Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Design of a Lattice-based Access Control Scheme

Chia-Chu Chiang[1], Coskun Bayrak[1], Remzi Seker[1], Umit Topaloglu[2], Murat Demirer[1,3], Nasrola Samadi[1], Suleyman Tek[1], Bian Jiang[1], GuangXu Zhou[1], and Xiaoran Wang[1]
[1]Department of Computer Science
University of Arkansas at Little Rock
2801 South University Avenue
Little Rock, Arkansas, USA
[2]UAMS
[3]Computer Science Department
Faculty of Science and Letters
Kultur Universitesi, Istanbul, Turkey
{cxchiang|cxbayrak}@ualr.edu

***Abstract*—We survey the literature for access control schemes in a user hierarchy. Some schemes have already been shown to be insecure or incorrect. Many schemes assume very restrictive subordinating relationships existing in a hierarchy where users are grouped into partially ordered relationships without taking resources into consideration. We believe that a practical access control scheme should support access control in a lattice where users and resources are both together grouped into partially ordered relationships. In this paper, we present a scheme to achieve this goal. We also study existing schemes for their efficiency and performance. Based on the results of the study, we design an efficient scheme to support dynamic key management.**

***Keywords*—Access Control, Cryptographic Keys, Cryptography, Dynamic Key Management, Hierarchical Access Control, Lattice, and Rekeying**

I. INTRODUCTION

An access control scheme ensures that only legitimate users are able to access resources assigned to them. A simple access control scheme is to create secret keys for all the resources and only users are able to access resources using legitimate keys. A major problem of this kind of scheme is that users will be overwhelmed with keys as the number of resources increases. Akl and Taylor [1] proposed a cryptographic solution to this problem for access control in a user hierarchy represented by a partially ordered set. In the hierarchy, users are grouped into classes. The partial order on the hierarchy implies that a resource accessible to a class C_j is also accessible to a class C_i if $C_j \leq C_i$. The Akl and Taylor's scheme allows a user in a higher privileged class to derive the keys of its child nodes. Nowadays, there are many proposed schemes for access control in a user hierarchy. Some schemes [2-4] are proved to be insecure or incorrect in the articles [5-7]. Others [8-11] are claimed to be insecure in the article [12].

Many schemes [13-20] for access control assume very strict subordinating relationships in a user hierarchy. Users in a class have the same access to the same set of resources. However, access control for users having different access rights to multiple resources widely exists in practice. Users and resources usually have many-to-many relationships. For example, a company might provide services for different membership groups to access different resources. Generally, in such kind of applications with this type of access control, users are grouped into a partially ordered hierarchy as well as resources. Unfortunately, it is not clear to see whether resources are merged into the user hierarchies presented in a large number of published access control schemes. In this paper, we propose a Lattice-based access control key management scheme where users and resources are merged into a partially ordered lattice. The scheme employs a Central Authority (CA) to maintain partially ordered relations of users and resources. The CA is also responsible for assigning keys to classes. The scheme employs a simple and efficient key generation and derivation algorithm. In addition, the scheme further improves the efficiency of key management with the introduction of S-nodes to the lattice.

The paper is organized as follows. A literature survey is presented in Section 2. The main purpose of this survey is to understand how existing schemes differently handle access control in a user hierarchy. Most importantly, how efficiently these schemes perform dynamic key management in terms of performance, storage, and key updates. Section 3 lists a set of design requirements of our proposed scheme. According to the design requirements, our scheme with the key generation and derivation algorithm is presented in Sections 4 and 5. Section 6 analyzes the secure tolerance of our scheme. Our scheme is demonstrated with an example in Section 7. Finally, the paper is summarized in Section 8.

II. LITERATURE SURVEY OF EXISTING ACCESS CONTROL SCHEMES

After conducting a literature survey on access control, we classified the schemes into two categories: ones with public cryptography and ones without public cryptography. Since we are interested in designing a scheme to be applied in a public key infrastructure, the schemes without public cryptography [6, 13, 30] will not be considered in this research. For the schemes with public cryptography, the schemes are evaluated in terms of the computation time of key generation and derivation and the storage space required in key generation. The computation time is measured by the average number of operations performed in key generation and derivation. The

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

storage space is defined as the average number of secret keys and public parameters needed to perform computations. In addition, the scheme is also evaluated in terms of rekeying complexity defined as the average number of keys required to be updated as a user joins or leaves a class.

One of the key efficiency measures for Lattice-based access control schemes is the complexity of operations in key generation and derivation algorithms. Many schemes [12, 20, 22-23] using simple exclusive-or operations and hash functions in their key generation and derivation algorithms give reasonable performance O(n) where n is the number of nodes in the hierarchy. Several schemes [1, 8, 10, 13, 24-27] are less efficient due to the use of modular exponentiation, discrete logarithms, polynomial interpolations, and additional usage of encryption/decryption.

Schemes giving reasonable performance in key generation and derivation may require a large amount of storage space of public parameters and secret keys for computation. Such kind of schemes includes Akl and Taylor, Chen et. al., Chen and Huang, Chick and Tavares, MacKinnon et. al., and Zhong [1, 13, 20, 28-29].

Rekeying as a user joining or leaving a class determines a scheme's efficiency in dynamic key management. Very few schemes [12, 15] have rekeying handled locally where rekeying does not propagate to successors and predecessors of the affected class. Majorities of schemes [20, 30] require rekeying of all the successors of the affected class. Few schemes [1, 28] even require all the classes in the system to be rekeyed.

III. Design Requirements of Lattice Based Scheme

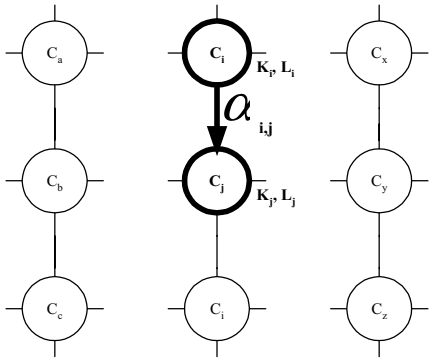
In general, a designer faces a variety of tradeoffs between algorithm complexity, efficiency, security, and space requirements. The following is a set of design requirements we consider,

- The scheme must support access control in a lattice represented by partially ordered relationships of users as well as resources.
- The scheme must be secure. Users in a higher-level class should be allowed to derive the keys of subordinating classes but not the other way around.
- The scheme must be efficient. The minimum requirements of our scheme are that all the keys should not be derived as a user joins or leaves a class. The computation of key generation and derivation should not be in an exponential time.
- The scheme should not require a large amount of storage space for storing parameters and secrets.
- The scheme must ensure to combat the ex-member problem. As a user joins or leaves a class, the keys of the impacted classes must be updated.

To meet the above requirements, we first propose a lattice to form partially ordered relationships of users and resources. The lattice will support many-to-many relationships between users and resources. A sample implementation of the lattice can be found in [31]. Our scheme uses a hash function and the exclusive-or operation to implement the key generation and derivation algorithm. In [32], Sklavos and Koufopavlou had the experiments on the performance of hash functions. The result of their work will help select a hash function with the better performance. The rekeying of all the successors of the affected class is enforced in the scheme to combat the ex-member problem. We also introduce a new concept of S-class nodes in the lattice to improve the performance of the key management protocol. The additions of S-nodes to the lattice also reduce the total amount of public parameters required for the key computation.

IV. Key Generation and Derivation

Suppose C_j is the immediate successor of C_i where $C_j \le C_i$ in a lattice. CA randomly selects two large positive integers as a secret key K_i and a public parameter L_i for the class C_i. CA also selects two large positive integers as K_j and L_j for the class C_j. A public parameter $\alpha_{i,j} = h(K_i \oplus L_j) \oplus K_j$ is computed and assigned to the edge of C_i and C_j.

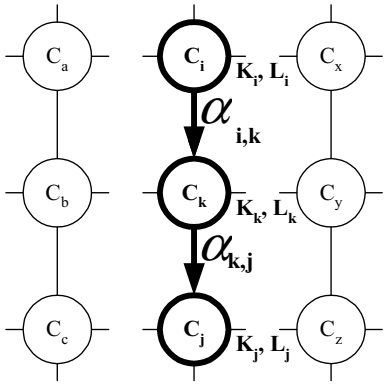


To derive the key of C_j, CA uses the public parameter L_j of C_j and computes $h(K_i \oplus L_j) \oplus \alpha_{i,j}$ for K_j.

V. Dynamic Key Management Protocol

A. Adding a Class

Assume a class C_k is to be added to the lattice between C_i and C_j where $C_j \le C_i$. CA randomly selects two large positive integers as K_k and L_k for C_k. For any class C_i with $C_k \le C_i$, CA computes $\alpha_{i,k} = h(K_i \oplus L_k) \oplus K_k$. For any class C_j with $C_j \le C_k$, CA computes $\alpha_{k,j} = h(K_k \oplus L_j) \oplus K_j$.

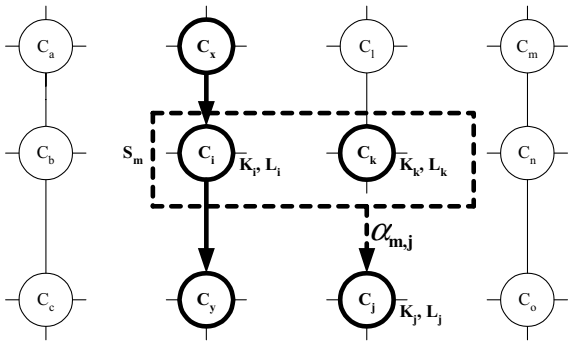


B. Deleting a Class

Assume a class C_k is to be removed from the lattice. CA removes C_k with K_k and L_k. For any class C_i with $C_k \le C_i$, CA removes $\alpha_{i,k}$ and creates $\alpha_{i,j}$. For any class C_j with $C_j \le C_k$, CA removes $\alpha_{k,j}$.

C. Adding an S-node

Suppose a lattice has C_x, C_i, and C_j where $C_j \le C_i \le C_x$. A new class C_k is to be added to the lattice with $C_j \le C_k$. CA first removes $\alpha_{i,j}$ and then groups C_i and C_k into an S-node class, S_m. CA then randomly selects two large positive integers as K_m and L_m for S_m. CA computes $\alpha_{m,j}$ between S_m and C_j. If there is another new class C_z to be added to the lattice with $C_j \le C_z$, CA just simply adds C_z to S_m.



D. Deleting an S-node

Deleting a class from an S-node is just simply removing the class from the S-node. If the resulting S-node only has one class left, our scheme will still keep the class in the S-node because there is a high probability that a new class might join the S-node later.

VI. Security Analysis

To make sure a user in a lower privileged class not to derive the keys of its predecessors, let us look at the key derivation algorithm. Assume two classes C_i and C_j in a lattice with $C_j \le C_i$. To derive the key of C_i, users in C_j require to derive K_i from $h(K_i \oplus L_j) \oplus \alpha_{i,j}$. Although L_j and $\alpha_{i,j}$ are public parameters, the users in C_j still have no way to generate K_i since this would require inversion of the h function.

In addition, users of a class in an S-node have no way to derive the keys of the other sibling classes because each class has its own private key. The key of an S-node is only used for the key derivations of its successors of the S-node. Thus, the users in the S-node can access their common resources in the lower privileged classes.

VII. An Example

Consider the access control table in Table I and the corresponding lattice in Figure 1. The user access control table is a conceptual model that specifies the rights that each user accesses for each file. Each row in this table indicates a list of files that can be accessed by this particular user. Each column indicates a list of users who have the rights to access this particular file.

TABLE I. A USER ACCESS CONTROL TABLE

	F_1	F_2	F_3
USER 1	√		
USER 2	√	√	
USER 3			√
USER 4	√	√	√

The user access control table is implemented into a lattice hierarchy shown in Figure 1. Each node in this lattice specifies the user with the corresponding file that this user has the rights to access. The lattice in Figure 1 also specifies the precedence of a user to the subordinates. For example, USER 2 in C_2 has the rights to access F_1 in C_4 and USER 4 has the rights to access F_1, F_2, and F_3 in C_2, C_3, and C_4. The CA will create K_i, L_i, $\alpha_{i,j}$, and S_i of all the nodes based on the neighborhood relationship among the nodes in this lattice. We also assume all the files need to be encrypted before saving on a computer and decrypted for uses by the users.

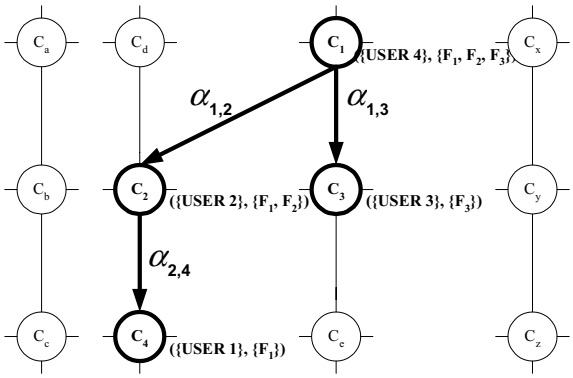


Figure 1. A lattice from table I.

Next, we add ({USER 5}, {F_4}) and ({USER 6}, {F_1, F_4}) to the access table as shown in Table I. The resulting access control table is created in Table II where new users USER 5 and USER 6 and new files F_4 are joined for the access control.

TABLE II. A USER ACCESS CONTROL TABLE WITH ADDITIONS OF ({USER 5}, {F_4}) AND ({USER 6}, {F_1, F_4})

	F_1	F_2	F_3	F_4
USER 1	√			
USER 2	√	√		
USER 3			√	
USER 4	√	√	√	√
USER 5				√
USER 6	√			√

A lattice corresponding to Table II is depicted in Figure 2 where C_2 and C_5 are grouped into an S-node, S_7. An edge between S_7 and C_4 is created with the public parameter, $\alpha_{7,4}$ indicating that User 2 and User 6 are both sharing a common file F_1 at the same precedence level in the lattice. CA improves the performance in fetching F_1 by following the common link $\alpha_{7,4}$ for User 2 and User 6. Since USER 2 is using the common

link to C_4, the link between C_2 and C_4 is no longer needed for access control. Although USER 6 in C_5 is in S_7 with USER 2 in C_2 and USER 6 is not sharing F_4 with USER 2, therefore CA needs to create a link between C_5 and C_6. The link allows CA to access F_4 for USER 4 and USER 6 but not USER2.

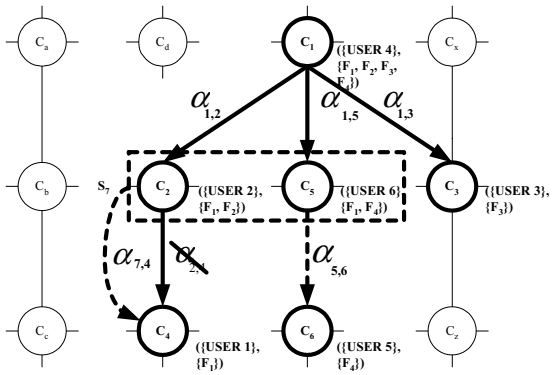


Figure 2. A lattice with additions of ({USER 5}, {F_4}) and ({USER 6}, {F_1, F_4}).

Table III creates K_i, L_i, $\alpha_{i,j}$, and S_i of all the nodes in the lattice as shown in Figure 2. The link with $\alpha_{2,4}$ is removed from the table.

TABLE III. COMPUTATIONS OF PARAMETERS AND KEYS IN FIGURE 2

	L_i	$\alpha_{i,j}$	K_i	S_i
C_1	L_1	$\alpha_{1,2} = h(K_1 \oplus L_2) \oplus K_2$ $\alpha_{1,3} = h(K_1 \oplus L_3) \oplus K_3$ $\alpha_{1,5} = h(K_1 \oplus L_5) \oplus K_5$	K_1	–
C_2	L_2	$\alpha_{2,4} = h(K_2 \oplus L_4) \oplus K_4$	K_2	S_7
C_3	L_3	–	K_3	–
C_4	L_4	–	K_4	–
C_5	L_5	$\alpha_{5,6} = h(K_5 \oplus L_6) \oplus K_6$	K_5	S_7
C_6	L_6	–	K_6	–
S_7	L_7	$\alpha_{7,4} = h(K_7 \oplus L_4) \oplus K_4$	K_7	–

Next, let us add ({USER 7}, {F_1, F_5}) and ({USER 4}, {F_5}) to the access control table as shown in Table IV. A new row and a new column are added to the control access table to specify the access rights of USER 7 on F_1 and F_5.

TABLE IV. A USER ACCESS CONTROL TABLE WITH ADDITIONS OF ({USER 7}, {F_1, F_5}) AND ({USER 4}, {F_5})

	F_1	**F_2**	**F_3**	**F_4**	**F_5**
USER 1	√				
USER 2	√	√			
USER 3			√		
USER 4	√	√	√	√	√
USER 5				√	
USER 6	√			√	
USER 7	√				√

The corresponding lattice of table IV is depicted below. Now, since USER 7 is sharing F_1 with USER 2 and USER 6, there is no action to be taken in creating its own link to C_4. Instead, CA just joins C_8 to S_7.

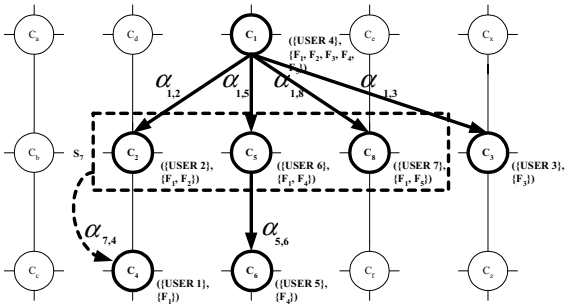


Figure 3. A lattice with additions of ({USER 7}, {F_1, F_5}) and ({USER 4}, {F_5}).

In this scenario, C_8 just joins S_7 with its own key K_8 and public parameter L_8. Table V lists K_i, L_i, $\alpha_{i,j}$, and S_i of all the nodes in the lattice depicted in Figure 3.

TABLE V. COMPUTATIONS OF PARAMETERS AND KEYS IN FIGURE 3

	L_i	$\alpha_{i,j}$	K_i	S_i
C_1	L_1	$\alpha_{1,2} = h(K_1 \oplus L_2) \oplus K_2$ $\alpha_{1,3} = h(K_1 \oplus L_3) \oplus K_3$ $\alpha_{1,5} = h(K_1 \oplus L_5) \oplus K_5$ $\alpha_{1,8} = h(K_1 \oplus L_8) \oplus K_8$	K_1	–
C_2	L_2	–	K_2	S_7
C_3	L_3	–	K_3	–
C_4	L_4	–	K_4	–
C_5	L_5	$\alpha_{5,6} = h(K_5 \oplus L_6) \oplus K_6$	K_5	S_7
C_6	L_6	–	K_6	–
S_7	L_7	$\alpha_{7,4} = h(K_7 \oplus L_4) \oplus K_4$	K_7	–
C_8	L_8	–	K_8	S_7

At this moment, assume USER 4 in C_1 is going to access F_1 and F_4. CA is required to do the following steps to derive K_4 and K_5 for the encryption/decryption of F_1 and F_4.

1. Compute $K_5 = h(K_1 \oplus L_5) \oplus \alpha_{1,5}$;
2. Compute $K_4 = h(K_7 \oplus L_4) \oplus \alpha_{7,4}$; and
3. CA sends K_4 and K_5 to the users in C_1 to encrypt/decrypt F_1 and F_4.

Next, let us remove USER 2 from the user access table. It means that F_2 will no longer become inaccessible to USER 4 in C_1. In this scenario, CA just removes C_2 from S_7 without impacting the other nodes in the lattice. The resulting lattice is depicted in Figure 4.

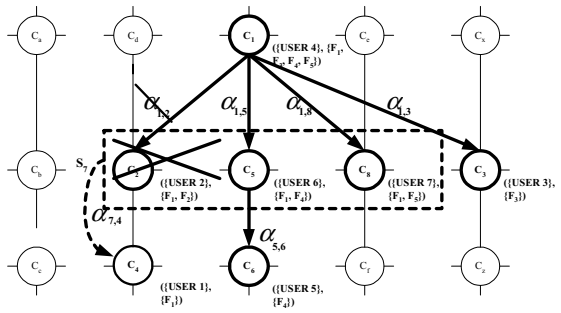


Figure 4. A lattice with removal of {USER 2}.

Table VI lists K_i, L_i, $\alpha_{i,j}$, and S_i of all the nodes in the lattice shown in Figure 4. The shaded row in Table VI indicates C_2 is no longer considered by CA for access control.

TABLE VI. COMPUTATIONS OF PARAMETERS AND KEYS IN FIGURE 4

	L_i	$\alpha_{i,j}$	K_i	S_i
C_1	L_1	$\alpha_{1,3} = h(K_1 \oplus L_3) \oplus K_3$ $\alpha_{1,5} = h(K_1 \oplus L_5) \oplus K_5$ $\alpha_{1,8} = h(K_1 \oplus L_8) \oplus K_8$	K_1	–
C_2	L_2	–	K_2	S_7
C_3	L_3	–	K_3	–
C_4	L_4	–	K_4	–
C_5	L_5	$\alpha_{5,6} = h(K_5 \oplus L_6) \oplus K_6$	K_5	S_7
C_6	L_6	–	K_6	–
S_7	L_7	$\alpha_{7,4} = h(K_7 \oplus L_4) \oplus K_4$	K_7	–
C_8	L_8	–	K_8	S_7

Next, let us assume USER 6 in C_5 is removed from the system. Since USER 6 in C_5 no longer needs to access F_4, the link between C_5 and C_6 should be removed from the lattice, thus CA removes C_5 and the link C_5 to C_6 with $\alpha_{5,6}$. After the removal of C_5 from S_7, C_6 becomes a successor of the predecessor of C_5 which is C_1. Thus, CA creates a new link from C_1 to C_6 with $\alpha_{1,6}$. In addition, USER 4 in C_1 no longer has the precedence over USER 6 in C_5, CA removes the link between C_1 and C_5 with $\alpha_{1,5}$. The resulting lattice is depicted in Figure 5.

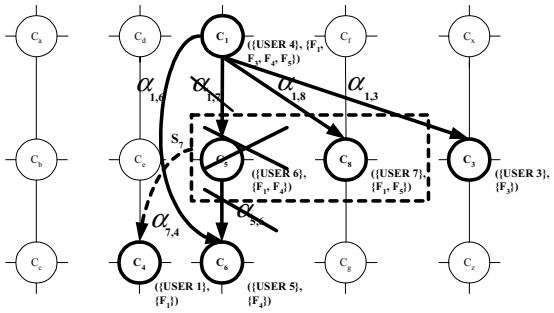


Figure 5. A lattice with removal of ({USER 6}, {F_1, F_4}).

With the all scenarios demonstrated above, the final lattice is shown as follows.

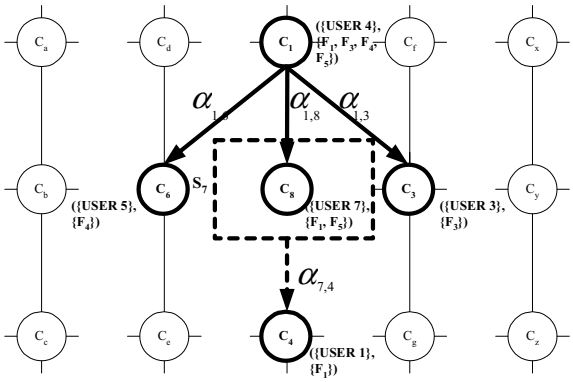


Figure 6. The final lattice.

You might notice that S_7 only contains a node C_8 that should be removed from the lattice. The motivation of having an S-node in a lattice is that the nodes in the S-node are all sharing the successors. Therefore, it might be a great possibility that F1 in C4 would be shared again by new users together with USER 7 later. That is the reason we decided to keep an S-node with only one node left in a lattice.

VIII. SUMMARY

We presented an efficient access control scheme in this paper with the following features,

- The scheme must support access control in a lattice represented by partially ordered relationships of users as well as resources.
- The key generation and derivation algorithm is simple and efficient.
- The scheme introduces a new concept of S-nodes to the lattice that further improves the performance of key generation and derivation and reduces the storage requirements.
- The scheme is resistant to collusion in that users in a class cannot derive the keys of its predecessors.
- The scheme does not require the entire system to be rekeyed as a user joins or leaves a class. Neither the scheme propagates the rekeying to predecessors or successors.

With the above features, our scheme shows more efficiency and less storage requirements than other schemes. Our scheme uses a hash function and the exclusive-or operation in the key generation and derivation algorithm. Compared to the schemes [1, 8, 10, 13, 24-27] using modular exponentiation, discrete logarithms, polynomial interpolations, and additional usage of encryption/decryption, our scheme is more efficient than these schemes. Comparable to the schemes with reasonable performance [13, 20, 28-29], our scheme uses less amount of storage space for storing public parameters and secret keys. Rekeying is definitely not needed for the entire system. Instead, rekeying only occurs locally. Definitely our schemes perform

more efficiently in rekeying than majorities of schemes [1, 12, 15, 20, 30].

ACKNOWLEDGMENT

The work was supported in part by the Department of Defense (DOD) under Award No. H98230-07-C-0403.

REFERENCES

[1] Selim G. Akl and Peter D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM Transactions on Computer Systems*, Vol. 1, No. 3, September 1983, pp. 239-248.

[2] Tzer-Shyong Chen and Yu-Fang Chung, "Hierarchical Access Control Based on Chinese Remainder Theorem and Symmetric Algorithm," *Computer & Security*, Vol. 21, No. 6, 2002, pp. 565-570.

[3] Hui-Feng Huang and Chin-Chen Chang, "A New Cryptographic Key Assignment Scheme with Time-Constraint Access Control in a Hierarchy," *Computer Standards & Interfaces*, Vol. 26, 2004, pp. 159-166.

[4] Chu-Hsing Lin, "Dynamic Key Management Schemes for Access Control in a Hierarchy," *Computer Communications*, Vol. 20, No. 15, December 1997, pp. 1381-1385.

[5] Narn-Yih Lee and Tzonelih Hwang, "Comments on 'Dynamic Key Management Schemes for Access Control in a Hierarchy'," *Computer Communications*, Vol. 22, No. 1, January 1999, pp. 87-89.

[6] Qiang Tang and Chris J. Mitchell, "Comments on a Cryptographic Key Assignment Scheme," *Computer Standards & Interfaces*, Vol. 27, No. 3, March 2005, pp. 323-326.

[7] Sheng Zhong and Tianwen Lin, "A Comment on the Chen-Chung Scheme for Hierarchical Access Control," *Computer & Security*, Vol. 22, No. 5, 2003, pp. 450-452.

[8] Victor R. L. Shen and Tzer-Shyong Chen, "A Novel Key Management Scheme Based on Discrete Logarithms and Polynomial Interpolations," *Computer & Security*, Vol. 21, No. 2, 2002, pp. 164-171.

[9] Wen-Guey 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, January/February 2002, pp. 182-188.

[10] Tzong-Chen Wu and Chin-Chen Chang, "Cryptographic Key Assignment Scheme for Hierarchical Access Control," *International Journal of Computer Systems Science and Engineering*, Vol. 1, No. 1, 2001, pp. 25-28.

[11] Jyh-haw Yeh, Randy Chow, and Richard Newman, "A Key Assignment for Enforcing Access Control Policy Exceptions," *Proceedings of the International Symposium on Internet Technology*, 1998, pp. 54-59.

[12] Mikhail J. Atallah, Keith B. Frikken, and Marina Blanton, "Dynamic and Efficient Key management for Access Hierarchies," *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005, pp. 190-202.

[13] Tzer-Shyong Chen, Yu-Fang Chung, and Chang-Sin Tian, "A Novel Key Management Scheme for Dynamic Access Control in a User Hierarchy," *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, September 2004, pp. 396-401.

[14] Hung-Yu Chien and Jinn-Ke Jan, "New Hierarchical Assignment without Public Key Cryptography," *Computers & Security*, Vol. 22, No. 6, September 2003, pp. 523-526.

[15] Fuh-Gwo Jeng and Chung-Ming Wang, "A Practical and Dynamic Key Management Scheme for a User Hierarchy," *Journal of Zhejiang University Science A*, Vol. 7, No. 3, 2006, pp. 296-301.

[16] Chu-Hsing Lin, "Hierarchical Key Assignment without Public-Key Cryptography," *Computer & Security*, Vol. 20, No. 7, October 2001, pp. 612-619.

[17] Adrian Penrig, Dawn Song, and J. D. Tygar, "ELK: a New Protocol for Efficient Large-Group Key Distribution," *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2001)*, 2001, pp. 247-262.

[18] Yan Sun, Wade Trappe, and K. J. Ray Liu, "Topology-Aware Key Management Schemes for Wireless Multicast," *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '03)*, Vol. 3, 2003, pp. 1471-1475.

[29] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, February 2000, pp. 16-30.

[20] Sheng Zhong, "A Practical Key Management Scheme for Access Control in a User Hierarchy," *Computers & Security*, Vol. 21, No. 8, November 2002, pp. 750-759.

[21] Yuliang Zheng, "On Key Agreement Protocols Based on Tamper-Proof Hardware," *Information Processing Letters*, Vol. 53, 1995, pp. 49-54.

[22] Mingxing He, Pingzhi Fan, Firoz Kaderali, and Ding Yuan, "Access Key Distribution Scheme for Level-Based Hierarchy," *Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'03)*, August 2003, pp. 942-945.

[23] Chu-Hsing Lin and Wei Lee, "Efficient Secret Sharing with Access Structures in a Hierarchy," *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, March 2005, pp. 123-126.

[24] Chin-Chen Chang, Iuon-Chang Lin, Hui-Min Tsai, and Hsiao-Hsi Wang, "A Key Assignment Scheme for Controlling Access in Partially Ordered User Hierarchies," *Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)*, March 2004, pp. 376-379.

[25] Manik Lal Das, Ashutosh Saxena, Ved P. Gulati, and Deepak B. Phatak, "Hierarchical Key Management Scheme Using Polynomial Interpolation," *ACM SIGOPS Operating Systems Review*, Vol. 39, No. 1, January 2005, pp. 40-47.

[26] Hui-Min Tsai and Chin-Chen Chang, "A Cryptographic Implementation for Dynamic Access in a User Hierarchy," *Computer & Security*, Vol. 14, No. 2, 1995, pp. 159-166.

[27] Mark Vroblefski, Andrew Chen, Benjamin Shao, and Matthew Swinarski, "Managing User Relationships in Hierarchies for Information System Security," *Decision Support Systems*, Vol. 43, No. 2, March 2007, pp. 408-419.

[28] Gerald C. Chick and Stafford E. Tavares, "Flexible Access Control with Master Keys," In *Advances in Cryptology – CRYPTO '89*, Vol. 435, Lecture Notes in Computer Science (LNCS), 1990, pp. 316-322.

[29] Stephen MacKinnon, Peter Taylor, Henk Meijer, and Selim G. Akl, "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy," *IEEE Transactions on Computers*, Vol. 34, No. 9, September 1985, pp. 797-802.

[30] Yuliang Zheng, Thomas Hardjono, and Jennifer Seberry, "New Solutions to the Problem of Access Control in a Hierarchy," University of Wollongong, Technical Report, Preprint No. 93-2, 1993, Retrieved from *http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/2255/http:zSzzSziaks-www.ira.uka.dezSztazSzSecurityzSzDiverseszSztr-93-2.pdf/zheng93new.pdf*, December 3, 2007.

[31] Qiong Zhang and Yuke Wang, "A Centralized Key Management Scheme for Hierarchical Access Control," *Proceedings of the 2004 Global Telecommunications Conference (GLOBECOM '04)*, Vol. 4, 2004, pp. 2067-2071.

[19] Nicolas Sklavos and Odysseas Koufopavlou, "Access Control in Networks Hierarchy: Implementation of Key Management Protocol," *International Journal of Network Security*, Vol. 1, No. 2, September 2005, pp. 103-109.