Journal of King Saud University – Computer and Information Sciences (2013) **25**, 219–228

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

A novel proxy signature scheme based on user hierarchical access control policy

Ashok Kumar Das *, Ashish Massand, Sagar Patil

Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, Andhra Pradesh 500 032, India

Received 14 May 2012; revised 27 November 2012; accepted 2 December 2012
Available online 13 December 2012

KEYWORDS

Proxy signature;
Hierarchical access control;
Polynomial interpolation;
Security;
Discrete logarithm;
Hash function

Abstract In this paper, we propose a new security protocol for proxy signature by a hierarchy of proxy signers. In this protocol, the original signer delegates his/her signing capability to a predefined hierarchy of proxy signers. Given the documents of a security class to be signed by the original signer, our scheme suggests a protocol for the hierarchy of proxy signers to sign the document on behalf of the original signer. The concept of hierarchical access control limits the number of people who could sign the document to the people who have the required security clearances. User in a security class requires two secret keys: one which identifies his/her security clearance, and that can also be derived by a user of upper level security clearance and second is his/her private key which identifies him/her as a proxy signer for the signature generation. We show that our scheme is efficient in terms of computational complexity as compared to the existing related proxy signature schemes based on the hierarchical access control. Our scheme also supports addition and deletion of security classes in the hierarchy. We show through security analysis that our scheme is secure against possible attacks. Furthermore, through the formal security analysis using the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool we show that our scheme is also secure against passive and active attacks.

1. Introduction

A digital signature can help to provide the integrity and authenticity of a message, such as a document, file, etc. This concept cannot however be applied directly to a situation where the original signer delegates his/her signing capability to a proxy signer to sign the documents on behalf of himself/herself. Mambo et al. (1996) introduced the concept of proxy signature. In a proxy signature scheme, the original signer delegates his/her signing capability to a proxy signer. The proxy signer signs the documents on behalf of original signer. A verifier can then verify that the document is signed by a proxy signer. Further, the verifier can also verify its validity and that the proxy signer has the capability to do so. More precisely, in a proxy signature

* Corresponding author. Tel.: +91 40 6653 1506; fax: +91 40 6653 1413.
E-mail addresses: iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in (A.K. Das), ashishsunil.massand@students.iiit.ac.in (A. Massand), sagar.patil@students.iiit.ac.in (S. Patil).
URLs: http://www.iiit.ac.in/people/faculty/ashokkdas/, https://sites.google.com/site/iitkgpakdas/ (A.K. Das).

Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

1319-1578 © 2012 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jksuci.2012.12.001

scheme there is an original signer who delegates his/her signing capability to a proxy signer by issuing a proxy key. The proxy signer then signs a message on behalf of the original signer. From the proxy signature signed by a proxy signer using the proxy key, any verifier can check the original signer's digital delegation as well as the proxy signer's digital signature. Several proxy signature schemes have been proposed in the literature (Mambo et al., 1996; Shao, 2003; Tan et al., 2002). A survey on proxy signatures can be found in Das et al. (2009).

Hierarchical access control is an interesting area of research in the field of cryptography. In such a system, there is a predefined hierarchy which gives the information of the security clearance of a user belonging to a particular security class. This hierarchy can be represented by a partially ordered set (poset). A hierarchy is constructed by dividing the users into a number n of disjoint security classes, say $SC_1, SC_2, \ldots, SC_n$ which are partially ordered with a binary relation $\leqslant$ defined on the set $U = \{SC_1, SC_2, \ldots, SC_n\}$. In such a set, $SC_i \leqslant SC_j$ implies that SC_j has the higher security clearance than that for SC_i, and the documents accessed by SC_i can also be accessed by SC_j, but the reverse is not allowed. Consider an example of user hierarchy shown in Fig. 1. In this figure, a hierarchy is constructed by a set of five disjoint security classes SC_1, SC_2,SC_3, SC_4, and SC_5. SC_1 has the highest security clearance level. The security classes SC_2, SC_4, and SC_5 at the bottom level contain the lowest security clearances. SC_1 can access the secret information possessed by SC_2, SC_3, SC_4 and SC_5. SC_3 can again access to the secret information held by SC_4 and SC_5.

Akl and Taylor (1983) proposed the first-ever hierarchical access control based key assignment scheme. Chung et al. (2008) proposed a key management and derivation scheme based on the elliptic curve cryptosystem. In their approach, the secret key of each security class can be determined by a trusted centralized authority. Their scheme has the ability to solve the dynamic key management efficiently and flexibly. However, it is recently shown in Das et al. (2012) that their scheme is vulnerable to exterior root finding attack in which an attacker (adversary) who is not a user in any security class in a user hierarchy can derive the secret key of a security class by using the root finding algorithm. Many dynamic access control schemes have been proposed in the literature, some of them are Lin (1997), Akl and Taylor (1983), Shen and Chen (2002), Zhong (2002), Sandhu (1988), Giri and Srivastava (2007, 2008), Wu and Wei (2005), and Odelu et al. (in press, 2012, 2013).

In this paper, we devise a new proxy signature scheme based on hierarchical access control. The main motivation behind our new approach is as follows. In the absence of an original signer, the signing capability can be delegated to a group/set of users in a user hierarchy wherein the security clearances of those users are predefined. Note that the user hierarchy contains only the security classes for proxy signers, not for any original signers. For example, in an organization some important documents can be singed on behalf of the head of that organization. There is a predefined hierarchy of security clearances among the members of that organization. A document of security clearance containing in the security class SC_i can be signed by that security class on behalf of the head of the organization. For example, in Fig. 1, the documents containing to the security class SC_5 can be singed by either SC_1 or SC_3 or SC_5. The verifier needs to verify the following two conditions: (i) whether the right person has signed the document and (ii) whether the user has been delegated the signing capability to sign the document.

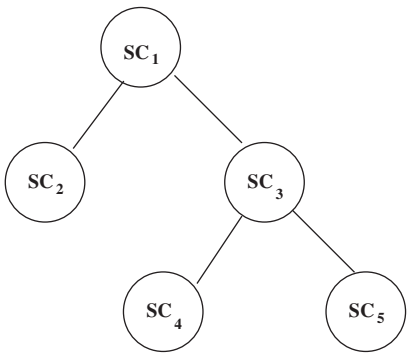

Figure 1 An example of a poset in a user hierarchy.

A widely accepted list of required security requirements for a proxy signature is given below (Das et al., 2009):

- Strong unforgeability: A designated proxy signer can only have the ability to create a valid signature on behalf of the original signer, whereas the original signer and other third parties cannot create a valid proxy signature.
- Strong identifiability: Anyone can determine the identity of the proxy signer from the corresponding proxy signature.
- Strong undeniability: This property tells that once a proxy signer has created a valid proxy signature on behalf of the original signer, he/she cannot deny later the signature creation.
- Verifiability: This means that the verifier can be convinced of the original signer's agreement from the proxy signer.
- Distinguishability: Anyone can distinguish the proxy signatures from the normal signatures.
- Secrecy: The original signer's private key must not be derived from any information, such as the sharing of the proxy key, proxy signatures, etc.
- Prevention of misuse: The proxy signer cannot use the proxy key for any other purpose than it is made for.

Giri et al. (2009) introduced the concept of proxy signature based on hierarchical access control. Their scheme contains the hierarchical access control scheme followed by a proxy signature scheme. A trusted central authority is responsible for generating and assigning keys to the users in the hierarchy. Their access control scheme is based on Newton's interpolating polynomials. Further, their scheme is also secure against different attacks.

Our scheme is different from the scheme proposed by Giri et al. In our scheme, we define a set of designated proxy signers in the user hierarchy selected by the original signer. Based on the document pertaining to a particular security level class, the original signer selects a user, called the proxy signer, from that set based on their availability and work load who can sign those documents on behalf of the original signer. In our scheme, the secret key of the proxy signer is embedded into the proxy signature so that any verifier can verify that the right person only has signed the document. We show through analysis and simulation using the AVISPA tool that our scheme is secure against possible attacks. Furthermore, our scheme is also efficient as compared to Giri et al.'s scheme.

The rest of this paper is organized as follows. In Section 2, we discuss some mathematical preliminaries which are needed

to describe and analyze our scheme. In this section, we describe briefly the mathematical background on the one-way hash function, the polynomial evaluation over a finite field and the discrete logarithm problem. In Section 3, we introduce our new proxy signature scheme which uses the concept of hierarchical access control policy. In Section 4, we analyze the performance and security for our proposed scheme. In Section 5, we show through simulation that our scheme is secure against passive and active attacks. For this purpose, we use the AVISPA tool for formal security analysis in this paper. In Section 6, we compare the performance of our scheme with Giri et al.'s scheme. Finally, we conclude the paper in Section 7.

2. Preliminaries

In this section, we describe briefly the mathematical background on the one-way hash function, the polynomial evaluation over a finite field and the discrete logarithm problem, which are useful for discussing and analyzing our proposed scheme.

2.1. Hash functions

A one-way hash function $H:\{0,1\}^* \rightarrow \{0,1\}^n$ takes an arbitrary-length input $x \in \{0,1\}^*$, and produces a fixed-length (say, n-bits) output $H(x) \in \{0,1\}^n$, called the message digest or hash value. The hash function can be applied to the fingerprint of a file, a message, or other data blocks, and it has the following important attributes (Stallings, 2003):

(i) The hash function, H can be applied to a data block of all sizes.
(ii) For any given input x, $H(x)$ is relatively easy to compute which enables easy implementation in software and hardware.
(iii) The output length of $H(x)$ is fixed.
(iv) From a given hash value $y = H(x)$ and the given hash function $H(\cdot)$, it is computationally infeasible to derive x.
(v) For any given input x, finding any other input $y(\neq x)$ such that $H(y) = H(x)$ is computationally infeasible.
(vi) Finding a pair of inputs (x, y), $(x \neq y)$, such that $H(x) = H(y)$ is again computationally infeasible.

An example of a one-way hash function is SHA-1 (Secure Hash Standard, 1995), which has the above desired properties (i) to (vi). However, the National Institute of Standards and Technology (NIST) does not recommend SHA-1 for top secret documents. Further, in 2011, Manuel showed collision attacks on SHA-1 (Manuel, 2011). As in Das (2012b,a) one can also use the recently proposed one-way hash function, Quark (Aumasson et al., 2010). Quark is a family of cryptographic hash functions which is designed for extremely resource-constrained environments like sensor networks and radio-frequency identification (RFID) tags. Like most one-way hash functions, Quark can be used as a pseudo-random function (PRF), a message authentication code (MAC), a pseudo-random number generator (PRNG), a key derivation function, etc. Quark is shown to be much efficient hash function than SHA-1. However, in this paper, we use SHA-2 as the secure one-way hash function in order to achieve top security. We may use only 160-bits from the hash digest output of SHA-2.

2.2. Polynomial evaluation in GF(q)

Given a polynomial $f(x)$ of degree t, where $f(x) = a_t x^t + a_{t-1}x^{t-1} + \cdots + a_1 t + a_0 \in GF(q)[x]$ whose coefficients a_i's are from the finite field, Galois field $GF(q)$,q being a prime. By applying Horner's rule (Cormen et al., 2010), we require t modular multiplications and t modular additions in order to find the value of $f(x)$ at a point $x = x_0$.

In a special case, if $f(x)$ is of the form $f(x) = \prod_{i=0}^{t} (x - a_i) + b \pmod q$, where a_i's and b are from $GF(q)$, then to find the value of b, we need any one of the values a_i's. Using that a_i, we can find b by $b = f(a_i) \pmod q$, for some i.

2.3. Computational problems

In this section, we describe two computational problems: Discrete Exponentiation Problem (DEP) and Discrete Logarithm Problem (DLP).

2.3.1. Discrete exponentiation problem (DEP)

Given a multiplicative group G and an element $g \in G$ having order n. Then, compute $y = g^x \bmod n$, for any given x. This problem is computationally easy/feasible even if n is large. This modular exponentiation can be done efficiently by using the repeated square-and-multiply algorithm with time complexity $O((\log_2 n)^3)$ (Delfs and Knebl, 2007).

2.3.2. Discrete logarithm problem (DLP)

Given an element g in a finite group G whose order is n, that is, $n = \#G_g$ (G_g is the subgroup of G generated by g) and another element y in G_g, find the smallest non-negative integer x such that $g^x = y$. It is relatively easy to calculate discrete exponentiation $g^x \pmod n$ given g, x and n, but it is computationally infeasible to determine x given y, g and n, when n is large.

3. The proposed hierarchical-based proxy signature scheme

In this section, we first list the notations to be used in our scheme. We then discuss the different phases of our scheme.

3.1. Notations

We use the notations listed in Table 1 for describing our proposed proxy signature scheme.

3.2. Different phases

Our scheme contains six phases: setup phase, delegation phase, proxy signature generation phase, proxy signature verification phase, and phases for addition of new security classes in the hierarchy and removal of existing security classes from the hierarchy.

3.2.1. Setup phase

In this phase, the system parameters for the original signer are selected. The central certification authority (CA) then chooses the system parameters for each security class in the hierarchy. This phase consists of the following steps:

Step 1. CA selects large primes p and q such that $p - 1$ is perfectly divisible by q, that is, $q \mid p - 1$.

Table 1 Notations used in the proposed scheme.

Symbol	Description
p	A large prime number
q	A larger prime such that $q—p-1$
A	Original signer
B	Proxy signer
V	Verifier
sk_i	Full secret key of security class, SC_i
s_i	Partial secret key of security class, SC_i
$F_i(x)$	Public polynomial of security class, SC_i
g	Generator of original signer, A
g_i	Generator of security class, SC_i
$S\|\|T$	Data S concatenates with data T
X_A	Secret key of original signer, A
X_B	Secret key of proxy signer, B
Y_A	Public key of original signer, A
Y_B	Public key of proxy signer, B
$H(\cdot)$	Secure one-way hash function
M_w	Warrant of a message, M

Step 2. CA then selects a generator $g \in Z_p^*$, where $Z_p^* = \{1, 2, \ldots, p-1\}$.

Step 3. Consider a user hierarchy where there are n security classes $SC_1, SC_2, \ldots, SC_n$ in the hierarchy with partially ordered relation $\leqslant$ defined on it. The CA then generates n generators $g_1, g_2, \ldots, g_n \in Z_q^*$ for n security classes in the user hierarchy, where $Z_q^* = \{1, 2, \ldots, q-1\}$.

Step 4. CA generates randomly the full secret key $sk_i \in Z_q^*$ and partial secret key $s_i \in Z_q^*$ for each security class SC_i in the user hierarchy.

Step 5. CA then calculates the public polynomial $F_j(x)$ for each security class SC_j in the hierarchy as $F_j(x) = \prod_{SC_i > SC_j}(x - g_j^{s_i}) + sk_j \pmod q$ as in Chung et al. (2008).

Step 6. CA sends (sk_i, s_i) to each security class SC_i in the hierarchy via a secure channel.

Step 7. CA also sends the public polynomial $F_i(x)$, q and g_i to each security class SC_i via a public channel. CA finally makes p, q, g, g_j's, $F_j(x)$'s and the hash function $H(\cdot)$ as public.

An example. Consider again the user hierarchy shown in Fig. 1. The public polynomials for the security classes constructed in Step 5 are as follows:

$$SC_1 : F_1(x) = (x - g_1^{s_0}) + sk_1 \pmod q, \text{ where } s_0 \text{ is given by CA}$$
$$SC_2 : F_2(x) = (x - g_2^{s_1}) + sk_2 \pmod q$$
$$SC_3 : F_3(x) = (x - g_3^{s_1}) + sk_3 \pmod q$$
$$SC_4 : F_4(x) = (x - g_4^{s_1})(x - g_4^{s_3}) + sk_4 \pmod q$$
$$SC_5 : F_5(x) = (x - g_5^{s_1})(x - g_5^{s_3}) + sk_5 \pmod q$$

3.2.2. Delegation phase

In order to delegate the signing capability to a proxy signer, the original signer A executes the following steps:

Step 1. The original signer A selects a random or pseudorandom integer $X_A (1 < X_A < p-1)$ and computes a public key $Y_A = g^{X_A} \pmod p$, where g is the generator in Z_p^*.

Step 2. A selects a random or pseudorandom integer $k \in Z_q^*$ and computes $r = g^k \pmod p$.

Step 3. Suppose the documents to be signed belonging to a security class in the hierarchy is known to the original signer A. However, as pointed out earlier in this paper that the original signer A is not any user of any security classes in the user hierarchy and the user hierarchy consists of only the security classes of proxy signers. A then selects a set P of designated proxy signers who are the users in security classes of the hierarchy. Since the document class is known to the original signer A, so the set P is also known to the original signer A. A can choose a user in a security class from P for which the user is available during that time period and also that user is not overloaded with other works. Let the proxy signer be a user in a security class $SC_i \in P$ and we call that user as the proxy signer, B in our scheme. For example, in Fig. 1, if the documents of SC_5 need to be signed, then A can select any one of the users in SC_1, SC_3 and SC_5 as a proxy signer.

Step 4. A then calculates $s = X_A r + k\ M_w \pmod q$, where M_w is a warrant message which includes the identities of the original signer and the proxy signer, and also an expiration date.

Step 5. The original signer A selects a proxy signer B from the set P of designated proxy signers and sends the message $\langle r, s, M_w \rangle$ to B via a secure channel.

3.2.3. Proxy signature generation phase

In order to generate signature on documents belonging to a security class $SC_c \in P$, the proxy signer B who is a user in security class $SC_i \in P$ $(SC_c \leqslant SC_i)$ proceeds with the following steps:

Step 1. On receiving the tuple (r, s, M_w) from the original signer A, the proxy signer B verifies whether $g^s = r^{M_w} Y_A^r \pmod p$ holds or not. If it holds, B accepts the tuple (r, s, M_w) as a valid tuple. Note that

$$g^s = g^{X_A \cdot r + k \cdot M_w \pmod q} \pmod p$$
$$= (g^{X_A})^r \cdot (g^k)^{M_w} \pmod p = r^{M_w} Y_A^r \pmod p.$$

Step 2. In order to compute the secret key sk_c of the security class SC_c, B first computes $u = g_c^{s_i} \pmod q$ using the public generator g_c of SC_c and its own class SC_i's partial secret key s_i. B then computes the full secret key sk_c as $sk_c = F_c(u) \pmod q = F_c\left(g_c^{s_i} \pmod q\right) \pmod q$ by evaluating the public polynomial $F_c(x)$ of the security class SC_c at the point u.

Step 3. B selects a random or pseudorandom integer $X_B (1 < X_B < q-1)$ and computes a public key $Y_B = g^{X_B} \pmod p$, where g is the generator in Z_p^*.

Step 4. Let M be the message (document of SC_c) to be signed by B. B computes $H(M\|M_w\|sk_c\|s_i)$, $s' = s + H(M\|M_w\|sk_c\|s_i)X_B \pmod q$ and $t = g^{s'} \pmod p$.

Step 5. B then computes the hash value $H(M\|M_w\|t)$.
The proxy signature on the message M is considered as the tuple $(M, M_w, H(M\|M_w\|sk_c\|s_i), r, H(M\|M_w\|t))$.

Step 6. B finally sends the message $\langle M, M_w, H(M\| M_w\|sk_c\|\| s_i), r, H(M\|M_w\|t)\rangle$ to the verifier or receiver V via a public channel.

3.2.4. Proxy signature verification phase

In order to verify the proxy signature on the message M, the verifier V does the following steps:

Step 1. V first computes $t' = r^{M_w} \cdot Y_A^r \cdot Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p$, using the received r, $H(M\|M_w\|sk_c\|s_i)$, M_w, and the public keys Y_A and Y_B.

Step 2. V then computes the hash value $H(M\|M_w\|\ t')$ using the computed value of t' and the received message M.

Step 3. Finally, V verifies whether the computed hash value $H(M\|M_w\|t')$ matches with the received hash value $H(M\ \|M_w\|t)$. If there is a match, V accepts B's signature as a valid signature; otherwise, V rejects the signature of B.

We have summarized the delegation, proxy signature generation and proxy signature verification phases of our scheme in Table 2.

3.2.5. Addition of new security classes in hierarchy

Some times we require to add/remove security classes into the hierarchy. Therefore, dynamic access control must be provided. In this phase, we show how a new security class can be added to the hierarchy. Suppose a new security class SC_k will be inserted into the hierarchy such that $SC_i \geqslant SC_k \geqslant SC_j$. CA renews the secret keys sk_j of successors SC_j of the newly added security class SC_k. CA also changes the public polynomial $F_j(x)$ with $F'_j(x)$ of SC_j. CA needs the following steps:

Step 1: CA updates the partial relationships that follow when the security class SC_k joins the hierarchy.

Step 2: CA selects the full secret key $sk_k \in Z_q^*$, the partial secret key $s_k \in Z_q^*$ randomly and generator $g_k \in Z_q^*$ for SC_k.

Step 3: For all SC_i that satisfies the relationship $SC_i \geqslant SC_k$ when the new class SC_k is inserted in the hierarchy, CA computes $g_k^{s_i} \pmod q$.

Step 4: CA then computes the public polynomial $F_k(x)$ for SC_k as $F_k(x) = \prod_{SC_i > SC_k}(x - g_k^{s_i}) + sk_k \pmod q$

Step 5: For all SC_i such that $SC_i \geqslant SC_k$ and for all SC_j such that $SC_i \geqslant SC_k \geqslant SC_j$ when the new class SC_k is inserted in the hierarchy, CA replaces the secret key sk_j with sk'_j and computes the updated public polynomial $F'_j(x)$ as $F'_j(x) = \prod_{SC_i > SC_k > SC_j}(x - g_j^{s_i})\ (x - g_j^{s_k}) + sk'_j \pmod q$.

Step 6: Finally, CA replaces $F_j(x)$ with $F'_j(x)$, and sends sk'_j to SC_j via a secure channel, and announces publicly $F'_j(x)$. CA also sends sk_k and s_k to SC_k via a secure channel, and announces publicly $F_k(x)$.

Note that in order to resist the exterior root finding attack, we have replaced sk_j by sk'_j in the updated polynomial $F'_j(x)$ for SC_j.

An example. Consider the user hierarchy shown in Fig. 2, in which the new security class SC_6 is added to the existing hierarchy shown in Fig. 1. Then, the public polynomials for the security classes will be as follows:

$$SC_1 : F_1(x) = (x - g_1^{s_0}) + sk_1 \pmod q, \text{ where } s_0 \text{ is given by CA}$$
$$SC_2 : F_2(x) = (x - g_2^{s_1}) + sk_2 \pmod q$$
$$SC_3 : F_3(x) = (x - g_3^{s_1}) + sk_3 \pmod q$$
$$SC_4 : F_4(x) = (x - g_4^{s_1})(x - g_4^{s_3}) + sk_4 \pmod q$$
$$SC_5 : F'_5(x) = (x - g_5^{s_1})(x - g_5^{s_3})(x - g_5^{s_6}) + sk'_5 \pmod q$$
$$SC_6 : F_6(x) = (x - g_6^{s_1}) + sk_6 \pmod q$$

3.2.6. Removal of existing security classes in hierarchy

This phase remains similar to that as in Chung et al. (2008). Now, if an existing security class SC_k, such that the relationship $SC_i \geqslant SC_k \geqslant SC_j$ breaks up, wants to leave from a user hierarchy, CA not only directly revokes information related to SC_k, but also it needs to alter the accessing relationship be-

Table 2 Different phases of the proposed scheme.

Original signer (A)	Proxy signer (B)	Verifier (V)
Delegation phase Sends via a secure channel: $\overset{\langle r,s,M_w\rangle}{\longrightarrow}$		
Proxy signature generation phase	Verifies if $g^s = r^{M_w} Y_A^r \pmod p$. If it holds, B accepts (r, s, M_w) as valid tuple. Computes $sk_c = F_c(g_c^{s_i}) \pmod q, H(M\|M_w\|sk_c\|s_i)$, $s' = s + H(M\|M_w\|sk_c\|s_i)X_B \pmod q$, $t = g^{s'} \pmod p$ and $H(M\|M_w\|t)$. Sends via a public channel: $\overset{\langle M,M_w,H(M\|M_w\|sk_c\|s_i),r,H(M\|M_w\|t)\rangle}{\longrightarrow}$	
Proxy signature verification phase		Computes $t' = r^{M_w} Y_A^r Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p$ and $H(M\|M_w\|t')$. Verifies if $H(M\|M_w\|t') = H(M\|M_w\|t)$. If above holds, V accepts B's signature; otherwise, V rejects B's signature.

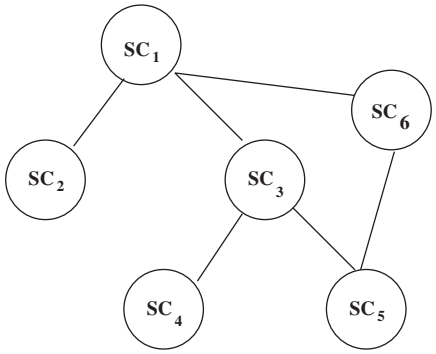

Figure 2 An example of a poset in a user hierarchy: when a new security class SC_6 is added to the existing hierarchy in Fig. 1.

tween the involved ex-predecessor SC_i and ex-successor SC_j of SC_k. CA requires the following steps for this purpose.

Step 1: CA needs to update the partial relationship that follows when SC_k is removed.

Step 2: For all SC_k such that the relationship $SC_k \geqslant SC_j$ holds, CA renews the secret key sk_j as sk'_j and the generator g_j as g'_j of SC_j. Then, for all SC_i such that the relationship $SC_i \geqslant SC_j$ holds, CA renews the relationship $SC_i \geqslant SC_j$ after removing SC_k, computes the public polynomial $F'_j(x)$ as $F'_j(x) = \prod_{SC_i > SC_j} (x - g_j^{'s_i}) + sk'_j \pmod q$ and replaces $F_j(x)$ with $F'_j(x)$.

Step 3: CA finally sends sk'_j to SC_j via a secret channel and announces g'_j and $F'_j(x)$ as public.

4. Analysis of the proposed scheme

In this section, we first show the correctness of our proposed scheme. We then analyze the computational overhead required for our scheme. Finally, we show that our scheme can tolerate different security attacks.

4.1. Correctness of the proposed scheme

Theorem 1. *If all the entities follow the scheme described in Section* 3*, the verification equation* $H(M\|M_w\|t') = H(M\|M_w\|t)$ *holds, where* $t' = r^{M_w} Y_A^r Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p$.

Proof. In order to prove $H(M\|M_w\|t') = H(M\|M_w\|t)$, it suffices to show that $t' = t$. Now,

$$
\begin{aligned}
t &= g^{s'} (mod p) = g^{s + X_B H(M\|M_w\|sk_c\|s_i) \pmod q} (\text{mod } p) \\
&= g^s g^{X_B H(M\|M_w\|sk_c\|s_i)} \pmod p \\
&= g^{X_A r + k M_w} \left(g^{X_B}\right)^{H(M\|M_w\|sk_c\|s_i)} \pmod p \\
&= \left(g^{X_A}\right)^r \left(g^k\right)^{M_w} Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p \\
&= r^{M_w} \cdot Y_A^r \cdot Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p = t'.
\end{aligned}
$$

Hence, the theorem is proved. □

4.2. Computational overhead

For analyzing the computational costs, we use the notations described in Table 3.

Table 3 Notations used for computational costs in the proposed scheme.

Symbol	Description
t_{mul}	Time taken by one modular multiplication operation
t_{exp}	Time taken by one modular exponentiation operation
t_h	Time taken to compute one hash value
t_{add}	Time taken for one modular addition operation
t_{poly}	Time taken for evaluating $F_i(x)$ at a point in $GF(q)$

```
role originalsigner (A, B, V   : agent,
            SKab: symmetric_key,
            H : hash_func,
            F : hash_func,
          Snd, Rcv: channel(dy))
played_by A
def=
       local State  : nat,
       K, G, P, Q, XA, YA, Mw :  text
       const alice_bob_na,  bob_alice_nb, bob_verifier_nc,
       verifier_bob_nd, subs1, subs2 : protocol_id

 init  State := 0

 transition
  1. State   = 0 ∧ Rcv(start) =|>
    State' := 1 ∧ K' := new()
              ∧ secret({K', XA}, subs1, A)
              ∧ Snd({exp(G, K').F(XA.exp(G, K').K'.Mw)}_SKab
                   .P.Q)
              ∧ witness(A, B, bob_alice_nb, K')
end role
```

Figure 3 Role specification in HLPSL for the original signer, A of our scheme.

```
role proxysigner (A,B,V   : agent,
            SKab: symmetric_key,
       H : hash_func,
            F : hash_func,
            Snd, Rcv: channel(dy))
played_by B
def=
       local State  : nat,
       K, G, P, Q, XA, YA, Gc, Si, SKc, XB, YB, M, Mw :  text
       const alice_bob_na,  bob_alice_nb, bob_verifier_nc,
           verifier_bob_nd, subs1, subs2 : protocol_id

 init  State := 0

 transition
  1. State   = 0 ∧ Rcv({exp(G, K').F(XA.exp(G, K').K'.Mw)}_SKab
                .P.Q) =|>
    State' := 1 ∧ M' := new()
              ∧ secret({K', XA}, subs1, A)
              ∧ secret({Si, SKc, XB}, subs2, B)
              ∧ Snd(M'.Mw.exp(G, K').H(M'.Mw.exp(G, (F(XA.exp(G, K')
                   .K'.Mw)).H(M'.Mw.SKc.Si).XB)).H(M'.Mw.SKc.Si).P.Q)
              ∧ witness(B, V, verifier_bob_nd, SKc)
end role
```

Figure 4 Role specification in HLPSL for the proxy signer, B of our scheme.

From the delegation phase described in Section 3.2.2, it is clear that the original signer requires the computational complexity $2t_{exp} + 2t_{mul} + t_{add}$ during this phase. The proxy signature generation phase described in Section 3.2.3 requires

```
role verifier  (A, B, V   : agent,
        H : hash_func,
             F : hash_func,
          Snd, Rcv: channel(dy))
played_by V
def=
       local State  : nat,
       K, G, P, Q, XA, YA, Gc, Si, SKc, XB, YB, M, Mw :  text
       const alice_bob_na,  bob_alice_nb, bob_verifier_nc,
            verifier_bob_nd, subs1, subs2 : protocol_id

  init  State := 0

 transition
  1. State   = 0 /\ Rcv(M'.Mw.exp(G, K').H(M'.Mw.exp(G, (F(XA.
                    exp(G, K').K'.Mw)).H(M'.Mw.SKc.Si).XB)).
                    H(M'.Mw.SKc.Si).P.Q) =|>
     State' := 1 /\ secret({K', XA}, subs1, A)
               /\ secret({Si, SKc, XB}, subs2, B)

end role
```

Figure 5 Role specification in HLPSL for the verifier, V of our scheme.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite\results\proxy_signature.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 8.28s
  visitedNodes: 0 nodes
  depth: 15 plies
```

Figure 6 Result of the analysis using OFMC of our scheme.

the computational complexity $6t_{exp} + t_{poly} + t_{add} + 2t_{mul} + 2t_h$, whereas the proxy signature verification phase described in Section 3.2.4 requires the computational complexity $3t_{exp} + 2t_{mul} + t_h$.

4.3. Security analysis

In this section, we show that our scheme satisfies all the security requirements of a proxy signature. The security requirements are discussed in the following subsections.

4.3.1. Unforgeability

Let an original signer being an attacker try to forge a proxy signature on any arbitrary message M'. Suppose the attacker chooses $M', M'_w, H\left(M' \| M'_w \| sk_c \| s_i\right)'$ and tries to find s' such that $t = g^{s'} \pmod p$, where $s' = s + H(M\|M_w\| \ sk_c\|s_i)X_B \pmod q$. The attacker knows the values M, M_w, p, q, s, Y_A and Y_B. It is noted that the original signer, who is an attacker in this case, is not any user of the hierarchy. Therefore, to retrieve sk_c of the document class SC_c of the hierarchy, the attacker needs to know the partial secret keys s_i of designated proxy signers' security classes. But then computing the secret key sk_c of the document class SC_c of the hierarchy from the hash value $H(M\|M_w\|sk_c\|s_i)$ knowing M and M_w is computationally infeasible due to one-way property of the hash function $H(\cdot)$. Moreover, in order to compute s' the attacker needs to know the private key X_B of the proxy signer B, which is again a computationally infeasible problem because X_B is embedded with $H(M\|M_w\|t)$ in s'. Also to determine directly X_B from $Y_B = g^{X_B} \pmod p$ is a computationally infeasible problem due to the difficulty of solving DLP (discussed in

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
 C:\progra~1\SPAN\testsuite\results\proxy_signature.if

GOAL
  As Specified

BACKEND
  CL−AtSe

STATISTICS

  Analysed   : 981 states
  Reachable  : 978 states
  Translation: 0.02 seconds
  Computation: 0.08 seconds
```

Figure 7 Result of the analysis using CL-AtSe of our scheme.

```
SUMMARY
  SAFE
DETAILS
  STRONGLY_TYPED_MODEL
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH

PROTOCOL
  proxy_signature.if
GOAL
  %% see the HLPSL specification..
BACKEND
  SATMC
COMMENTS
STATISTICS
  attackFound              false     boolean
  upperBoundReached        true      boolean
  graphLeveledOff          3         steps
  satSolver                zchaff    solver
  maxStepsNumber           11        steps
  stepsNumber              3         steps
  atomsNumber              0         atoms
  clausesNumber            0         clauses
  encodingTime             0.24      seconds
  solvingTime              0         seconds
  if2sateCompilationTime   102.32    seconds
ATTACK TRACE
  %% no attacks have been found..
```

Figure 8 Result of the analysis using SATMC of our scheme.

Table 4 Performance comparison for computational costs between the proposed scheme and Giri et al.'s scheme.

Phase	Giri et al. (2009)	Ours
Delegation	$4t_{exp} + 6t_{mul} + 2t_{add}$	$2t_{exp} + 2t_{mul} + t_{add}$
Proxy signature generation	$2t_{exp} + t_h + t_{mul} + 2t_{add}$	$6\ t_{exp} + t_{poly} + t_{add} + 2t_{mul} + 2t_h$
Proxy signature verification	$3t_{exp} + t_h + 5t_{mul}$	$3t_{exp} + 2t_{mul} + t_h$

Table 5 Functionality comparison between the proposed scheme and Giri et al.'s scheme.

	Giri et al. (2009)	Ours
I_1	√	√
I_2	√	√
I_3	√	√
I_4	√	√
I_5	√	√
I_6	√	√
I_7	×	√

Section 2). Hence, the attacker does not have any ability to recompute $H(M'\|M'_w\|t)$ and as a result, the attacker, who is the original signer, cannot forge a proxy signature.

Consider the case where the attacker is one of the users in the set P of designated signers chosen by the original signer A. Now, other security classes in the set P can also derive the secret key sk_c of the document class SC_c of the hierarchy if those security classes have higher security clearances than that for SC_c. However, the original signer sends the delegation message $\langle r, s, M_w\rangle$ to the proxy signer via a secure channel. As a result, the attacker does not have any ability to recreate a valid proxy signature on behalf of the delegated proxy signer B and he/she cannot forge a proxy signature.

4.3.2. Identifiability

Any verifier can easily determine the relationship of the delegation between the original signer and a proxy signer due to the following reason. In our scheme, in order to verify a proxy signature the verifier requires the verification condition $H(M\|M_w\|t') = H(M\|\ M_w\|t)$, where $t' = r^{M_w} Y_A^r Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p$, $s' = s + H(M\|M_w\|sk_c\|s_i)X_B \pmod q$ and $t = g^{s'} \pmod p$. Recall that the warrant message M_w contains the identities of the original signer and a proxy signer along with an expiration date. Thus, the verifier can determine whether the signature was generated by a proxy signer on behalf of the original signer or not.

4.3.3. Undeniability

In signature generation of our scheme, the proxy signer computes first $s' = s + H(M\|M_w\|sk_c\|s_i)X_B \pmod q$ and then $t = g^{s'} \pmod p$. Thus, s' is computed using the private key X_B and the partial secret key s_i of the proxy signer and hence, t contains the private key X_B and the partial secret key s_i of that proxy signer. Due to usages of X_B and s_i in the proxy signature creation, the proxy signer has no way to deny later about this proxy signature creation by himself/herself on behalf of the original signer.

4.3.4. Verifiability

In the proxy signature generation phase of our scheme, after receiving the tuple (r, s, M_w) from the original signer A, the proxy signer B verifies the condition $g^s = r^{M_w} Y_A^r \pmod p$ using the public key Y_A of A, and other information such as g, s, r, M_w and p. Thus, the proxy signer can verify the delegation power of the original signer, and as a result our proposed scheme is verifiable.

4.3.5. Distinguishability

From our proxy signature verification phase described in Section 3.2.4, we note that the verifier V needs to compute $t' = r^{M_w} Y_A^r Y_B^{H(M\|M_w\|sk_c\|s_i)} \pmod p$ and then to check the verification condition $H(M\|M_w\|t') = H(M\|M_w\|t)$. Here t is generated by the proxy signer as $t = g^{s'} \pmod p$, $s' = s + H(M\|M_w\|sk_c\|s_i)X_B \pmod q$, and $r = g^k \pmod p$ and $s = X_A\ r + k\ M_w \pmod q$ are generated by the original signer. Thus, the verifier can distinguish the signature of the proxy signer from the normal signatures.

4.3.6. Secrecy

Note that during the delegation phase of our scheme, the original signer generates a private key $X_A (1 < X_A < p-1)$ and computes the public key $Y_A = g^{X_A} \pmod p$. After that the original signer selects a random integer $k \in Z_q^*$ and computes the public value $r = g^k \pmod p$. Finally, the original signer computes $s = X_A\ r + k\ M_w \pmod q$ and sends the message $\langle r, s, M_w\rangle$ to the proxy signer via a secure channel. Now, deriving k from $r = g^k \pmod p$ and X_A from $Y_A = g^{X_A} \pmod p$ is computationally infeasible due to the difficulty of solving DLP. Consequently, deriving X_A from $s = X_A r + k\ M_w \pmod q$ becomes a computationally infeasible problem. Thus, the original signer's private key X_A cannot be derived from any public information by an attacker, and as a result the secrecy property is also preserved by our scheme.

5. Simulation results for formal security analysis

In this section, we have implemented our scheme under the AVISPA model checkers for formal security analysis to verify whether there is any attack on our scheme or not. Cryptographic protocols are analyzed by the AVISPA tool and require to be specified in a language called HLPSL (High Level Protocol Specification Language), which is a role based language. In HLPSL, basic roles represent each participant's role, and the composition of roles for representing scenarios of basic roles. AVISPA supports four model checkers, called the back-ends. The On-the-fly Model-Checker (OFMC) is a back-end which is responsible for performing several symbolic techniques to explore the state space in a demand-driven way.

The CL-AtSe (Constraint-Logic-based Attack Searcher) is the second back-end which provides a translation from any security protocol specification written as transition relation in an intermediate format into a set of constraints which are effectively used to check whether there are possible attacks on protocols. Third back-end called the SAT-based Model-Checker (SATMC), and the fourth back-end called TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) approximate the intruder knowledge by using the propositional formula and regular tree languages respectively.

The backends produce the output in the following formats. The first printed section is called the SUMMARY which indicates whether the protocol is safe, unsafe, or whether the analysis is inconclusive. The second section called DETAILS, which explains under what condition the protocol is declared safe, or what conditions have been used for finding an attack, or finally why the analysis was inconclusive. The other sections called PROTOCOL, GOAL and BACKEND are the name of the protocol, the goal of the analysis and the name of the backend used, respectively. After comments and statistics, the trace of the attack (if any) is finally displayed in the usual Alice-Bob notation. More details on AVISPA tool can be found in von Oheimb (2005) and Automated Validation of Internet Security Protocols and Applications (2011).

Fig. 3 shows the specification in HLPSL language for the role of the initiator, the original signer A. A sends the message $\langle r, s, M_w \rangle$ to proxy signer B via a secure channel, who is a user in the security class SC_i in the set P.

Fig. 4 shows the specification in HLPSL language for the role of the proxy signer B. After receiving the message $\langle r, s, M_w \rangle$ from A, it sends the message $\langle M, M_w, H(M \| M_w \| sk_c \| s_i), r, H(M \| M_w \| t) \rangle$ to the verifier or receiver V via a public channel.

In Fig. 5, we have implemented the role of the verifier, V in HLPSL. The verifier receives the message $\langle M, M_w, H(M \| M_w \| sk_c \| s_i), r, H(M \| M_w \| t) \rangle$ from B via a public channel.

We assume that the intruder has knowledge of all public parameters. We have simulated our scheme using the Security Protocol ANimator for AVISPA (SPAN). The results are tested using OFMC, CL-AtSe and SATMC backends. The results of the analysis using OFMC, CL-AtSe and SATMC of our scheme are shown in Figs. 6–8. The summary of simulation results are as follows:

- OFMC reports the protocol is safe.
- CL-AtSe reports the protocol is safe.
- SATMC reports the protocol is safe.

From the detailed results of simulation, it is found that there are no possible passive and active attacks on our scheme.

6. Performance comparison with related schemes

In this section, we compare the performance and security of our scheme with other related schemes. To best of our knowledge, Giri et al.'s proxy signature scheme is only based on the hierarchical access control so far in the literature. For this reason, we thus compare the performance of our scheme with Giri et al.'s scheme only. For comparing the computational costs between Giri et al.'s scheme and our scheme, we have used the same notations described in Table 3. Performance comparison in terms of computational complexity between our scheme and Giri et al.'s scheme is shown in Table 4. Note that Giri et al.'s scheme requires more computational time during the delegation and proxy signature verification phases as compared to those for our scheme. However, our scheme requires more computational complexity during the proxy signature generation phase as compared to that for Giri et al.'s scheme. Overall, our scheme is comparable with Giri et al.'s scheme from the computational complexity point of view.

In Table 5, we have shown the functionality comparison between our scheme and Giri et al.'s scheme. In this table, we have used the following notations. I_1: Whether satisfies unforgeability security requirement or not; I_2: Whether satisfies identifiability security requirement or not; I_3: Whether satisfies undeniability security requirement or not; I_4: Whether satisfies verifiability security requirement or not; I_5: Whether satisfies distinguishability security requirement or not; I_6: Whether satisfies secrecy security requirement or not; I_7: Whether flexible in choosing the proxy signer or not. We note that our scheme is secure against the possible attacks which are demonstrated through the analytical and simulation results discussed in Sections 4.3 and 5, respectively. In addition, our scheme has the flexibility in choosing the proxy signers from a designated set of proxy signers based on their availability and work load, whereas Giri et al.'s scheme does not have that criteria.

7. Conclusion

We have proposed a new proxy signature scheme based on hierarchical access control. In our scheme the documents of a user belonging to a security class in the hierarchy can be signed by a proxy signer who is either that user or any user belonging to some security class in a set of designated proxy signers selected by the original signer. Our scheme satisfies all the security requirements needed by a proxy signature. The proposed scheme is also efficient in terms of computational complexity as compared with the existing related proxy signature schemes. In addition, dynamic access control such as addition of new security classes into the hierarchy and removal of existing security classes from the hierarchy is supported efficiently.

Acknowledgments

The authors would like to thank the anonymous reviewers and the Editor-in-Chief for providing constructive and helpful feedback.

References

Akl, S.G., Taylor, P.D., 1983. Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems (TOCS) 1 (3), 239–248.

Aumasson, J.P., Henzen, L., Meier, W., Plasencia, M.N., 2010. Quark: a lightweight hash. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010), vol. 6225. LNCS, pp. 1–15.

Automated Validation of Internet Security Protocols and Applications. <http://avispa-project.org/> (Accessed on October 2011).

Chung, Y.F., Lee, H.H., Lai, F., Chen, T.S., 2008. Access control in user hierarchy based on elliptic curve cryptosystem. Information Sciences 178 (1), 230–243.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2010. Introduction to Algorithms, third ed. Prentice Hall.

Das, A.K., A secure and effective user authentication and privacy preserving protocol with smart cards for wireless communications. Networking Science. http://dx.doi.org/10.1007/s13119-012-0009-8.

Das, A.K., 2012b. A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. International Journal of Information Security 11 (3), 189–211.

Das, M.L., Saxena, A., Phatak, D.B., 2009. Algorithms and approaches of proxy signatures: a survey. International Journal of Network Security 9 (3), 264–284.

Das, A.K., Paul, N.R., Tripathy, L., 2012. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. Information Sciences 209, 80–92.

Delfs, H., Knebl, H., 2007. Introduction to Cryptography – Principles and Applications. Springer.

Giri, D., Srivastava, P.D., 2007. An asymmetric cryptographic key assignment scheme for access control in tree structural hierarchies. International Journal of Network Security 4 (3), 348–354.

Giri, D., Srivastava, P.D., 2008. A cryptographic key assignment scheme for access control in poset ordered hierarchies with enhanced security. International Journal of Network Security 7 (2), 223–234.

Giri, D., Dalal, J., Srivastava, P.D., Singh, S.R., 2009. HACBPS: a hierarchical access control-based proxy signature. International Journal of Recent Trends in Engineering 2 (1), 222–226.

Lin, C.H., 1997. Dynamic key management schemes for access control in a hierarchy. Computer Communications 20 (15), 1381–1385.

Mambo, M., Usuda, K., Okamoto, E., 1996. Proxy signatures for delegating signing operation. In: Proceedings of the Third ACM Conference on Computer and Communications Security (CCS), pp. 48–57.

Manuel, S., 2011. Classification and generation of disturbance vectors for collision attacks against SHA-1. Designs, Codes and Cryptography 59 (1–3), 247–263.

Odelu, V., Das, A.K., Goswami, A., 2012. A novel key management mechanism for dynamic hierarchical access control based on linear polynomials. In: International Conference on Security in Computer Networks and Distributed Systems (SNDS 2012). Communications in Computer and Information Science Series (CCIS), vol. 335. Springer-Verlag, pp. 1–10.

Odelu, V., Das, A.K., Goswami, A., 2013. LHSC: an effective dynamic key management scheme for linear hierarchical access control. In: Fifth IEEE International Conference on Communication Systems and Networks (COMSNETS 2013).

Odelu, V., Das, A.K., Goswami, A., in press. An effective and secure key-management scheme for hierarchical access control in e-medicine system. Journal of Medical Systems.

Sandhu, R.S., 1988. Cryptographic implementation of a tree hierarchy for access control. Information Processing Letters 27 (2), 95–98.

Secure Hash Standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), US Department of Commerce, April 1995.

Shao, Z., 2003. Proxy signature schemes based on factoring. Information Processing Letters 85 (3), 137–143.

Shen, V.L.R., Chen, T.S., 2002. A novel key management scheme based on discrete logarithms and polynomial interpolations. Computers & Security 21 (2), 164–171.

Stallings, W., 2003. Cryptography and Network Security: Principles and Practices, third ed. Prentice Hall.

Tan, Z., Liu, Z., Tang, C., 2002. Digital proxy signature schemes based on DLP and ECDLP. In: MM Research Preprints, MMRC, AMSS, Academia, Sinica, Beijing, vol. 21, pp. 212–217.

von Oheimb, D., 2005. The high-level protocol specification language HLPSL developed in the EU project AVISPA. In: Proceedings of APPSEM 2005 Workshop.

Wu, J., Wei, R., 2005. An access control scheme for partially ordered set hierarchy with provable security. In: Selected Areas in Cryptography, vol. 3897. LNCS, pp. 221–232.

Zhong, S., 2002. A practical key management scheme for access control in a user hierarchy. Computers & Security 21 (8), 750–759.