem.

Proposition 2. If $i \prec l \prec j$ then we have the relation parameter (r_{ji},n_{ji}) computed as in Eq(3).

Insertion of a new user

Next let us consider how a new user is inserted into an existing hierarchy. The insertion process can also be easily fulfilled. All that the agent has to do is update several of the relation parameters that have a connection with the new user. In brief, only those branches that are connected to or from the inserted node in the hierarchy require modification. All other branches remain unchanged.

Let us explain it more clearly using an example. Assume that a new user, l, is to be inserted into a position inferior to user j and superior to user i. The relationship of the access privileges is $i \prec l \prec j$. In this case, the agent has only to discard the old relation parameter (r_{ji},n_{ji}) and add two new, (r_{jl},n_{jl}) and (r_{li},n_{li}).

Cancellation of an existing user

The cancellation of an old user is as simple as the insertion of a new user. Reverse operations to the existing hierarchy must be performed. Imagine again the case described in the previous example. However, now the user l, in a position below j and above i, is to be cancelled. In reverse, the agent computes a new relation parameter, namely (r_{ji},n_{ji}), to replace the two old relation parameters (r_{jl},n_{jl}) and (r_{li},n_{li}).

Change of a secret key

For security considerations, it is allowable for a user to change his secret key as required. For this purpose, if a user's key is changed, then the relation parameters for those branches pointing inward or outward toward the user node must also be changed. Consider a user, namely i, is required to change his/her key from k_i into $k_i^{\star}$ for some reason. In this case, the values of the relation parameters associated with this user must be updated using the following method.

- For a user j with $i \prec j$, the new value for $(r_{ji}^{\star},n_{ji}^{\star})$ is computed as follows:

$$\begin{cases} r_{ji}^{*} = r_{ji} \oplus h(X \| ID_i \| k_i) \oplus k_i \oplus h(X \| ID_i \| k_i^{*}) \oplus k_i^{*} \\ n_{ji}^{*} = n_{ji} \oplus h(X \| ID_i \| k_i) \oplus h(X \| ID_i \| k_i^{*}) \end{cases} \quad \text{Eq(4)}$$

- For a user l with $l \prec i$, the new value for $(r_{il}^{\star},n_{il}^{\star})$ is computed as below:

$$\begin{cases} r_{il}^{*} = r_{il} \oplus h(X \| ID_i \| k_i) \oplus k_i \oplus h(X \| ID_i \| k_i^{*}) \oplus k_i^{*} \\ n_{il}^{*} = n_{il} \oplus h(X \| ID_l \| h(X \| ID_i \| k_i)) \oplus h(X \| ID_l \| h(X \| ID_i \| k_i^{*})) \end{cases} \quad \text{Eq(5)}$$

Note that the values of the relation parameters associated with the other branches remain the same.

Proposition 3.

For a user j with $i \prec pj$, the new value of $(r_{ji}^{\star},n_{ji}^{\star})$ is computed using Eq (4).

For a user l with $l \prec i$, the new value of $(r_{il}^{\star},n_{il}^{\star})$ is computed using Eq (5).

4.Cost and Security Analysis

In this section, the cost and the security of the proposed scheme are analyzed. First, the time measurement and the storage analysis are examined. As in Eq(1), 10 concatenations are required, 4 XORs and 5 hash functions to compute a relation parameter. Both of the XOR operations and concatenation can be done very efficiently compared to a multiplication with the same length of operands. The ratio of comparison between the times required for a XOR and a multiplication is about $O(n^2)$ to $O(n)$, where n is the length of the operands. Further, a hash function can also be performed efficiently. Schneier listed the encryption speeds of some hash functions in his book [pp. 456, 26]. For instance, running on a 33 MHz 486SX computer, SHA with a 160-bit output has an encryption speed of 75 kilobytes/second and MD5 with 128-bit output has a speed of 174 kilobytes/second.

To derive a key using Eq(2), 6 concatenations, 4 XORs and 3 hash functions are required. These operations can be done efficiently in a VLSI chip with low computation power. Note that for most existing schemes a large

number of modular multiplications are required. For instance, Akl's scheme [7] requires O($n \log n$) modular multiplications, n is the number of users. Similarly, the operations in equations (3), (4) and (5) can also be performed efficiently. The proposed scheme is based on the low-cost operations. There are no arithmetic operations, multiplication or exponentiation, required as in the existing schemes [7, 8, 9-12, 19, 21-22, 27].

Now, let us analyze the space required for our method. Let the personal key and the hash function output be length k each. Each user needs a space of O(k) bits for keeping the personal key. The agent needs O($|b|k$) bits for maintaining the relation parameters, where b is the number of branches in the simplified graph. In the worst case, it becomes O(n^2k), n is the number of users. However, Akl's scheme requires O($n^3 \log n$) of space.

The security level of the proposed scheme will be analyzed next. The possible attack methods for breaking the scheme for an adversary inside or outside the system are investigated.

Attack 1:

A lower-privileged user might intend to derive the secret key of his father node. In other words, some user i will try to reveal the secret key of user j with $i \prec j$. Since accessing the relation parameter (r_{ji}, n_{ji}) requires a correct password. He/she first has to break the password.

Consider a system with the relation parameters made public without requiring a password. Is it easier to derive the secret key of his/her father user? Under this case, from Eq (1), the correct value of X must still be acquired and then the one-way hash function must be broken. Based on the characteristics [26] of one-way hash functions, by knowing y, it is difficult to compute x such that $h(x)=y$ and difficult to find another message x' such that $h(x)=h(x')$. Moreover, this is 2^m times the guessing using the brute-force attack if h has m-bit output. For instance, the chosen-plaintext attack needs 2^{128} when m =128.

Attack 2:

We consider the case of a user in the system, namely user i, with m parent nodes, indicated as $j, j+1, \ldots$ and $j+m$. Now suppose that the lower-privileged user intends to derive the key of one of his parents. Is it easy to break the system? Basically, the assailant must solve the following system of equations

$$\begin{cases} r_{(j+v)i} = h(X \| ID_{j+v} \| k_{j+v}) \oplus h(X \| ID_i \| k_i) \oplus k_{j+v} \oplus k_i \\ n_{(j+v)i} = h(X \| ID_i \| h(X \| ID_{j+v} \| k_{j+v})) \oplus h(X \| ID_i \| k_i) \end{cases}, v = 0,\ldots,m. \quad \text{Eq(6)}$$

When the relation parameters (r_{ji}, n_{ji}), $(r_{(j+l)i}, n_{(j+l)i})$, …, and $(r_{(j+m)i}, n_{(j+m)i})$ are known, we still can not find an efficient method for the solution. Further, since the system has m+1 equations with m+3 unknowns, it cannot be solved uniquely. Interested readers are encouraged to try to break it.

Attack 3:

Let us consider the situation in Attack 2 again. Now instead of an assailant from inside the system, he/she might be an outsider. In this case, the intruder has no personal secret key for himself/herself. We shall take into account that it is possible for him/her to derive the secret key of some users. To put it in brief, he/she is solving Eq(6) for some k_i's without any available information. It will be more difficult than Attack 2.

Attack 4:

We assume that there is a higher-privileged user j with two lower-privileged child-users, namely i and l, and their secret keys are indicated as k_j, k_i and k_l, respectively. Now we are interested in the case of attack by collaboration. In other words, two users are cooperatively trying to derive the secret key of their common father. The simultaneous equations that must be solved for this case are as follows

$$\begin{cases} k_i = h(X \| ID_j \| k_j) \oplus h(X \| ID_i \| h(X \| ID_j \| k_j)) \oplus n_{ji} \oplus k_j \oplus r_{ji} \\ k_l = h(X \| ID_j \| k_j) \oplus h(X \| ID_l \| h(X \| ID_j \| k_j)) \oplus n_{jl} \oplus k_j \oplus r_{jl} \end{cases} \quad \text{Eq(7)}$$

From the above equations, since X and k_j are unknowns, the difficulty of breaking still rests on the one-way hash function.

Attack 5:

Finally, consider the example stated in Attack 4 again. However, we take into account the case when a user is intending to derive the key of his/her sibling branching from the same father. From Eq (7), we know that k_i (or k_l) are not helpful in solving another equation. That is, the secret key of a user can not be broken easily by a sibling.

5. Conclusions and Discussions

A new solution to the problem of access control for users organized in a hierarchy is proposed in this paper. For security functions, a tamper-proof chip may be embedded in a hardware device or exist in a computing environment. Most of the existing schemes in the literature require a large number of arithmetic computations, multiplications or exponentiations with a modulus. They are not suitable for a chip with less computing power. However, only simple and low cost operations are needed in the presented scheme. This new solution has the properties of low cost implementation, high computation efficiency and little required memory on a tamper-proof chip. The secret key of a user can be changed without affecting the keys of other users. It is easy for a new user to be inserted into or an old class cancelled from the existing system. It is straightforward and efficient to derive a lower-privileged key by owning a higher-privileged key and not the reverse. It is simple for an ancestor to deduce the key of his/her grandchild. Finally, we would like to discuss the limitations of the proposed approach. The security of the proposed scheme relies on the three assumptions: (1) tamper-proof devices, (2) the security of the one-way hash function, and (3) a single trusted agent. The centralized single agent may probably become a limitation in this scheme. All of the personal secret keys are generated and known by the agent. Further, the common random number X and the personal identity are stored in the chip. If the assumption of tamper-proof chip is removed, a cryptanalyst may have higher probability to break this scheme.

There is another implicit assumption used in security analysis, especially for Attack 1. We assume that an attacker wants to find out a specific key value k_j so to own the access privilege of user j. Consider the following scenario for a large company having hierarchical organization with many divisions and subdivisions. An employee with very low access privilege and key k_i wants to gain higher access privilege. Is it necessary for him to get the key of his direct superior? No, he might try to get a key with much higher access right. If the attacker succeeds this key, it can be used to derive the other keys in the hierarchy. However, this sort of attack doesn't rely on the scheme we proposed, but is rather related to the inherent property of the hierarchical model.

Acknowledgement: The author would like to thank the anonymous referees for comments and suggestion on this paper.

References

[1] D. E. Denning, S. G. Akl, M. Morgenstern and P.G. Nermann, Views for multilevel database security, *Proc. IEEE Symp. on Security and Privacy*, Oakland. CA. April 7-9, 1986, pp.156-172.

[2] D. E. Denning, Cryptographic checksums for multilevel database security, *Proc. IEEE Symp. on Security and Privacy, Oakland*, CA, April 29-May 2, 1984, pp.52-61.

[3] G. I. Davida, D. L. Wells and J. B. Kam, A database encryption system with subkeys, *ACM Trans. Database systems*, 6(2), June 1981, pp.312-328.

[4] J. McHugh and A. P. Moore, A security policy and formal top level specification for a multilevel secure local area network, *Proc. IEEE Symp. on Security and Privacy*, Oakland, CA, April 7-9, 1986, pp.34-39.

[5] D. McCullough, Specifications for multilevel security and a hook-up property, *Proc. IEEE Symp. on Security and Privacy*, Oakland, CA, April 27-29, 1987, pp.161-166.

[6] L. J. Fraim, Scomp: a solution to multilevel security problem, *IEEE Computer*, July 1983, pp.26-34.

[7] S. G. Akl and P. D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, *ACM Trans. on Computer System*, 1(3), August 1983, pp. 239-247.

[8] S. T. Mackinnon, P. D. Taylor, H. Meijer and S. G. Akl, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy, *IEEE Trans. Computers*, C-34(9), 1985, pp.797-802.

[9] L. Harn and H.Y. Lin, A cryptographic keys generation scheme for multilevel data security, *Computer Security*, 9(1990), pp.539-546.

[10] R. S. Sandhu, Cryptographic implementation of a tree hierarchy for access control, *Information Processing Letters*, 27(1988), pp.95-98.

[11] C. C. Chang, R. J. Hwang and T. C. Wu, Cryptographic key assignment scheme for access control in a hierarchy, *Information Systems*, 17(3), 1992, pp.243-247.

[12] C. C. Chang and D. J. Buehrer, Access control in a hierarchy using a one-way trapdoor function *Computers and Mathematics with Applications* 26(5), 1993, pp.71-76.

[13] H. T. Liaw and M. Y. Chiou, Dynamic cryptographic key assignment schemes in a hierarchy, *Proceedings of the fifth National Conference on Information Security*, Taipei, Taiwan, May 1995, pp.88-99.

[14] W. P. Lu and M. K. Sundareshan, A model for multilevel security in computer networks, *Proceedings 1988 INFCOM*, New Orleans, LA, March 1988, pp.1095-1104.

[15] M. Maekawa, A. E. Oldehoeft and R. R. Oldehoeft, *Operating Systems - Advanced Concepts*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1987.

[16] K. Siil, Adaptive applications to multilevel secure UNIX systems, *Proceedings 7th Int. Conf. and Exhibition on Information security*, Brighton, UK., May 1991.

[17] C. C. Chang, and H. T. Liaw, Assigning digital key pairs to determine relationships in a user hierarchy structure, *Transactions of Information Processing Society of Japan*, Vol. 35, No. 10, Oct. 1994, pp.2189-2196.

[18] C. C. Chang, J. K. Jan, and D. J. Buehrer, A scheme to determine the relationship between two users in a hierarchy, *Computers and Security*, Vol.13, 1994, pp.255-261.

[19] C. H. Lin, C. C. Chang, and R. C. T. Lee, A dynamic access control mechanism in information protection systems, *Journal of Information Science and Engineering*, Vol.6, 1990, pp.25-33.

[20] C. H. Lin, C. C. Chang, and R.C.T. Lee, Hierarchy representations based on arithmetic coding for dynamic information protection systems, *Information Sciences*, Vol.64, No.1-2, Oct.1992, pp.35-48.

[21] G. Horng, A key management approach for access control in user hierarchies, *Proceedings of International Computer Symposium*, Hsinchu, Taiwan R.O.C., 1994, pp.439-444.

[22] H. M. Tsai and C. C. Chang, A cryptographic implementation for dynamic access control in a user hierarchy, *Computer and Security*, Vol.14, 1995, pp.159-166.

[23] D. E. Denning, *Cryptography and data security*, Addison-Wesley, 1982.

[24] T. Leighton and S. Micali, Secret-key agreement without public-key cryptography (extended abstract), *Advances in Cryptology – CRYPTO'93*, Lecture Notes in computer Science 773 (Springer, Berlin, 1994), pp. 456-479.

[25] Y. Zheng, On key agreement protocols based on tamper-proof hardware, *Information Processing Letters*, 53, 1995, pp.49-54.

[26] B. Schneier, *Applied cryptography*, 2nd Edition, John Wiley & Sons, Inc, 1996, pp.429-455.

[27] C. H. Lin, Dynamic key management schemes for access control in a hierarchy, *Computer Communications*, 20, 1997, pp.1381-1385.