Proceedings of the 17th IASTED International Conference
Parallel and Distributed Computing and Systems
November 14-16, 2005, Phoenix, AZ, USA

AN INDEPENDENT SET APPROACH TO SOLVING THE COLLABORATIVE ATTACK PROBLEM

Anne V.D.M. Kayem, Selim G. Akl, and Pat Martin
School of Computing
Queen's University
Kingston, Ontario, K7L,3N6, CANADA
email: kayem@cs.queensu.ca, akl@cs.queensu.ca, and martin@cs.queensu.ca

ABSTRACT
Access control in distributed databases has tended to favor a hierarchical approach implemented via cryptographic schemes. In such schemes, a central authority generates keys for each level in the hierarchy such that users at a given level can compute, from their own key, the keys of users below them and gain access to information items they hold. Previous schemes proposed, have been found to be either vulnerable to "collaborative attack"[1] or inefficient. This paper presents a method of assigning keys at each level in the hierarchy such that the probability of their being combined to generate illegal keys is minimized. We model the problem as a graph whose vertices represent the keys generated, and whose edges indicate the probability that their end points can be combined to generate a "collaborative attack". The concept of independent sets is then used to demonstrate the feasibility of our approach.

KEY WORDS
Hierarchical access control, distributed databases, security.

1 Introduction

The ability of distributed databases to allow for the execution of multiple concurrent transactions in an open and flexible manner across a myriad of applications, was amongst the factors that gave impetus to the development of web based information systems. These systems provide a transparent environment where users can exchange information without having to grapple with complex decisions about hardware and software constraints. In recent years, however, the increased complexity of the underlying distributed database systems and availability of personal information on such systems, has made the idea of access control a necessary measure to address the problem of information disclosure. Distributed databases are composed of many distinct entities (users or processes generated by a user's activities) and these entities usually have distinct privileges of accessing different parts or depths of resources (data items) in the system. Data items are usually classified into security classes C_i, $1 \leq i \leq m$, which are ordered by the binary relation '$\preceq$' such that they form a partial order hierarchy. In the partial order hierarchy '$C_j \preceq C_i$' indicates that the security clearance of C_j is lower than that of C_i and that entities that are permitted to access data items in C_i are entitled to access items in C_j. Application examples include the unequal relations in the military, government organizations and commercial enterprises.

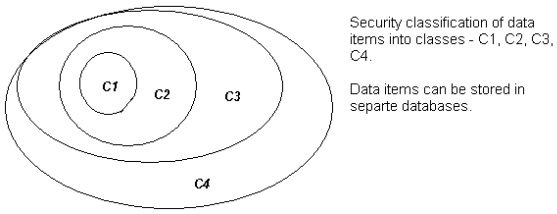

Figure 1. Disjoint set representation of data item classification

For example in Fig. 1, to handle e-commerce, a company may choose to classify data into four classes: "Company Financial Status" - C_1, "Customer Files" - C_2, "Product Catalogue" - C_3, and "Web Pages" - C_4; such that ***Directors*** can access all the data $\{C_1 \cup C_2 \cup C_3 \cup C_4\}$, ***Customer Service Representatives*** can access everything but the "Company Financial Status" data $\{C_2 \cup C_3 \cup C_4\}$, ***Web Site Maintainers*** can access the "Product Catalogue" and "Web Pages" $\{C_3 \cup C_4\}$ and ***Customers*** can only access "Web Pages" $\{C_4\}$. Assigning cryptographic keys to entities such that an entity at a higher level can derive the keys of lower classes is a good solution to the problem of controlling access to these data items [2,5,11]; first of all because the scheme does not depend on the physical security of the system on which the data resides and secondly because the storage space required for keys higher up in the hierarchy is minimised. Instead of attributing multiple keys to entities higher up in the hierarchy, the Central Authority (CA) only has to give each a single key from which all the keys for data items with lower levels of clearance can be deduced (computed).

These interesting qualities as well as their capacity to provide confidentiality, integrity and origin authenticity of a request for connection, have constituted the principal mo-

[1] A "collaborative attack" occurs when two or more users at the same level in the hierarchy collaboratively compute, from their respective keys, a key (higher up in the hierarchy) to which they are not entitled.

tivation for access control schemes in distributed database systems [5,6,7,8]. Effective access control in these systems is modeled as a hierarchical tree where entities are classified into n disjoint sets, $S = \{U_1, U_2, ..., U_n\}$. A partially ordered set $(S, \preceq)$ is defined such that '$\preceq$' represents the binary relation between two sets of entities. Thus entities in set U_i can retrieve data items with a security clearance of C_i, and $U_j \preceq U_i$ indicates that entities in U_i are entitled to access data items (C_j) held by entities in U_j, but the reverse situation should not occur.

However, the theoretical access control solutions on which these schemes are based have been shown to be vulnerable to the problem of "collaborative attack", whereby two or more users collude (using their respective keys) to compute a key to which they are not entitled [2,3,4,5,7]. Fig. 2 gives a graphical illustration of how this might occur. Consider a situation in which Bob and Alice are web site maintainers (see Fig. 1) for the same firm. Bob decides to steal access into "company financial status" data, so he convinces Alice to hand over her private key. A combination of both keys according to some function Bob defines, generates a key that grants him read/write access to the higher level which he normally should not be able to access. This is a serious problem for organizations where information sensitivity is an issue. A solution based on the concept of using maximum dispersion to solve the problem of "collaborative attacks" in hierarchical access control schemes, is presented in this paper.

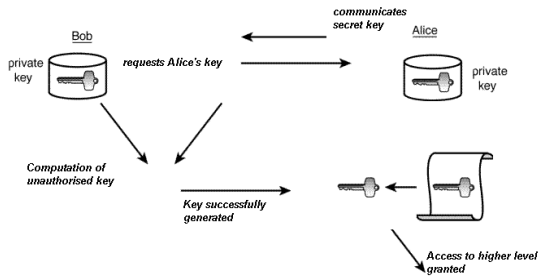

Figure 2. Scenario of a "collaborative attack"

The paper is structured as follows. In section 2 a brief review of related work on key assignment schemes is presented. It was demonstrated in [2] and [3] that the simpler of the Akl-Taylor cryptographic schemes, designed to handle access control in a hierarchy, is vulnerable to "collaborative attack". Section 3 reviews an example, drawn from Mackinnon et al. [3], to show how this might occur. We present our solution for the general case, and give an illustrative example to demonstrate its feasibility, in section 4. A number of concluding remarks are made in section 5.

2 Related Work

Access control in distributed systems is typically modeled along the lines of mandatory access control. Mandatory access control is a system of access control that assigns security labels or clearances to system resources and allows access only to users (persons, processes, devices) with distinct levels of authorization [19]. Attributing clearance levels to data items is particularly adapted to these systems since access cannot be controlled from a single central point. Typically, these controls are enforced by the operating system and implemented by multilevel secure (MLS) distributed databases [8], according to the properties outlined by Bell-Lapadula in [18] namely:

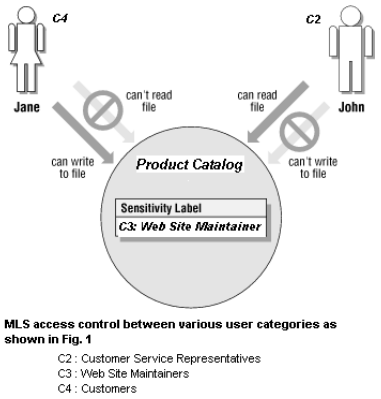

Figure 3. Illustration of MLS security properties [19]

- **Simple security property :** An entity can read from an object (data item) as long as the entity's security level is the same as, or higher than the object's security level. This is sometimes called the *no read up* property.
- ***-property :** An entity can write to an object (data item) as long as the entity's security level is the same as or lower than the object's security level.

Fig. 3 gives an indication of what these properties imply. These two properties prevent direct flow of information from objects and/or entities at a lower level, and are the basis of all MLS models. The first scheme to consider cryptography as a method of enforcing these conditions in any system where access control is modeled along the lines of MLS, was published in [2]. Cryptographic schemes in this context present several advantages that make them useful for enforcing security in distributed databases where entities communicate via untrusted communication networks. Since protection applies not only to data records stored in the database but also to messages broadcast during message passing between system entities, interception of data packets is useless unless the interceptor possesses the proper key.

The publication of the Akl-Taylor hierarchical access control cryptographic scheme [2] triggered the development of several schemes for the distributed database systems scenario [3,4,5,6,7,8,11,12]. Most of the schemes proposed are based on public-key cryptography because of

the increased security and convenience it provides: private keys never need to be transmitted or revealed to anyone. In a secret-key system, by contrast, the secret keys must be transmitted (either manually or through a communication channel), and there may be a chance that an eavesdropper can discover the secret keys during their transmission.

In [2], an entity can from its own cryptographic key derive the keys of all the entities below itself. More importantly, the scheme protects against "collaborative attack". However, as the number of entities in the system grows so does the size of the keys held by higher level entities thereby increasing the number of computations and time required to derive lower level keys.

To address this problem, Mackinnon et al. [3] presented an improved algorithm designed to determine an optimal assignment of keys by attributing the smallest primes to the longest chains in the poset that defines the hierarchy. The existence of an algorithm to achieve an optimal decomposition of the poset in polynomial time remains an open problem. A simple heuristic algorithm based on using a reduced number of distinct primes was proposed to resolve the problem. Although this improves on the efficiency of [2], a large amount of storage is still required for the public parameters.

Sandhu used one-way functions in order to resolve the problem of access control in an efficient manner [17]. With this scheme, each secret key K_i for a security class C_i is generated with its own identity and its immediate ancestor's secret key through a one-way function. The drawback of this scheme is the large computational overhead which completely defeats the purpose of efficient key generation.

In 1990, Harn and Lin [11] proposed an approach whereby instead of using a top-down strategy as in the Akl-Taylor scheme, a bottom-up key generation scheme was presented. With this solution the space required to store the public parameters for most security classes is much smaller than that required by the previous schemes. However, when there are many security classes in the system, a large amount of storage space is required to store the public parameters.

In order to address these issues, Chang et al. [13] proposed a scheme based on Newton's interpolation method and a predefined one-way function. Again the drawback of this scheme is in the time required for key derivation and generation. In addition, these schemes have been shown to be vulnerable to "collaborative attacks" [7,13].

These solutions handled the problem as though keys attributed to users were intended to be for an indeterminate period and that the only time when it was necessary to regenerate keys was when a new user joined or left the system. In practical scenarios it is likely that users may belong to a class for a short period of time and then get a promotion, which may give them the right to a higher security clearance.

Tzeng [4] proposed the first time-bound key assignment scheme to handle this situation. His solution supposes that each class C_i has many class keys $K_{i,t}$, where $K_{i,t}$ is the key of class C_i during time period t. The scheme is very efficient in terms of space requirements because a user need only keep the information item $I(i, t_1, t_2)$, which is independent of the total number of classes for the time period from t_1 to t_2, in order to be able to derive all the keys to which they are entitled. However the computation of a cryptographic key, say $K_{i,t}$, requires expensive public-key and Lucas computations [14], and therefore have limited the implementation of this scheme in the distributed environment. The scheme was also shown to be vulnerable to a "collaborative attack" in [14]. Chien [5] proposed a solution based on a tamper resistant device in order to address this problem but this solution has also been shown to be vulnerable to attack [15].

3 Towards a Solution

We shall assume, based on the evidence given in section 2, that no matter how well an access control protocol is specified, there is always the possibility that it will have some flaw that makes it susceptible to a "collaborative attack" and that when this is not the case, the time and space requirements make it inefficient. The solution we propose, aims to demonstrate that correct identification and elimination of the keys generating the problem can minimise the risk of attack as well as provide efficiency in key generation and derivation. In order to situate our argument in the context of distributed database systems, we briefly review a simple scheme described in [2] as an example of a scheme that that seemingly satisfies the access control requirements but is provably vulnerable to "collaborative attack".

3.1 "Collaborative Attack" - An example

To generate keys the CA, U_0 chooses a secret key K_0 and two distinct large primes, p and q, and makes their product $M = p.q$ public but keeps p and q secret. Each class of users U_i is attributed a public integer t_i such that the property (1), holds

$$t_j | t_i \quad if\ and\ only\ if \quad U_i \preceq U_j \qquad (1)$$

Next K_i is obtained with $K_i = K_0^{t_i} (mod\ M)$ and communicated to U_i. In the case where $U_i \preceq U_j$, according to (1) $t_j | t_i$ is an integer and hence U_j can compute K_i with

$$K_i = K_0^{t_i} = K_0^{t_j (t_i | t_j)} = K_j^{t_i | t_j} (mod\ M)$$

If this is not the case, then the computation is considered to be infeasible.The choice of the t_is is conducted in an ad-hoc fashion due to efficiency requirements. In Fig. 4 the t_i associated with each class of users U_i is indicated on top of the node which represents that class.

The weakness of this scheme, as was shown in [2], is in the ad-hoc selection of t_i which makes the scheme vulnerable to "collaborative attack". For example, in Fig. 4,

two users with keys $K_i = K_0^4$ and $K_j = K_0^9$ respectively, can compute K_0 with the following equation

$$(K_i)^{-2} K_j = K_0^{-8} K_0^9 = K_0 (mod\ M)$$

K_0 is the key from which all the keys in the system are derived, so by successfully obtaining this key, users in U_3 and U_5 obtain a key granting them access to all the resources in the system. The solution proposed in [2] to avoid an attack of this sort was shown to be inefficient in [2] and [3]. Indeed, the t_i yielded from $t_i = \prod_{U_j \preceq U_i} p_j$ where p_j is the sequence of distinct primes chosen by U_0 grow large as the number of entities n increases. [3] proposes an improvement but, as mentioned above the solution requires a large amount of storage.

4 Independent set algorithm

It was shown in [16] that any hierarchical cryptographic access control protocol in which the central authority (CA) selects a single RSA modulus n as the basis of key generation, inevitably allows for the derivation of illegal keys in the system. Hence there is always a possibility that, no matter how well a scheme is defined, if keys generated are not tested for the possibility of their being combined to generate "illegal keys", some may exist that are likely to cause "collaborative attacks". The following sub-sections outline our solution, based on the graph theoretic concept of independent sets. Basically, keys liable to generate "illegal keys" are identified and eliminated through the application of an algorithm to compute an independent set of keys from keys generated for a given level in the hierarchy.

4.1 Preliminaries and Assumptions

Let us assume that the partial order hierarchy, is always obeyed and that the CA generates keys for each level in the hierarchy in such a manner as to enforce these rules. Assume also that the keys generated, at each level, form a graph whose structure is defined on the basis of the likelihood of their being combined to generate a "collaborative attack". The keys represent the vertices in the graph while an edge between any two vertices would

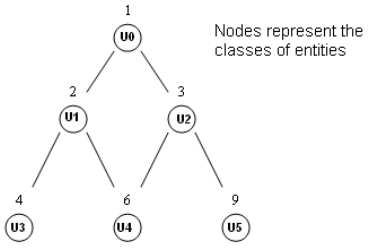

Figure 4. Assignment of t_i (indicated by integer values at top of node) to users

indicate that they can be combined to compute an illegal key.

Example: (see Fig. 4)
Combining K_0^4 and K_0^9 with the simple equation shown below generates a key K_0^1 that belongs in the hierarchy

$$\left(K_0^4\right)^{-2} \left(K_0^9\right) = K_0 \left(mod\ M\right)$$

whereas applying the same equation to the keys K_0^4, K_0^6, and K_0^9 give:

$$\left(K_0^4\right)^{-2} \left(K_0^6\right) = K_0^{-8} K_0^6 \left(mod\ M\right) = K_0^{-2} \left(mod\ M\right)$$

$$\left(K_0^6\right)^{-2} \left(K_0^9\right) = K_0^{-12} K_0^9 \left(mod\ M\right) = K_0^{-3} \left(mod\ M\right)$$

keys which do not belong in the hierarchy. According to this equation for collusion, there would be an edge between K_0^4 and K_0^9, whereas there is none between K_0^4 and K_0^6, nor between K_0^6 and K_0^9. This is a very simple example because by the same reasoning, $\left(K_0^4\right)^{-1}$ and K_0^6 give K_0^2 which is also in the hierarchy. The assumption is that edges are generated between keys by testing the same conditions on all candidate keys. This can lead to a very sparse graph or a very dense graph in the best and worst cases respectively.

Definition 1: Key connectivity is implied by the ease with which any two keys at a level, say j, can be combined to generate a key at a higher level, say i. Thus, adjacent vertices are more likely to be combined successfully to derive an illegal key than non-adjacent vertices.

The need to find large independent sets typically arises in dispersion problems where the objective is to identify a set of mutually separated points. For example, in our case, we want to assign keys at every level in the access control hierarchy such that no two keys are close enough to be combined to derive another key (particularly a key belonging to a higher level). The principal weakness of many cryptographic schemes lies in their vulnerability to attacks based on key connectivity.

Definition 2: The graph $G = (V, E)$ is the representation of key connectivity where V represents the keys and E the edges between these keys. Note that the edges of G are obtained based on all best current knowledge of possible attacks. The absence of an edge between two keys in no way guarantees that these two keys cannot be used in a "collaborative attack".

Definition 3: A set of vertices $I \subset V$ is called ***independent*** if no pair of vertices in I is connected via an edge in G; an independent set is called ***maximal*** if, by including any other vertex not in I, the independence property is violated [1,10].

The largest independent set would be the maximum possible number of keys that can be derived such that the

conditions for security are not violated. The set can therefore best accommodate the demands of the system to ensure that all the classes in the system get a unique key. Since the problem of determining a largest independent set is NP-hard we use a heuristic to obtain an approximate solution in polynomial time. Intuitively, the algorithm works as follows. The key with the lowest value and degree of connectivity is selected, added to the independent set and then it, as well as all the vertices adjacent to it, are deleted. The process is repeated until the graph is empty or until the number of keys generated is equal to the number of keys required. As the definition implies, independent sets are not unique. Hence, in the best case a largest independent set is obtained and in the worse case scenario where all the keys are adjacent to the lowest-degree and lowest-value vertex, only one key is obtained.

4.2 Algorithm

Algorithm 1 MIS Algorithm

1: Choose a random vertex v from the set C
2: Compare $dvalue$ to $dvalues$ of all adjacent vertices
3: **if** $dvalue <$ *all adjacent dvalues* **then**
4: select v and update K with new entry
5: **else**
6: **if** $dvalue >$ *all adjacent dvalues* **then**
7: Drop vertex v selected
8: Select a vertex v with the minumum $dvalue$ amongst the adjacent vertices
9: **if** $dvalue =$ *all adjacent dvalues considered* **then**
10: select vertex with smallest $rvalue$
11: **end if**
12: **else**
13: **if** $dvalue =$ *all adjacent dvalues* **then**
14: select vertex with smallest $rvalue$
15: **end if**
16: **end if**
17: **end if**
18: C is then updated so that all the vertices that were selected for inclusion in K are deleted from it

The CA selects a modulus n_i for each level in the hierarchy, where i represents the hierarchical level. On the basis of this and the number of users in the system, a set of candidate keys, C is generated. An independent set I is then computed from the keys generated and these are then distributed to the users at the specified level. Let K be a set of size $|V|$, where V is the set of keys generated for a level i. When the algorithm terminates, K will store all the v_i which satisfy the conditions for inclusion in the independent set. Initially K is empty, and during each iteration of the algorithm some v_i is inserted in to K. Let C be a set of size $|V|$, where C is considered is the set of candidate vertices for inclusion in K.

For each level in the hierarchy, the set K is initially set to empty and the set of candidate vertices, C, holds all the $v_i \in V$. Each vertex in C is characterized by the unique random key ($rvalue$) assigned to it by the CA computed on the basis of the modulus n selected for a level i and its degree ($dvalue$), that is the number of edges that link it to adjacent keys. Let $v_1, v_2 \in V$. If $dvalue(v_1) \leq dvalue(v_2)$ then v_1 is considered to be a better candidate for inclusion in I, because its connectivity to the other keys is less than v_2's.The simplest reasonable heuristic is to find the lowest-degree vertex, add it to the independent set and delete it and all the vertices adjacent to it. The MIS Algorithm is executed sequentially for every level in the hierarchy.

This procedure is then repeated for the vertices left in C and K is progressively augmented. The incremental augmentation ends when C is empty. On average this algorithm converges after $O(\log|V|)$ iterations.

4.2.1 Example

Consider for example Fig. 5(a), with vertices labeled $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ such that $\{v_1 < v_2 < ... < v_6\}$ generated for a level, say i, in the hierarchy such that $1 \leq i \leq n$. Each $v_j, 1 \leq j \leq 6$ represents the $rvalue$ of a vertex. When the algorithm was executed initially on the graph Fig. 5(a), vertex v_2 was selected at random and found to have the lowest $rvalue$ compared to the adjacent vertices $\{v_3, v_4, v_5, v_6\}$. It however has the highest degree - $dvalue$. So the choice of vertices is between v_5 *and* v_6. v_5 has the smaller value (considering the indices), so it gets selected which leaves the scenario in Fig. 5(b). Here v_1 gets selected but happens to have the largest $dvalue$ amongst the three, so automatically the choice goes to v_3 and v_6. v_3 gets selected first and its adjacent node v_1 is deleted; which leaves v_6 unconnected so it is added to the set. In the final analysis the keys selected would be $\{v_3, v_5, v_6\}$.

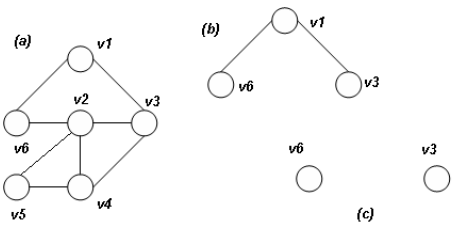

Figure 5. Example of selection of independent set

If we consider for example that v_2 is representative of the key $K_i = K_0^4$ and, v_6 of $K_j = K_0^9$, then it can be observed that the "collaborative attack" problem is eliminated because the algorithm will either select one or the other but never both. So the possibility of combined key attacks is removed.

It is obvious from this illustration that several different combinations would generate correct and valid independent sets. We do not consider this to be a disadvantage

but rather an advantage because different sets could be generated off line and attributed when there is a demand for a new set of keys. This would be the case when a user joins or leaves the system and the original structure is maintained. Thus, the method can in the best case contribute to an improvement in the efficiency of the key generation scheme [2].

5 Conclusion

In this paper, we have presented a solution to the problem of "collaborative attacks" in cryptographic access control protocols for distributed database systems. The approach we have adopted is based on the principle of computing an independent set from the vertices in a graph. This graph is constructed from the set of keys generated by the central authority (CA) for each level in the access control hierarchy. The keys are considered to represent the vertices in the graph and they were connected via edges indicative of the likelihood of their being combined to generate a "collaborative attack". Adjacent keys signify a higher attack likelihood than non-adjacent keys.

The independent set approach towards resolving the problem clearly illustrates that the problem of key selection to prevent "collaborative attacks" in hierarchical access control systems can be reduced to the classical graph problem of determining a largest independent set. As the problem is NP-hard, a heuristic is used to achieve an efficient (but perhaps suboptimal) solution in polynomial time. Nevertheless, as illustrated, the solution is feasible.

The drawback of this scheme is that it adds computational overhead on the system. The algorithm requires $O(\log|V|)$ (V is the number of vertices or keys) time at least, for the key selection process. The problems of efficient key management and redistribution when a user is added to the system remain. Areas for future work include: *Determining an optimal complexity bound for the key selection process* and *Devising an efficient key management and redistribution scheme*.

Practical applications for the scheme include web based distributed database systems where access control is an issue and where key selection and regeneration needs to be performed "on the fly" because of the dynamic nature of this environment. In such systems users may join or leave the system dynamically. Access control in such systems needs to be handled in a smooth and seamless fashion so that the operations are hardly perceptible to or affect the execution of other processes.

REFERENCES

[1] M. Adams, A Parallel Maximal Independent Set Algorithm, *Proc. 5th Copper Mountain Conf. on Iterative Methods*, 1998.

[2] S.G. Akl and P.D. Taylor, Cryptographic Solution to a Problem of Access Control in a hierarchy, *ACM Transactions on Computer Systems*, 1(3), 1983, 239-248.

[3] S.J. Mackinnon, P.D. Taylor, H. Meijer and S.G. Akl, An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy, *IEEE Transactions on Computers*, c-34(9), 1985, 797-802.

[4] W-G. Tzeng, A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy, *IEEE Transactions on Knowledge and Data Engineering*, 14(1), 2002, 182-188.

[5] H-Y Chien, Efficient Time-Bound Hierarchical Key Assignment Scheme, *IEEE Transactions on Knowledge and Data Engineering*, 16(10), 2004, 1301-1304.

[6] I-C. Lin, M-S. Hwang and C.-C. Chang, A New Key Assignment Scheme for Enforcing Complicated Access Control Policies in Hierarchy, *Future Generation Computer Systems*, 19, 2003, 457-462.

[7] M-S. Hwang and W-P. Yang, Controlling Access in Large Partially Ordered Hierarchies using Cryptographic Keys, *The Journal of Systems and Software*, 67, 2003, 99-107.

[8] A. De Santis, A.L. Ferrara and B. Masucci, Cryptographic Key Assignment Schemes for any Access Control Policy, *Information Processing Letters*, 92, 2004, 199-205.

[10] M. Luby, A Simple Parallel Algorithm for the maximal independent set problem, *SIAM Journal of Computing*, 15(4), 1986, 1036-1052

[11] L Harn and H.Y. Lin, A Cryptographic Keys Generation Scheme for Multilevel Data Security, *Computer Security*, 9, 1990, 539-546.

[12] C.H. Lin, Hierarchical Key Assignment without Public-Key Cryptography, *Computer Security*, 20(7), 2001, 612-619

[13] M.-S. Hwang, C.-C. Chang, and W.-P. Yang, Modified Chang-Hwang-Wu Access Control Scheme, *IEE Electronics Letters*, 29(24), 1993, 2095-2096

[14] X. Yi, Security of Tzeng's Time-Bound Key Assignment Scheme for Access Control in a Hierarchy, *IEEE Transactions on Knowledge and Data Engineering*, 15(4), 2003, 1054-1055

[15] A. De Santis, A. Ferrari and B. Masucci, On the Insecurity of a Time-bound Hierarchical Key Assignment Scheme, *http://www.cacr.math.uwaterloo.ca/*, 2005 Technical reports, 2005, CACR 2005-07

[16] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone *Handbook of Applied Cryptography* (CRC Press. 2001)

[17] R.S. Sandhu, Cryptographic Implementation of a Tree Hierarchy for Access Control, *Information Processing Letters*, 27, 1988, 95-98.

[18] D. Bell and L. Lapadula, Secure Computer Systems: Mathematical Foundations and Model, *MITRE Technical Report*,MITRE Corporation, 1974.

[19] http://www.oreilly.com/catalog/csb/chapter/fig.03.03.gif