A Cryptographic Solution to Implement Access Control in a Hierarchy and More

Indrakshi Ray
CS Department
Colorado State University
Fort Collins, CO 80523-1873
iray@cs.colostate.edu

Indrajit Ray
CS Science Department
Colorado State University
Fort Collins, CO 80523-1873
indrajit@cs.colostate.edu

Natu Narasimhamurthi
ECE Department
Univ. of Michigan Dearborn
Dearborn, MI 48128
nnarasim@umich.edu

ABSTRACT

The need for access control in a hierarchy arises in several different contexts. One such context is managing the information of an organization where the users are divided into different security classes depending on who has access to what. Several cryptographic solutions have been proposed to address this problem – the solutions are based on generating cryptographic keys for each security class such that the key for a lower level security class depends on the key for the security class that is higher up in the hierarchy. Most solutions use complex cryptographic techniques: integrating these into existing systems may not be trivial. Others have impractical requirement: if a user at a security level wants to access data at lower levels, then all intermediate nodes must be traversed. Moreover, if there is an access control policy that does not conform to the hierarchical structure, such policy cannot be handled by existing solutions. We propose a new solution that overcomes the above mentioned shortcomings. Our solution not only addresses the problem of access control in a hierarchy but also can be used for general cases. It is a scheme similar to the RSA cryptosystem and can be easily incorporated in existing systems.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: [Public key cryptosystems]; G.2.m [**Discrete Mathematics**]: [Miscellaneous]

General Terms

Security

Keywords

Access control, Hierarchy, Cryptography

1. INTRODUCTION

In any organization the personnel are frequently organized in the form of a hierarchy and there is the requirement that information is distributed over the hierarchy on a "need-to-know" basis. The information maintained by the organization is often stored in a database and different users have different kinds of access to these information. The information is typically organized in the form of relations and different granularity of access are achieved using views [4]. However, views have inherent problems associated with them [8]. An alternate way of ensuring confidentiality is by using cryptography. The data is encrypted in such a manner that only authorized users can decrypt it. In this paper, we propose one such technique.

Several solutions based on cryptographic techniques [1, 2, 12, 6, 9, 14, 18, 23] that address the problem of access control in a hierarchy have been proposed. Most of them employ complex cryptographic techniques [1, 2, 12, 6, 14]; integrating these with existing systems may not be very trivial. Others have undesirable requirements [1, 2, 12]. (See the related work for details.) Majority of these work, as expected, do not address the problem of how their mechanism can be adapted if the access control policy does not adhere to this hierarchical structure.

The implementation we propose is completely general and can be used to implement different kinds of access control policies of an organization [15] whenever the access control policy can be represented as a Directed Acyclic Graph (DAG). In general, the access control policies will follow the hierarchical structure of an organization. For example, a manager should be able to access any record that his employee can access. Many a times, however, the access control policy may not follow this organizational structure. We illustrate this with an example. Consider the hierarchy presented in figure 1. The project supervisor is senior to both programmers and test engineers. Thus project supervisors should have access to the documents created by test engineers and programmers. However, sometimes the test engineers may not want the project supervisor to view their incomplete work. Again project supervisors need not view all the documents created by the project members. Thus, different kinds of access control are needed for different documents created in an organization. In this paper we propose a mechanism for implementing such general access control policies. Providing a generalized mechanism for implementing different kinds of access control policies is one of the goals of this work.

The rest of the paper is organized as follows. Section 2 mentions some related work on hierarchical access control. Section 3 gives a general overview of our approach. Section 4 discusses the theory on which our approach is based. Section 5 presents the algorithms for key generation. Section 6 comments on the security of our scheme. Section 7 gives the algorithms for key management when the organizational hierarchy is altered. Section 8 presents an alternate solution to this problem and outlines the advantage our proposed approach over this alternate solution. Section 9 concludes the paper.

SACMAT'02, June 3-4, 2002, Monterey, California, USA.

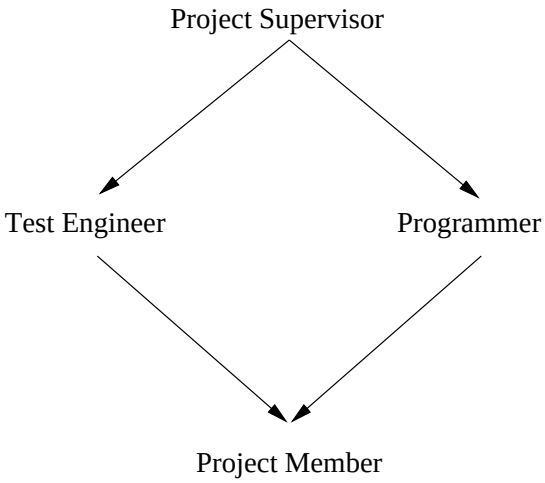

Figure 1: Enterprise Wide Personnel Hierarchy

2. RELATED WORK

A number of works [1, 2, 6, 9, 12, 14, 18, 23] relating to access control in a hierarchy have been proposed. In almost all these works, there is a relationship between the key assigned to a node and those assigned to its children. The difference between the related works lie mostly in the different cryptographic techniques employed for key generation. Some of these techniques [1, 6, 9, 12, 14] are extremely complex. Below we outline a few important works in this area.

One of the early solutions to the hierarchical access control problem was proposed by Akl and Taylor [1, 2]. Their solution was based on the RSA cryptosystem [17]. In this work the authors choose the exponents in such a way that the key of a child node can be easily derived from the key of its parent. Mackinnon et al. [12] gave an optimal algorithm for selecting suitable exponents. One potential drawback of these schemes is that if a user at a node wishes to access information stored at a descendant node, he must traverse all the intermediate nodes between his node and the descendant node. This may not be very desirable for cases where the length of the hierarchy is large. Another drawback is that addition of a new node N_i to the leaf of the hierarchy results in key generation of all ancestors of N_i.

Sandhu [18] proposed a key generation scheme for a tree hierarchy. The solution was based on using different one-way functions to generate the key for each child node in the hierarchy. The one-way function was selected based on the name or identity of the child. When a new child is added, the keys for the ancestors do not have to be recomputed. This work, however, does not deal for the case of a general poset. Zheng et al. [23] proposed solutions for the general poset. The authors present two solutions – one for indirect access to nodes (in which to access data at a lower node, the user has to traverse the intermediate nodes) and the other for direct access to nodes.

3. OVERVIEW OF OUR APPROACH

Our approach is simple. We formulate a new encryption protocol that is used to encrypt the data stored in a database. The encryption ensures data integrity as well as data confidentiality. The data is encrypted with appropriate keys at the same time it is generated. Different portions of the database are encrypted with different keys. A user, who has to retrieve information from the database will attempt to decrypt the entire database with a decrypting key that is assigned to him. However, the user is able to decrypt successfully only that portion(s) of the database for which the user is authorized. The remaining portion of the database is not decrypted successfully.

Since the access control technology is based on encrypting with the appropriate key, we first present the theory on which the key generation is based.

4. THEORY BEHIND THE CRYPTOGRAPHIC TECHNIQUES

For the following exposition we use the term *message* to denote any piece of information that needs to be stored in the database.

Definition 1. The set of *messages* M is the set of non negative integers m that are less than an upper bound N, i.e.

$$M = \{m | 0 \leq m < N\} \tag{1}$$

Definition 2. Given an integer a and a positive integer N, the following relationship holds,

$$a = qN + r \text{ where } 0 \leq r < N \text{ and } q = \lfloor a/N \rfloor \tag{2}$$

where $\lfloor x \rfloor$ denotes the largest integer less than x. The value q is referred to as the *quotient* and r is referred to as the *remainder*. The remainder r, denoted $a \bmod N$, is referred to as the *least positive residue* of $a \bmod N$.

Definition 3. For positive integers a, b and N, we say a is *equivalent* to b, modulo N, denoted by $a \equiv b \bmod n$, if $a \bmod n = b \bmod n$.

Definition 4. Two integers a, b are said to be *relatively prime* if their only common divisor is 1, that is, $\gcd(a,b) = 1$.

Definition 5. The integers $n_1, n_2, \ldots, n_k$ are said to be *pairwise relatively prime*, if $\gcd(n_i, n_j) = 1$ for $i \neq j$.

Definition 6. The Euler's totient function $\phi(N)$ is defined as the number of integers that are less than N and relatively prime to N. Below we give some properties of totient functions that are of importance.

1. $\phi(N) = N - 1$ if N is prime.
2. $\phi(N) = \phi(N_1)\phi(N_2)\ldots\phi(N_k)$ if $N = N_1 N_2 \ldots N_k$ and N_1, N_2, $\ldots$, N_k are pairwise relatively prime.

THEOREM 1. *Euler's theorem states that for every a and N that are relatively prime,*

$$a^{\phi(N)} \equiv 1 \bmod N$$

PROOF. We omit the proof of Euler's theorem and refer the interested reader to any book on number theory [13] or cryptography [20]. □

COROLLARY 1. *If $0 < m < N$ and $N = N_1 N_2 \ldots N_k$ and N_1, N_2, $\ldots$, N_k are primes, then $m^{x\phi(N)+1} \equiv m \bmod N$.*

Definition 7. A *key* K is defined to be the ordered pair $< e, N >$, where N is a product of distinct primes, $N \geq M$ and e is relatively prime to $\phi(N)$; e is the *exponent* and N is the *base* of the key K.

Definition 8. The *encryption* of a message m with the key $K = < e,N >$, denoted as $[m,K]$, is defined as

$$[m, < e,N >] = m^e \mod N \tag{3}$$

Definition 9. The *inverse* of a key $K =< e,N >$, denoted by K^{-1}, is an ordered pair $< d,N >$, satisfying $ed \equiv 1 \mod \phi(N)$.

THEOREM 2. *For any message m.*

$$\left[[m,K],K^{-1}\right] = \left[\left[m,K^{-1}\right],K\right] = m \tag{4}$$

where $K =< e,N >$ and $K^{-1} =< d,N >$.

PROOF. We first show that

$$\left[[m,K],K^{-1}\right] = m$$

$$\begin{array}{lll} LHS & & \\ = & \left[[m,K],K^{-1}\right] & \\ = & \left[m^e \mod N, K^{-1}\right] & \text{(Def. 8)} \\ = & (m^e \mod N)^d \mod N & \text{(Defs. 8, 9)} \\ = & m^{ed} \mod N & \text{(mod arith)} \\ = & m^{(x\phi(N)+1)} \mod N & \text{(Defs. 2, 9)} \\ = & m \mod N & \text{(Coro. 1)} \\ = & m & \text{(since } m < N\text{)} \\ = & RHS & \end{array}$$

By symmetry $\left[\left[m,K^{-1}\right],K\right] = m$. □

COROLLARY 2. *An encryption, $[m,K]$, is* one-to-one *if it satisfies the relation*

$$\left[[m,K],K^{-1}\right] = \left[\left[m,K^{-1}\right],K\right] = m$$

Definition 10. Two keys $K_1 =< e_1,N_1 >$ and $K_2 =< e_2,N_2 >$ are said to be *compatible* if $e_1 = e_2$ and N_1 and N_2 are relatively prime.

Definition 11. If two keys $K_1 =< e,N_1 >$ and $K_2 =< e,N_2 >$ are *compatible*, then the *product* key, $K_1 \times K_2$, is defined as $< e,N_1N_2 >$.

LEMMA 1. *For positive integers a, N_1 and N_2,*

$$(a \mod N_1N_2) \equiv a \mod N_1$$

PROOF. Let $a = N_1N_2x + N_1y + z$, where x, y and z are integers.

$$\begin{array}{ll} LHS & \\ = & (a \mod N_1N_2) \mod N_1 \\ = & \left(N_1N_2x+N_1y+z-\left\lfloor\frac{N_1N_2x+N_1y+z}{N_1N_2}\right\rfloor \times N_1N_2\right) \mod N_1 \\ = & (N_1y+z) \mod N_1 \\ = & z \end{array}$$

$$\begin{array}{ll} RHS & \\ = & (a \mod N_1) \\ = & (N_1N_2x+N_1y+z) \mod N_1 \\ = & z \end{array}$$

Hence the proof. □

THEOREM 3. *For any two messages m and $\hat{m}$, such that $m, \hat{m} < N_1, N_2$,*

$$[m,K_1 \times K_2] \equiv [\hat{m},K_1] \; mod N_1 \;\; \textit{if and only if} \;\; m = \hat{m} \tag{5}$$

$$[m,K_1 \times K_2] \equiv [\hat{m},K_2] \; mod N_2 \;\; \textit{if and only if} \;\; m = \hat{m} \tag{6}$$

where K_1 is the key $< e,N_1 >$, K_2 is the key $< e,N_2 >$ and $K_1 \times K_2$ is the product key $< e,N_1N_2 >$.

PROOF. The proof for (6) is the same as that for (5). We just consider the proof for (5).

[If part]

Given $m = \hat{m}$ we have to prove that

$$[m,K_1 \times K_2] \equiv [\hat{m},K_1] \mod N_1$$

or

$$[m,K_1 \times K_2] \mod N_1 = [\hat{m},K_1] \mod N_1$$

$$\begin{array}{lll} L.H.S. & = & [m,K_1 \times K_2] \mod N_1 \\ & = & (m^e \mod N_1N_2) \mod N_1 \quad \text{(Def. 8 and Def. 11)} \\ & = & m^e \mod N_1 \quad \text{(subs. } m^e \text{ for a lemma 1)} \end{array}$$

$$\begin{array}{lll} R.H.S. & = & [\hat{m}^e \mod N_1] \mod N_1 \\ & = & \hat{m}^e \mod N_1 \quad \text{(idempotency of mod op.)} \\ & = & m^e \mod N_1 \quad \text{(since m} = \hat{m}\text{, given)} \end{array}$$

[Only If part]

Given $[m,K_1 \times K_2] \equiv [\hat{m},K_1] \mod N_1$, we have to prove $m = \hat{m}$

$$\begin{array}{lll} [m,K_1 \times K_2] & \equiv & [\hat{m},K_1] \mod N_1 \\ [m,K_1 \times K_2] \mod N_1 & = & [\hat{m},K_1] \mod N_1 \text{ (Def. 3)} \\ (m^e \mod N_1N_2) & = & (\hat{m}^e \mod N_1) \text{ (Defs. 8, 11)} \\ m^e \mod N_1 & = & (\hat{m}^e \mod N_1) \mod N_1 \text{ (lem. 1)} \\ [m, < e,N_1 >] & = & [\hat{m}, < e,N_1 >] \text{ (Def. 8)} \\ m & = & \hat{m} \text{ (encryption is one-to-one)} \end{array}$$

□

5. KEY GENERATION

Most often in an organization the access requirements match its hierarchical structure. That is, a person higher up in the hierarchy will usually have access to the documents that can be accessed by persons lower in the hierarchy. So we present the key generation technique for this case first. Later on, we describe how to accomodate the cases where the access requirement does not follow this hierarchical pattern.

An organization is usually structured in the form of a hierarchy. Thus, we define a hierarchy of levels as follows.

Definition 12. We consider an organization structure that can be represented as a partially ordered set (poset), $(L,<)$. L is the set of *levels* of the organization and $<$ is the *dominance relation* between the levels.

1. If L_i and L_j are two levels such that $L_i < L_j$, then we say that L_j *strictly dominates* L_i and L_i *is strictly dominated by* L_j.
2. If L_i and L_j are two levels such that $L_i < L_j$ and $L_j < L_i$, then we say that the two levels L_i and L_j are *equal* and denote this by $L_i = L_j$.
3. If levels L_i and L_j are such that either $L_i < L_j$ or $L_i = L_j$, then we say that L_i *is dominated by* L_j or L_j *dominates* L_i and denote this by $L_i \leq L_j$.

4. Two levels L_i and L_j are said to be *incomparable* if nether $L_i \leq L_j$ nor $L_j \leq L_i$.

5. We say L_y is a *parent* of L_x if $L_x < L_y$ and there is no element L_z such that $L_x < L_z < L_y$. If $L_x < L_y$, then L_x is said to be a *descendant* of L_y and L_y is said to be an *ancestor* of L_x.

Since the organizational structure is a poset, it can be depicted in the form of a Hasse diagram [19].

Definition 13. Each level L_i in the organizational hierarchy is associated with a unique pair of keys, (K_i, K_i^{-1}) which we term the *default keys* for level L_i. K_i is the *default encryption key* of level L_i and K_i^{-1} is the *default decryption key* of level L_i. If the access requirements match the hierarchical structure of the organization, then a person at level L_i uses the default encryption key K_i for encrypting documents and the key K_i^{-1} for decrypting the encrypted documents.

A document encrypted with key K_i possesses the following property:

1. it can be decrypted by key K_i^{-1} where K_i^{-1} is the default decryption key of level L_i,

2. it can be decrypted by key K_j^{-1} where K_j^{-1} is the default decryption key of level L_j such that $L_j > L_i$.

3. it cannot be decrypted by key K_k^{-1} where K_k^{-1} is the default decryption key of level L_k and $L_k < L_i$ or L_k is incomparable with L_i.

5.1 Determining the Default Encryption Key and Default Decryption Key

ALGORITHM 1. *Default keys generation*
Input: *(i)* **L** *- the set of levels and (ii)* $<$ *- the dominance relation*
Output: *(i)* **K** *- the set containing the default encryption key for each level and (ii)* K^{-1} *- the set containing the default decryption key for each level.*

Procedure *GenerateDefaultKeys(*(**L**,$<$)*)*
begin
  $\mathbf{K} := \{\}$
  $\mathbf{F} := \{\}$
  for *each* $L_i \in \mathbf{L}$ **do**
    $factor_i := 1$
  while $\mathbf{F} \neq \mathbf{L}$ **do**
  begin
    choose a maximal element L_i *in* $\mathbf{L} - \mathbf{F}$*;*
    for *each parent* L_j *of* L_i **do**
      $factor_i = factor_i * N_j$
    choose random $N_i^{'}$ *where* $gcd(N_i{}', factor_i) = 1$
    $N_i = N_i^{'} * factor_i$
    $\mathbf{F} = \mathbf{F} \cup \{L_i\}$
  end
  // Remove factors that have been included multiple times.
  $\mathbf{D} := \{\}$
  while $\mathbf{D} \neq \mathbf{L}$ **do**
  begin
    choose a maximal element L_i *in* $\mathbf{L} - \mathbf{D}$*;*
    for *each ancestor* L_k *of* L_i **do**
    begin
      c := no. of distinct paths from L_i *to* L_k
      $N_i = N_i / (N_k^{'})^{c-1}$
    end
    $\mathbf{D} = \mathbf{D} \cup L_i$
  end
  choose exponent e obeying the requirements in Def. 7
  for *each* $L_i \in \mathbf{L}$ **do**
  begin
    $K_i := < e, N_i >$
    $K_i^{-1} := < d_i, N_i^{'} >$ *where* $e * d_i \equiv 1 \bmod \phi(N_i^{'})$
    $\mathbf{K^{-1}} = \mathbf{K^{-1}} \cup K_i^{-1}$
    $\mathbf{K} := \mathbf{K} \cup K_i$
  end
  return K
end

The default encryption and decryption keys can adequately provide access control for cases where the access requirements match the hierarchical structure of the organization.

5.2 Example 1: Student Transcript Information

To illustrate our approach we consider an academic organization – the College of Engineering at some hypothetical university. The College of Engineering is headed by a *Dean*. Below the Dean are the *Department Chairpersons* who are responsible for managing the individual departments. Each *Faculty* (except for cases of joint appointment) is overseen by the respective Department Chair. The *Students* are at the very bottom of this hierarchy. Each student is advised by one or more faculty members. The organizational hierarchy is shown in figure 2.

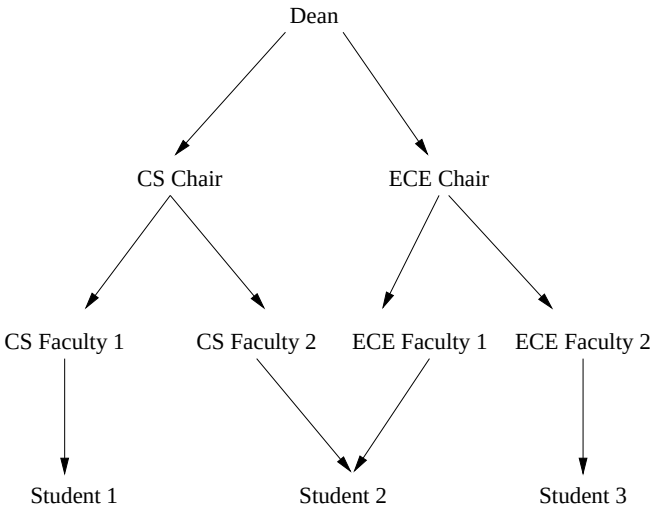

Figure 2: Access Requirements matching Enterprise Wide Personnel Hierarchy

We need to maintain the information about the transcripts of individual students. Since this is sensitive information, we need to protect it. The following are the access requirements.

1. Each student is able to view his or her transcript.

2. A faculty advising a student is able to see his transcript.

3. The chair of the department in which a student is majoring is able to view the student's transcript.

4. The dean is able to view the student's transcript.

For this example, we consider the case for three students, namely, $Student_1$, $Student_2$, $Student_3$. Each of these students have a faculty advisor who monitors and advises the student. $Student_1$'s advisor is *CS Faculty*$_1$. $Student_2$ is co-advised by *CS Faculty*$_2$ and *ECE Faculty*$_1$. $Student_3$'s advisor is *ECE Faculty*$_2$. This is illustrated in figure 2. These are the access requirements for this example:

1. $Student_1$, *CS Faculty*$_1$, *CS Chair*, *Dean* can view $Student_1$'s transcript.
2. $Student_2$, *CS Faculty*$_2$, *ECE Faculty*$_1$, *CS Chair*, *ECE Chair*, *Dean* can view $Student_2$'s transcript.
3. $Student_3$, *ECE Faculty*$_2$, *ECE Chair*, *Dean* can view $Student_3$'s transcript.

Note that, in this case the access requirements match the organizational hierarchical structure. That is, if a person X has access to some information, then a person Y at a higher level in the hierarchy will also have access to that information.

5.2.1 Access Control for Example 1

The access control requirement for Example 1 follows the hierarchical structure of the organization. Thus, using the default encryption keys the access can be appropriately restricted. Consider the hierarchy shown in figure 2. The keys for the various people are as follows.

1. The default encryption, decryption keys for $Student_i$ are denoted respectively by $K_{S_i}, K_{S_i}^{-1}$ where $1 \leq i \leq 3$.
2. The default encryption, decryption keys for *CS Faculty*$_i$ are denoted respectively by $K_{CF_i}, K_{CF_i}^{-1}$ where $i = 1$ or 2.
3. The default encryption, decryption keys for *ECE Faculty*$_i$ are denoted respectively by $K_{EF_i}, K_{EF_i}^{-1}$ where $i = 1$ or 2.
4. The default encryption, decryption keys for *CS Chair* are denoted respectively by $K_{C_{Chair}}, K_{C_{Chair}}^{-1}$.
5. The default encryption, decryption keys for the *ECE Chair* are denoted respectively by $K_{E_{Chair}}, K_{E_{Chair}}^{-1}$
6. The default encryption, decryption keys for the *Dean* are denoted respectively by K_{Dean}, K_{Dean}^{-1}.

The key K_{Dean} is chosen as per definition 7. Once the *Dean*'s key has been fixed, the other keys as generated by Algorithm 1 are as follows:

1. Default Encryption Key of *Dean* :
   $K_{Dean} =< e, N_{Dean}^{'} >$
   Default Decryption Key of *Dean* :
   $K_{Dean}^{-1} =< d_{Dean}, N_{Dean}^{'} >$,
   where $e * d_{Dean} \equiv 1 \bmod \phi(N_{Dean}^{'})$
2. Default Encryption Key of *CS Chair* :
   $K_{C_{Chair}} = K_{Dean} \times K_{C_{Chair}^{'}}$,
   where $K_{C_{Chair}^{'}}$ is *compatible* with K_{Dean}
   that is, $K_{C_{Chair}} =< e, N_{Dean}^{'} * N_{C_{Chair}}^{'} >$
   Default Decryption Key of *CS Chair* :
   $K_{C_{Chair}}^{-1} =< d_{C_{Chair}}, N_{C_{Chair}}^{'} >$,
   where $e * d_{C_{Chair}} \equiv 1 \bmod \phi(N_{C_{Chair}}^{'})$
3. Default Encryption Key of *ECE Chair* :
   $K_{E_{Chair}} = K_{Dean} \times K_{E_{Chair}}^{'}$,
   where $K_{E_{Chair}}^{'}$ is *compatible* with K_{Dean}
   that is, $K_{E_{Chair}} =< e, N_{Dean}^{'} * N_{E_{Chair}}^{'} >$
   Default Decryption Key of *ECE Chair* :
   $K_{E_{Chair}}^{-1} =< d_{E_{Chair}}, N_{E_{Chair}}^{'} >$,
   where $e * d_{E_{Chair}} \equiv 1 \bmod \phi(N_{E_{Chair}}^{'})$
4. Default Encryption Key of *CS Faculty*$_1$:
   $K_{CF_1} = K_{C_{Chair}} \times K_{CF_1}^{'}$,
   where $K_{CF_1}^{'}$ is *compatible* with $K_{C_{Chair}}$
   that is, $K_{CF_1} =< e, N_{Dean}^{'} * N_{C_{Chair}}^{'} * N_{CF_1}^{'} >$
   Default Decryption Key of *CS Faculty*$_1$:
   $K_{CF_1}^{-1} =< d_{CF_1}, N_{CF_1}^{'} >$,
   where $e * d_{CF_1} \equiv 1 \bmod \phi(N_{CF_1}^{'})$
5. Default Encryption Key of *CS Faculty*$_2$:
   $K_{CF_2} = K_{C_{Chair}} \times K_{CF_2}^{'}$,
   where $K_{CF_2}^{'}$ is *compatible* with $K_{C_{Chair}}$
   that is, $K_{CF_2} =< e, N_{Dean}^{'} * N_{C_{Chair}}^{'} * N_{CF_2}^{'} >$
   Default Decryption Key of *CS Faculty*$_2$:
   $K_{CF_2}^{-1} =< d_{CF_2}, N_{CF_2}^{'} >$,
   where $e * d_{CF_2} \equiv 1 \bmod \phi(N_{CF_2}^{'})$
6. Default Encryption Key of *ECE Faculty*$_1$:
   $K_{EF_1} = K_{E_{Chair}} \times K_{EF_1}^{'}$,
   where $K_{EF_1}^{'}$ is *compatible* with $K_{E_{Chair}}$
   that is, $K_{EF_1} =< e, N_{Dean}^{'} * N_{E_{Chair}}^{'} * N_{EF_1}^{'} >$
   Default Decryption Key of *ECE Faculty*$_1$:
   $K_{EF_1}^{-1} =< d_{EF_1}, N_{EF_1}^{'} >$,
   where $e * d_{EF_1} \equiv 1 \bmod \phi(N_{EF_1}^{'})$
7. Default Encryption Key of *ECE Faculty*$_2$:
   $K_{CF_2} = K_{E_{Chair}} \times K_{EF_2}^{'}$,
   where $K_{EF_2}^{'}$ is *compatible* with $K_{E_{Chair}}$
   that is, $K_{CF_2} =< e, N_{Dean}^{'} * N_{E_{Chair}}^{'} * K_{EF_2}^{'} >$
   Default Decryption Key of *ECE Faculty*$_2$:
   $K_{EF_2}^{-1} =< d_{EF_2}, N_{EF_2}^{'} >$,
   where $e * d_{EF_2} \equiv 1 \bmod \phi(N_{EF_2}^{'})$
8. Default Encryption Key of $Student_1$:
   $K_{S_1} = K_{CF_1} \times K_{S_1}^{'}$, where $K_{S_1}^{'}$ is compatible with K_{CF_1}
   that is, $K_{S_1} =< e, N_{Dean}^{'} * N_{C_{Chair}}^{'} * N_{CF_1}^{'} * N_{S_1}^{'} >$
   Default Decryption Key of $Student_1$:
   $K_{S_1}^{-1} =< d_{S_1}, N_{S_1}^{'} >$,
   where $e * d_{S_1} \equiv 1 \bmod \phi(N_{S_1}^{'})$
9. Default Encryption Key of $Student_2$:
   $K_{S_2} = K_{CF_2} \times K_{EF_1} \times K_{S_2}^{'}$,
   where $K_{S_1}^{'}$ is compatible with K_{CF_1} and K_{EF_1}
   that is, $K_{S_2} =< e, N_{Dean}^{'} * N_{C_{Chair}}^{'} * N_{E_{Chair}}^{'} * N_{CF_2}^{'} * N_{EF_1}^{'} * N_{S_2}^{'} >$
   Default Decryption Key of $Student_2$:
   $K_{S_2}^{-1} =< d_{S_2}, N_{S_2}^{'} >$,
   where $e * d_{S_2} \equiv 1 \bmod \phi(N_{S_2}^{'})$
10. Default Encryption Key of $Student_3$:
    $K_{S_3} = K_{EF_2} \times K_{S_3}^{'}$,
    where $K_{S_3}^{'}$ is compatible with K_{EF_2}
    that is, $K_{S_3} =< e, N_{Dean}^{'} * N_{E_{Chair}}^{'} * N_{EF_2}^{'} * N_{S_3}^{'} >$

Default Decryption Key of $Student_3$:
$K_{S_3}^{-1} =< d_{S_3}, N'_{S_3} >$,
where $e * d_{S_3} \equiv 1 \bmod \phi(N'_{S_3})$

Consider $Student_1$'s transcript. If this transcript is encrypted with key K_{S_1}, then any of the keys K_{Dean}^{-1}, $K_{C_{Chair}}^{-1}$, $K_{CF_1}^{-1}$, $K_{S_1}^{-1}$ can be used to decrypt it. Thus all the personnel higher up in the hierarchy can decrypt this transcript using their own default decryption key.

5.3 Determining a Customized Encryption Key

As illustrated by Examples 2 and 3, sometimes the access requirement does not match the organizational structure and then a different encryption key, which we term *customized encryption key*, must be used to protect this information. The decryption key, however, remains the same.

ALGORITHM 2. *Customized encryption key generation*
Input: *(i)* **L** *- the set of levels, (ii)* $<$ *- the dominance relation, (iii)* L_i *- the level for which the default encryption key is being generated, (iv)* **A** *- the set of levels that do not dominate* L_i *but who are to be given access, (v)* **D** *- the set of levels that dominate* L_i *and who are to be denied access, (vi)* **K** *- the set of default encryption keys.*
Output: Kc_i *- the customized key generated for level* L_i.

Procedure *GenerateCustomEncKey*(**L**,$<$,L_i, **D**, **A**, **K**)
begin
  $allow := \{\}$;
  $Nc_i := N_i$;
  while A $\neq$ $allow$ **do**
  begin
    for *each* $L_j \in$ **A** $-$ $allow$ *that is minimal* **do**
    begin
      $Nc_i := N_i * N_j$
      // Deny access to ancestors of L_j
      for *each parent* L_k *of* L_j **do**
        $Nc_i := Nc_i / N_k$
    end
    $allow := allow \cup L_j$
  end
  $deny := \{\}$;
  while D $\neq$ $deny$ **do**
  begin
    for *each* $L_j \in$ **D** $-$ $deny$ *that is minimal* **do**
    begin
      $Nc_i := N_i / N_j$
      // Give access to ancestors of L_j
      for *each parent* L_k *of* L_j **do**
        $Nc_i := Nc_i * N_k$
      // If N_k *has been included multiple times:*
      for *each parent* L_k *of* L_j **do**
      begin
        c := no. of paths from L_k *to* L_j
        $Nc_i := Nc_i / (N_k)^{c-1}$
      end
      $deny := deny \cup L_j$
    end
  end
  $Kc_i =< e, Nc_i >$
  return Kc_i
end

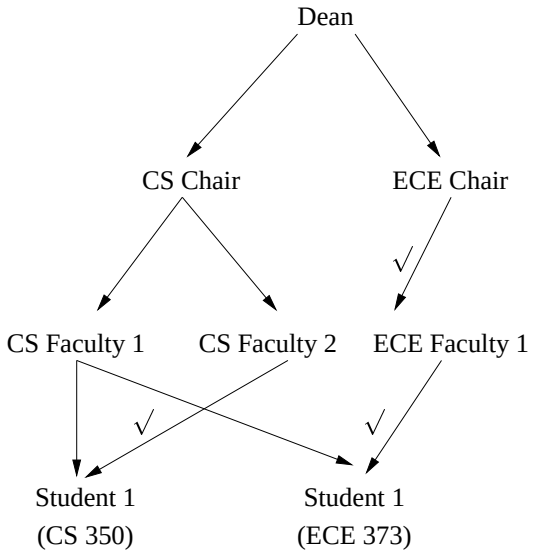

Figure 3: Access Requirements for Individual Courses

5.4 Example 2: Individual Course Grade Information

The organizational hierarchy in this case is the same as before (please refer to figure 2). We need to maintain information about the grades each student receives in each of the courses he/she has taken. The access requirements are complicated in this case:

1. Each student is able to view his or her grades for the courses he or she has taken.
2. The faculty offering the course is able to see the grades for all students who have taken the course under the faculty.
3. The chair of the department which has offered the course can view the grades of the students who have taken that course.
4. A student's advisor is able to see his grades in all the courses.
5. The student's department chair can view the student's grades for all the courses.
6. The dean is able to view the grades of all students that have taken a course offered by a department in the college.

Suppose $Student_1$ takes two courses: (i) CS 350 offered by the Computer Science Department and taught by $CS\ Faculty_2$ and (ii) ECE 373 offered by the ECE Department and taught by $ECE\ Faculty_1$. This is shown in figure 3. Note that figure 3 shows only the relevant portion of the hierarchy given in figure 2.

The access requirements are as follows:

1. $Student_1$'s CS 350 grade can be viewed by $Student_1$, $CS\ Faculty_1$, $CS\ Faculty_2$, *CS Chair*, *Dean*.
2. $Student_1$'s ECE 373 grade can be viewed by $Student_1$, $CS\ Faculty_1$, $ECE\ Faculty_1$, *ECE Chair*, *CS Chair*, *Dean*.

Note that, in this case the access requirements do not match the organizational hierarchy given in figure 2. More access is required than permitted by the organizational hierarchy - for example, $ECE\ Faculty_1$ must be given access to the ECE 373 grades of $Student_1$.

5.4.1 Access Control for Example 2

In this case the access patterns do not match the organizational structure. For example, *CS Faculty*$_2$ must have access to the CS 350 grade of *Student*$_1$ even though *CS Faculty*$_2$ is not his advisor. Thus default keys cannot be used and custom keys are required to encrypt the grades obtained in the individual courses.

1. Key for encrypting *Student*$_1$'s CS 350 grade:
   $Kc_{S_{1,C350}} =< e, N'_{Dean} * N'_{C_{Chair}} * N'_{CF_1} * N'_{CF_2} * N_{S_1} >$
2. Key for encrypting *Student*$_1$'s ECE 373 grade:
   $Kc_{S_{1,E373}} =< e, N'_{Dean} * N'_{C_{Chair}} * N'_{E_{Chair}} * N'_{CF_1} * N'_{EF_1} * N_{S_1} >$

If key $Kc_{S_{1,C350}}$ is used for encrypting CS 350 grade of *Student*$_1$, then the encrypted grade can be decrypted by any of the default decryption keys of *Student*$_1$, *CS Faculty*$_1$, *CS Faculty*$_2$, *CS Chair* and the *Dean*. If key $Kc_{S_{1,E373}}$ is used for encrypting ECE 373 grade of *Student*$_1$, then the encrypted grade can be decrypted by any of the default decryption keys of *Student*$_1$, *CS Faculty*$_1$, *ECE Faculty*$_1$, *CS Chair*, *ECE Chair* and the *Dean*.

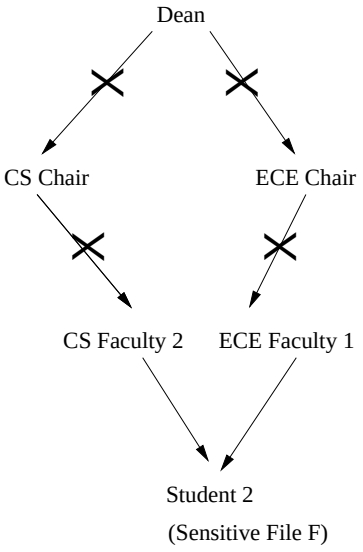

Figure 4: Access Requirements for a Sensitive Project

5.5 Example 3: Sensitive Project Information

As a part of the curriculum, the students are required to do a Software Design Project. Some of these projects involve proprietary data whose disclosure should be kept to a minimum level. Thus, there is a need for encrypting the files associated with the project. The access requirements are as follows.

1. The faculty members who advise the student on this project have access to the files.
2. The student has access to the files.
3. No other person is given access to the files.

Student$_2$ is working on a project with faculty members *CS Faculty*$_2$ and *ECE Faculty*$_1$. File F contains sensitive information which only the student and the project advisors can view. This case is illustrated in figure 4. Note that in figure 4 the entire organizational hierarchy is not shown – only the part pertinent to the example is given.

The access requirements are as follows:

1. Sensitive file F can be viewed by *Student*$_2$, *CS Faculty*$_2$ and *ECE Faculty*$_1$.

This is an example where the access requirement does not follow the organizational hierarchy. People higher up in the hierarchy (the Dean, the Department Chairperson) is not given the access to these files.

5.5.1 Access Control for Example 3

In this case the organizational structure is that given in figure 2. The sensitive file F must be protected such that only the faculty advisors (*CS Faculty*$_2$, *ECE Faculty*$_2$ and *ECE Faculty*$_1$) and the student (*Student*$_2$) have access to this file. No other person can have access. Thus, for protecting the project work the default encryption key is not adequate. Customized encryption key must be used.

The student encrypts file F using the following customized encryption key Kc_{S_2}. The customized encryption key Kc_{S_2} is generated using Algorithm 2. For this example,

$$Kc_{S_2} =< e, N'_{CF_2} * N'_{EF_1} * N'_{S_2} >$$

The encrypted file F can be decrypted by the default decryption keys of *CS Faculty*$_2$, *ECE Faculty*$_2$, *ECE Faculty*$_1$ or *Student*$_1$.

6. SECURITY OF THE PROPOSED MECHANISM

Our scheme is based on the RSA cryptosystem. Its security is based on the difficulty of factoring large prime numbers. We do need to mention, however, that the low exponent attack on the RSA cryptosystem [10] does not arise in our case. The low exponent attack occurs in the following way: suppose the same message m is encrypted with different keys sharing the same exponent. Let the exponent $e = 3$ and the different keys are $K_1 =< e, N_1 >$, $K_2 =< e, N_2 >$, $K_3 =< e, N_3 >$, etc. By using the Chinese Remainder Theorem [13] an attacker can get m^e. Now if he can guess e correctly, then by extracting the eth root of m^e, he can obtain m.

To avoid this problem, we choose a large exponent e (e is substantially larger than the number of levels). Since the complexity of raising a number to the e^{th} power grows as $\log e$, choosing a large exponent does not significantly increase the computational complexity. Also in our mechanism, the data is appropriately encrypted and stored only in one place. Having multiple copies of same data encrypted with different keys does not arise in our case.

7. ORGANIZATIONAL CHANGES AFFECTING KEY MANAGEMENT

As outlined in Section 5, the default encryption keys generated are dependent on the hierarchical structure of the organization. If restructuring takes place in the organization, the Hasse diagram representing the personnel hierarchy will be modified resulting in the change of the default encryption keys. In this section we give algorithms that result in the modification of default encryption keys when the organization structure is changed.

Any restructuring can be expressed as modifications to the Hasse Diagram. A Hasse diagram can be modified by using combinations of the four primitive operations

1. adding an edge between existing nodes – this corresponds to the scenario when a new hierarchical relationship is established between two existing persons in the organization.
2. deleting an edge from an existing node – this corresponds to the scenario when an existing hierarchical relationship is broken.

3. adding a node – this corresponds to the case when a new person joins the organization.

4. deleting a node – this corresponds to the case when a person leaves the organization.

For each of these operations we give an algorithm stating how the default encryption key must be changed because of the operation. The default decryption key, however, remains the same.

ALGORITHM 3. *Default enc. key change with edge insertion*
Input: *(i)* **L** *- the set of levels, (ii)* $<$ *- the dominance relation, (iii)* (i,j) *- the new directed edge that is to be inserted from level i to level j, (iv)* **K** *- the set of default encryption keys.*
Output: **K** *- the set containing the default encryption key for each level.*

Procedure *ChangeDefEncKeysEdgeIns(***L**,$<$,(i,j), **K***)*
begin
  $N_j := N_i * N_j$;
  for *each descendant k of j* **do**
    if(i,j) *insertion results in a new path from i to k* **then**
      $N_k := N_k * N_i$
  for *each* $L_i \in$ **L do**
  begin
    $K_i :=< e, N_i >$
    $\mathbf{K} := \mathbf{K} \cup K_i$
  end
  return K
end

ALGORITHM 4. *Default enc. key change with edge deletion*
Input: *(i)* **L** *- the set of levels, (ii)* $<$ *- the dominance relation, (iii)* (i,j) *- the directed edge from level i to level j that will be removed, (iv)* **K** *- the set of default encryption keys.*
Output: **K** *- the set containing the default encryption key for each level.*

Procedure *ChangeDefaultEncKeysEdgeDel(***L**,$<$,(i,j), **K***)*
begin
  $N_j := N_j / N_i$;
  for*each descendant k of j* **do**
    if *edge* (i,j) *deletion results in no path from i to k* **then**
    begin
      $N_k := N_k / N_i$ *//this will eliminate access to parents of* L_i
      for*each parent l of i* **do**
        if *there is a path from l to k after deleting* (i,j) **do**
          $N_k := N_k * N_l$
    end
  for *each* $L_i \in$ **L do**
  begin
    $K_i :=< e, N_i >$
    $\mathbf{K} := \mathbf{K} \cup K_i$
  end
  return K
end

ALGORITHM 5. *Default key generation with node insertion*
Input: *(i)* **L** *- the set of levels, (ii)* $<$ *- the dominance relation, (iii) i - the node that will be added, (iv)* **K** *- the set of default encryption keys, (v)* $\mathbf{K^{-1}}$ *- the set of default decryption keys.*
Output: *(i)* **K** *- the set containing the default encryption key for each level, (ii)* $\mathbf{K^{-1}}$ *- the set containing the default decryption key for each level.*

Procedure *AddDefaultEncKeyNodeIns(***L**,$<$,*i*,**K**, $\mathbf{K^{-1}}$*)*
begin
  Choose random $N_i^{'}$ *that is relatively prime to existing* N_k
  $N_i := N_i^{'}$
  Get the exponent e of any key K_k *in* **K**
  $K_i :=< e, N_i >$
  $K_i^{-1} :=< d_i, N_i >$ *where* $e * d_i \equiv 1 mod \phi(N_i)$
  $\mathbf{K} := \mathbf{K} \cup K_i$
  $\mathbf{K^{-1}} := \mathbf{K^{-1}} \cup K_i^{-1}$
end

ALGORITHM 6. *Default keys removal with node deletion*
Input: *(i)* **L** *- the set of levels, (ii)* $<$ *- the dominance relation, (iii) i - the node that will be added, (iv)* **K** *- the set of default encryption keys, (v)* $\mathbf{K^{-1}}$ *- the set of default decryption keys.*
Output: *(i)* **K** *- the set containing the default encryption key for each level, (ii)* $\mathbf{K^{-1}}$ *- the set containing the default decryption key for each level.*

Procedure *RemoveDefaultKeysNodeDel(***L**,$<$,*i*,**K**, $\mathbf{K^{-1}}$*)*
begin
  if *there are no edges incident on node i* **then**
  begin
    $\mathbf{K} := \mathbf{K} - K_i$
    $\mathbf{K^{-1}} := \mathbf{K^{-1}} - K_i^{-1}$
  end
end

8. AN ALTERNATE SOLUTION

The problem of access control in a hierarchy can be solved using an alternative solution[1] that can potentially have lesser computational requirements. Everybody in the organization has a public-private key pair. Suppose an employee wants to share a message *m* with his *n* superiors, denoted by S_1, S_2, ..., S_n. For each superior S_i, the employee encrypts the message *m* with the public key of superior S_i. Thus, he performs $(n+1)$ encryptions (*n* for his superiors and one for himself). He stores these $(n+1)$ encrypted messages. Each superior S_i can use his private key to retrieve the message.

This alternate solution has a problem: multiple encrypted copies of data must be stored. Storing multiple copies of encrypted data can be a source of inconsistency. For example, suppose the employee decides to change *m* to $\hat{m}$. After making this change, the employee is supposed to encrypt $\hat{m}$ with the public keys of his *n* superiors. However, the employee forgets to encrypt $\hat{m}$ for superiors S_j and S_k. In such a case superiors S_j and S_k will be accessing the previous version of the data which is *m*. This source of inconsistency associated with redundant data does not arise in our case because there is only one copy of the encrypted data. Moreover, keeping multiple encrypted copies of the same data leads to more exposure for the data which may not be desirable. This problem does not arise in our case.

Our solution has another advantage, namely, *mutual access awareness* – each person having the encrypted data has the knowledge of who else can view this data. For any data object we can have the *need-to-know* list which specifies the persons who can access the document. Anyone having this list can verify that only those persons in the list and no one else can decrypt the corresponding data object. This is not possible for the alternate solution.

One might, however, argue that our scheme is more computation intensive for large hierarchies. This is because the base *N* of the

[1]The authors would like to thank the anonymous reviewer for this alternate solution.

default encryption key increases with the number of levels in the hierarchy. However there are techniques (for example using Fast Fourier Transforms) by which the encryption can be done in an efficient manner [5]. These techniques are especially useful if the base of the keys are large. Details of computational complexity will be treated in a future work.

9. CONCLUSION AND FUTURE WORK

In this paper we have presented a new encryption technique for securing information in a database. The implementation that we propose is completely general; we have shown how the different access control policies in an organization can be implemented by our technique.

A lot of work is yet to be done. The first step is to implement the algorithms that have been proposed; this experience will help us in detecting subtle flaws that we may have overlooked. Performance analysis and scalabilty studies need to be done before our method can be used in real world scenarios. Finally, we wish to show how the discretionary and the mandatory access control policies [22] of an organization can be implemented using the technology that we proposed.

The need for hierarchical access control arises in other contexts as well. For example, different kinds of hierarchies, such as, class composition hierarchy and class inheritance hierarchy arises in object-oriented database systems [16, 21]. We need to investigate whether our mechanism can be applied to implement the access control policies [3, 7, 11] (such as, visibility from above and visiblity from below policies) desirable in such hierarchies.

10. REFERENCES

[1] S. G. Akl and P. D. Taylor. Cryptographic Solution to a Multilevel Security Problem. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 237–249. Plenum Press, NY, August 1982.

[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):239–248, 1983.

[3] E. Bertino, S. Jajodia, and P. Samarati. Access Controls in Object-Oriented Database Systems: Some Appproaches and Issues. In *Advanced Database Concepts and Research Issues*, volume 759 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[4] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security* . Addison-Wesley, Reading, MA, 1994.

[5] Ç. Koç. High-Speed RSA Implementation. Technical Report TR 201, RSA Laboratories, RSA Data Security, Inc., Nov. 1994.

[6] G. C. Chick and S. E. Tavares. Flexible Access Control with Master Keys . In G. Brassard, editor, *Advances in Cryptology: Proceedings of Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 316–322. Springer-Verlag, 1990.

[7] K. R. Dittrich, M. Hartig, and H. Pfefferle. Discretionary Access Control in Structurally Object-Oriented Database Systems. In *Database Security II: Status and Prospects*, IFIP, pages 105–121. Elsevier Science, North Holland, 1989.

[8] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, Reading, MA, 3rd edition, 2002.

[9] L. Harn and H. Y. Lin. A Cryptographic Key Generation Scheme for Multi-level Data Security. *Computer & Security*, 9(6):539–546, 1990.

[10] B. Kaliski and M. Robshaw. The Secure Use of RSA. *CryptoBytes*, 1(3):7–13, 1995.

[11] M. Larrondo-Petrie, E. Gudes, H. Song, and E. B. Fernandez. Security Policies in Object-Oriented Databases. In *Database Security III*, IFIP, pages 257–269. Elsevier Science, North Holland, 1990.

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

[13] I. Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers*. John Wiley and Sons, 4th edition, 1980.

[14] K. Ohta, T. Okamoto, and K. Koyama. Membership Authentication for Hierarchical Multigroup using the Extended Fiat-Shamir Scheme. In I. B. Damgård, editor, *Advances in Cryptology: Proceedings of EuroCrypt '90*, volume 473 of *Lecture Notes in Computer Science*, pages 316–322. Springer-Verlag, 1991.

[15] I. M. Olson and M. D. Abrams. Information Security Policy. In M. D. Abrams, S. Jajodia, and H. J. Podell, editors, *Information Security: An Integrated Collection of Essays*, pages 160–169. IEEE Computer Society Press, 1995.

[16] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A Model of Authorization for Next-Generation Database Systems. *ACM Transactions on Database Systems*, 16(1):88–131, March 1991.

[17] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):637–647, 1978.

[18] R. S. Sandhu. Cryptographic Implementation of a Tree Hierarchy for Access Control. *Information Processing Letters*, 27(2):95–98, 1988.

[19] E. R. Scheinerman. *Mathematics: A Discrete Introduction*. Brooks/Cole, Pacific Grove, CA, 1st edition, 2000.

[20] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice-Hall, second edition, 1999.

[21] M. B. Thuraisingham. Mandatory Security in Object-Oriented Database Systems. In *Proceedings of the International Conference on Object-Oriented Programming Systems, Languages and Applications*, pages 203–210, New Orleans, LA, October 1989.

[22] U.S. Department of Defense Computer Security Center. *Trusted Computer System Evaluation Criteria (The Orange Book)*, Dec. 1985.

[23] Y. Zheng, T. Hardjono, and J. Seberry. New Solutions to the Problem of Access Control in a Hierarchy. Technical Report Preprint 93-2, Department of Computer Science, University of Wollongong, February 1993. Available from `http://citeseer.nj.com/28503.html`.