

# Attacks and Comments on Several Recently Proposed Key Management Schemes

Niu Liu<sup>1</sup>, Shaohua Tang<sup>1\*</sup>, and Lingling Xu<sup>1</sup>

School of Computer Science & Engineering,  
South China University of Technology, Guangzhou, China  
niuliu83@gmail.com, shtang@IEEE.org, xulingling810710@163.com

**Abstract.** In this paper, we review three problematic key management (KM) schemes recently proposed, including Kayam's scheme for groups with hierarchy [10], Piao's group KM scheme [14], Purushothama's group KM schemes [17]. We point out the problems in each scheme. Kayam's scheme is not secure to collusion attack. Piao's group KM scheme is not secure. The hard problem it bases is not really hard. Purushothama's scheme has a redundant design that costs lots of resources and doesn't give an advantage to the security level and dynamic efficiency of it. We also briefly analyze the underlying reasons why these problem emerge.

**Keywords:** key management schemes; GKM; hierarchical access control; HAC; cryptanalysis; collusion attack; hard problem; secure group communication.

## 1 Introduction

Key management schemes are schemes designed to help a group of users or subgroups share secret keys when they want to communicate or share data with each other securely.

From the aspect of the structure of a group, we have two kinds of key management schemes. Some schemes deal with the secure communication equally shared by all members, the goal is to maintain and update a secret session key, we call them session key distribution schemes or group key management schemes (GKMS), such as Chiou's [3] Wong's [19] and Aparna's [2] schemes. Some other schemes focus on secure contents access and communication of groups with hierarchy, we may call them hierarchical access control schemes (HACS), such schemes include Crampton's [4], Zou's schemes [20] and etc.

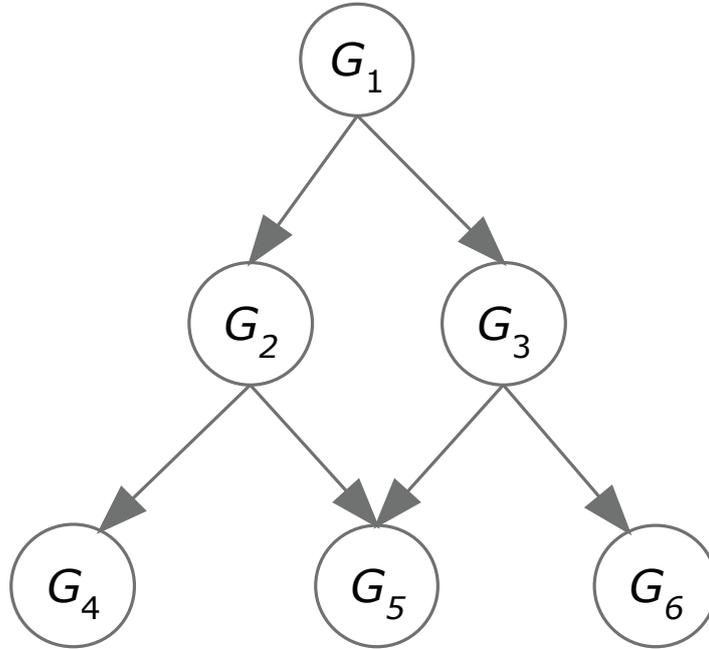
Based on how keys are formed and managed, there're two kinds of schemes. One is centralized. Schemes of this kind need a trusted third party, a group controller GC, to distribute keys and manage the dynamic changes in a group. The other is distributed or contributory schemes whose keys are generated and managed by uniform contribution of all group members. In this paper, we focus on the centralized KM schemes.

---

\* Corresponding Author (email: shtang@IEEE.org)

To design a cryptographic key management scheme is *deceptively* easy. One may propose a scheme that is intuitively secure, and then security flaws emerge later on [8]. Hereby, we reviewed some KM schemes recently proposed and found that there are several different problems in design that researchers may encounter frequently.

*The first problem* happens in HACS, the **collusion attack**. In HACS, users in higher level have access to the lower levels. The relation of the hierarchical structure of a group is usually shown by a directed acyclic graph (DAG) as shown in Fig.1. In this figure,  $G_1$  has access to contents or communication of all other members,  $G_2$  has access to  $G_4$  and  $G_5$ . We also use partial ordering  $G_i \preceq G_j$  to describe that  $G_j$  has access to  $G_i$ . The collusion attack happens when there is a security flaw in design leads to that the collaboration of two or more members gain access to a member they are not supposed to have. We show our attack on Kayem's HACS [10], the sub-poset key management scheme (SPKM).



**Fig. 1.** A DAG example of illustrating hierarchical relation of a group.

*The second problem* lies in the **hard problem** of a KM scheme. When a designer thinks the security of his scheme is guaranteed by a hard problem, he should check at least the following two points: whether the problem is really a hard one; whether it is hard in respect of a constant security parameter, and

not in a variable that can't be defined before implementation. We take Piao's GKMS [14] as our example.

*The third problem* is not in security, but in **efficiency**. When a design that cost a lot of resources, such as storage, computing and etc., but achieves no better security level or dynamic efficiency of its original scheme, we call it a redundant design. The problem may hide itself in the complicated procedure from customers or even designers themselves. It's necessary to make a KM scheme efficient as well as secure. Normally, security and efficiency are in the two opposite sides of the balance when design. It's the best choice to firstly find out and cut down the redundant design considering efficiency issue, since there's no need to compromise for the unnecessary part. We take Purushothama's schemes [17], both GKMS and HACS, to show the problem.

We notice that designing a KM scheme is a sophisticated work that needs both inspired insights and careful rethinking. Our paper presents several problems in different aspects, though not complete, of it. Our aim is to provide cases for reference of further research on designing.

The following parts of this paper are organized as follows. Section 2 presents a table of notations that will be used later on. Section 3 shows the collusion attack problem of Kayem's SPKM [10]. Section 3.1 briefly introduces how the SPKM works; Section 3.2 shows how the collusion attack works, and the problem in design of the SPKM. Section 4 shows the flaws of Piao's scheme [14]. Section 4.1 gives an overview of his scheme; Section 4.2 shows that Piao's scheme has some security flaws assuming that the hard problem of it is OK; Section 4.3 shows that the problem their scheme builds upon is not really hard and the way their scheme uses a hard problem is problematic. Section 5 shows the problem of redundant design of Purushothama's schemes [17]. In Section 5.1 and section 5.2, we briefly review the two similar schemes, wong's LKH scheme [19] and Purushothama's GKMS; In Section 5.3, we compare these two schemes and comment on the design of Purushothama's. We draw conclusion in Section 6.

## 2 Notations

We give Table.1 for the convenient referring of notations.

## 3 Collusion Attack

In 2010, Kayem et al. proposed a sub-poset key management scheme(SPKM) [10, 11]. It is derived from Akl and Taylor scheme [1]. The aim of the SPKM is to improve the efficiency of Akl's scheme by linearly bounding the exponents assigned to classes. Unfortunately, we find that the SPKM is collusion-labile. Kayem uses a principle to avoid collusion attack in the SPKM, but we figure out that the principle in itself is not a sufficient condition of collusion-proof.

**Table 1.** Notations.

Notations	Meaning
GKMS	Group key management schemes.
HACS	Hierarchical access control schemes, also KM schemes for groups with hierarchy.
GC	The group controller, a trusted third party.
$G_i$	Subgroup with ID $i$ in a hierarchical group.
$K_i$	Key to access contents or communication of subgroup $G_i$ in HACS.
$t_i$ or $t_{x,y}$	The exponent of a subgroup $G_i$ or $G_{x,y}$ in the SPKM [10].
$a b$	It stands for that $a$ divides $b$ .
$G_i \preceq G_j$	$G_j$ has access to $G_i$ .
$\{x\}_k$	Symmetric encryption of $x$ by key $k$ .
$U_i$	User with ID $i$ in a GKMS.
GK	The group key, or the session key to secure group communication.
KEK	Key encryption key.
$k_i$	A KEK for a user $u_i$ or a virtual node of key-tree.
$a \bmod b$	The residue class of $a$ modulo $b$ .
$\gcd(a, b)$	The greatest common divisor of $a$ and $b$ .

### 3.1 An overview of the SPKM

In the SPKM, the group controller(GC) randomly and secretly chooses two big primes  $p$  and  $q$ , and a secret base key  $K_0$ . It computes key  $K_i$  according to the following formula, and distributes it to each subgroup  $G_i$  secretly :

$$K_i \equiv K_0^{t_i} \bmod M, \quad (1)$$

where  $M$  is the product of  $p$  and  $q$ .  $M$  is public.  $t_i$  is the *public* exponent of a subgroup  $G_i$ .

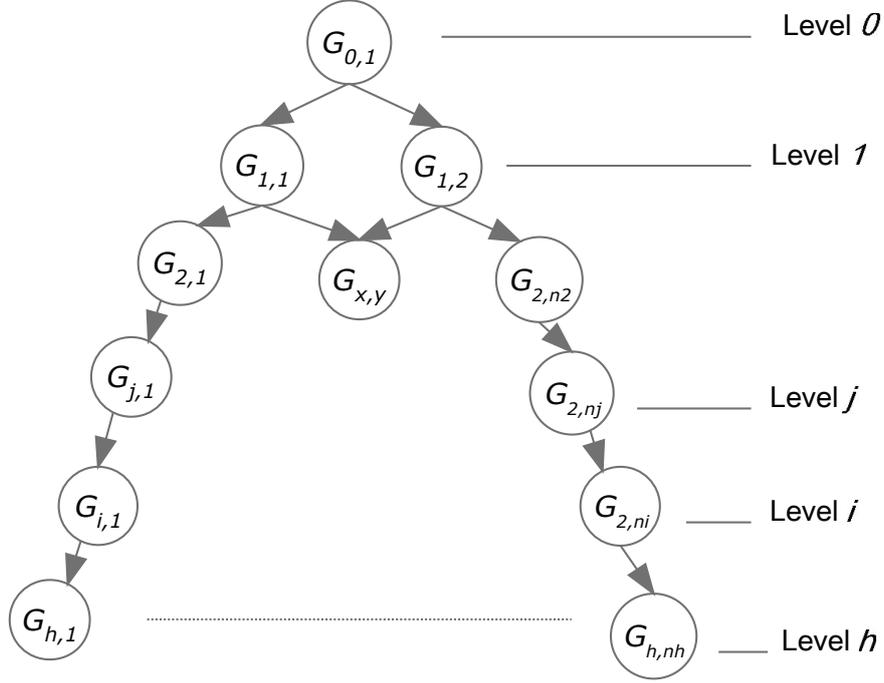
To a tree-like hierarchy as shown in figure 2, the secret key of the root node is  $K_0$ . When  $G_j$  has the access to  $G_i$ , we let  $t_j$  divides  $t_i$ . Then,  $G_j$  is able to derive  $G_i$ 's secret key  $K_i$  from his own key  $K_j$  as follows:

$$K_i \equiv K_0^{t_i} \equiv K_0^{t_j \times \frac{t_i}{t_j}} \equiv K_j^{\frac{t_i}{t_j}} \bmod M. \quad (2)$$

In the Akl and Taylor scheme, to prevent collusion attack, each subgroup  $G_i$  is given a unique prime number  $p_j$ , and the exponent  $t_i$  is assigned as follows:

$$t_i = \prod_{G_j \not\subseteq G_i} p_j. \quad (3)$$

In the SPKM, the GC chooses  $t_i$  in a different way to avoid collusion attack. Subgroups are considered as in different levels as shown in Fig. 2. The scheme only considers the lower-level subgroups' collusion attack on the upper-level subgroups, but not other types such as sibling collusion or etc. To be specific, the SPKM gives two properties of collusion (*Property* 4.1 and 4.2 in his paper [11]). Subgroups of level  $l$  may collude to attack on a higher level subgroup, if one of the following properties satisfies:



**Fig. 2.** An example of key management hierarchy.

1. *Property 4.1:* for some  $x < l$ ,  $\gcd(t_{l,1}, \dots, t_{l,n_l}) = \gcd(t_{x,1}, \dots, t_{x,n_x})$ ;
2. *Property 4.2:* for some exponent  $t_{x,y}$ ,  $\gcd(t_{l,y_1}, t_{l,y_2}) | t_{x,y}$ , where  $x < l$  and  $1 \leq y_1 < y_2 \leq n_l$ ,

where  $\gcd$  is the greatest common divisor,  $n_i$  is the number of subgroups in level  $i$ .

Kayem also gives another principle equivalent to the above, the *Definition 3.3.1* in the Chapter 3 of his monograph [10], that the  $\gcd$  of all  $t_i$  of a lower level  $i$  shall not be a divisor of any  $t_j$  of an upper level subgroup. That is, for a  $t_{x,y}$  where  $x < l$ , if

$$\gcd(t_{l,1}, \dots, t_{l,n_l}) | t_{x,y} \quad (4)$$

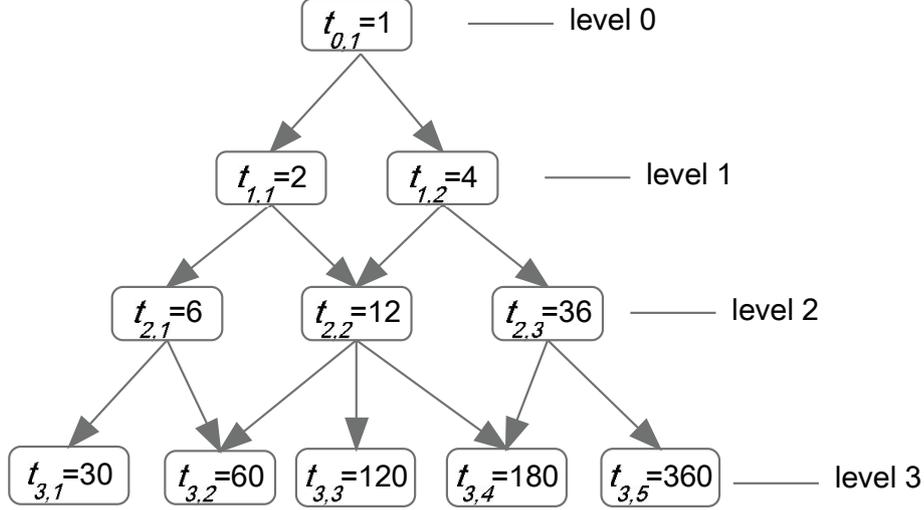
satisfies, then subgroup  $G_{x,y}$  is vulnerable to collusion attack of subgroups in level  $l$ . We denote the  $\gcd$  of the  $l$ -th subgroups' exponents  $\gcd(t_{l,1}, \dots, t_{l,n_l})$  as  $\text{GCD}(l)$  for short in the following parts.

### 3.2 The collusion attack on the SPKM

We give an implemented example of the SPKM as shown in Fig. 3.

We have:  $\text{GCD}(0) = 1$ ,  $\text{GCD}(1) = 2$ ,  $\text{GCD}(2) = 6$ ,  $\text{GCD}(3) = 30$ . We can verify that the above mentioned *Property 4.1*, *4.2* and *Definition 3.3.1* are not

satisfied. It holds that  $\text{GCD}(l) \nmid t_{x,y}$  for all  $x < l$ . It is therefore considered as collision-free by the SPKM. But we find that it's still collision-labile.



**Fig. 3.** A case of the SPKM that vulnerable to collusion attack.

The secret base key is  $K_0$ ,  $M$  is a public modulus which is a product of two randomly chosen big prime numbers. The secret keys of  $G_{1,1}$ ,  $G_{1,2}$ ,  $G_{2,1}$  and  $G_{3,1}$  are  $K_{1,1} = K_0^2$ ,  $K_{1,2} = K_0^4$ ,  $K_{2,1} = K_0^6$  and  $K_{3,1} = K_0^{30} \pmod{M}$ .

Readers may check that in the hierarchical relations of the example in Fig. 3, subgroups  $G_{1,2}$  and  $G_{3,1}$  have no access to  $G_{1,1}$  and  $G_{2,1}$ . But when the former two subgroups collaborate and share their exponents and keys, they can figure out the secret keys of the latter two subgroups.

Because  $\text{gcd}(t_{1,2}, t_{3,1}) = \text{gcd}(4, 30) = 2$ , we have

$$\frac{(K_{1,2})^8}{K_{3,1}} = \frac{(K_0^4)^8}{K_0^{30}} = K_0^{4 \times 8 - 30} = K_0^2 = K_{1,1}. \quad (5)$$

Thus  $G_{1,2}$  and  $G_{3,1}$  can infer the secret key  $K_{1,1}$  of  $G_{1,1}$ . So is  $K_{2,1}$ .

The reason why the SPKM fails lies on it's principle of collusion-proof. As we've said before, the principles of it focus on one level subgroups' collusion but not inter-level collusion. To remedy the collusion-proof principle, we may change (4) to

$$\text{gcd}(t_{x_1,y_1}, t_{x_2,y_2}) \mid t_{x_3,y_3} \quad (6)$$

for any 3 subgroups  $G_{x_1,y_1}$ ,  $G_{x_2,y_2}$ ,  $G_{x_3,y_3}$ , where  $G_{x_1,y_1}$  and  $G_{x_2,y_2}$  have no access to  $G_{x_3,y_3}$ .

Besides there are some design problems in the SPKM. The assigned exponents don't fit the hierarchical relations well. In a same level of an HAC scheme,

one sibling is not supposed to have access of another. But the SPKM violates the rule. For example,  $G_{2,2}$  and  $G_{2,3}$ 's sibling  $G_{2,1}$ , who is supposed not to have access to the them, has it. Thus an implemented SPKM case actually represents another hierarchical relations other than the original one. For example, the actual hierarchical relations of the implemented SPKM case in Fig. 3 can be redrawn as Fig.4.

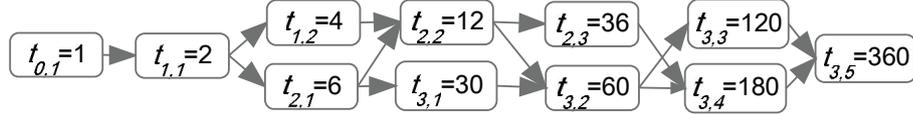


Fig. 4. The actual hierarchical relations of Fig.3.

We can infer from Fig.4 that there are lots of possible ways of collusion attack, such as  $G_{1,2}$  and  $G_{2,1}$ 's attack on  $G_{1,1}$ ,  $G_{2,2}$  and  $G_{3,1}$ 's attack on  $G_{2,1}$ ,  $G_{2,3}$  and  $G_{3,2}$ 's attack on  $G_{2,2}$  and etc.

## 4 The Use of a Hard Problem

Piao et al. proposed a polynomial-based key management scheme [14] in 2012. This is a centralized scheme. It has two parts, one is the intra-group KM scheme, and the other is the inter-group KM scheme. The former one is based on the problem of *univariate polynomial factorization* in a finite field, and the latter uses the pseudo-random number generator (PRNG) and the existing intra-group keys. Piao argues that by the result of Gao [6] and Kipnis [12], the problem of univariate polynomial factorization is NP-hard, so his scheme is secure. But, we find out that it is not the case. The main flaws of Piao et al.'s scheme lie in the intra-group part. It's sufficient for our purpose to only discuss this part. In this paper, we'll show its problems in details.

### 4.1 An overview of Piao et al.'s intra-group key management scheme.

Piao et al.'s intra-group KM scheme aims at distributing session keys in a group secretly. There is a trusted third party, a group controller (GC), to compute, broadcast public information and manage group members' leaving and joining.

For a group of  $n$  users, each user has a public ID  $i$ . Every user  $U_i$  shares a secret Key Encryption Key  $k_i$  with the GC. Then, to set up a session key  $GK$  among all these users in the group, the GC computes the following polynomial  $P$  over a finite field  $F_q$ , and broadcasts it through a public channel,

$$P = \prod_{j=1}^n (x - k_j) + GK. \quad (7)$$

When a user  $U_i$  in the group receives the polynomial  $P$ , he can figure out the session key  $GK$  by computing  $P(k_i) = GK$ .

When a new user joins or an old user leaves the group or etc., the GC randomly picks up a new session key  $GK'$ , recomputes a new polynomial (8) using KEKs of all  $n'$  legitimate users and broadcasts it.

$$P' = \prod_{i=1}^{n'} (x - k_i) + GK'. \quad (8)$$

Then, a legitimate user  $U_j$  is able to get the new session key  $GK'$  by computing  $P'(k_j)$ .

#### 4.2 Attacks on Piao et al.'s intra-group key management scheme.

Actually, Piao et al.'s intra-group KM scheme is a special case of the *double-invariant* schemes, an idea formed in Liu's discussion [13] on some HAC schemes. Schemes with the property of *double-invariance* are vulnerable to attack. Kamal [9] showed how to attack on this vulnerability of Piao et al.'s scheme, though he didn't use this idea.

We briefly introduce the idea. Assuming that a scheme PT at time  $t$  can be expressed in Chinese Remainder Theorem as follows [13],

$$\begin{cases} CRT[t] \equiv a[t]_1 \pmod{M[t]_1}, \\ \dots, \\ CRT[t] \equiv a[t]_i \pmod{M[t]_i}, \\ \dots, \end{cases} \quad (9)$$

where  $i = 1, \dots, n[t]$ .  $M[t]$  is the modulus,  $a[t]$  is the residue class.

If at time  $t'$ , which is after  $t$ , there hold  $a[t']_i = a[t]_i$  and  $M[t']_i = M[t]_i$  simultaneously for some  $i$  (at least one) while computing  $CRT[t']$ , then we call PT a *double-invariant* scheme. For more details, see Definition 2.1 of the paper [13]. All *double-invariant* schemes are vulnerable to attacks.

For an insider attacker  $\mathcal{A}$  who we assume to be  $U_1$  without loss of generality, the polynomial  $P$  and the session key  $GK$  are known to him. We rewrite (7) in a way similar to Chinese Remainder Theorem:

$$\begin{cases} P - GK \equiv 0 \pmod{(x - k_1)}, \\ \dots \\ P - GK \equiv 0 \pmod{(x - k_n)}. \end{cases} \quad (10)$$

After a new user, for example  $U_{n+1}$ , joins the group, (10) turns to:

$$\begin{cases} P' - GK' \equiv 0 \pmod{(x - k_1)}, \\ \dots \\ P' - GK' \equiv 0 \pmod{(x - k_n)}, \\ P' - GK' \equiv 0 \pmod{(x - k_{n+1})}. \end{cases} \quad (11)$$

Here, we may see that  $a[t']_i = a[t]_i = 0$  and  $M[t']_i = M[t]_i = x - k_i$  for  $i = 1, \dots, n$ . Thus Piao et al.'s scheme is *double-invariant*.  $\mathcal{A}$  may simply get

$x - k_{n+1}$  by  $\frac{P' - GK'}{P - GK}$ . Then, no matter whether  $\mathcal{A}(U_1)$  is in the group or not any more, he'll know every session key as long as  $U_{n+1}$  is still in the group.

Kamal's attacks [9] are more detailed than above. But he doesn't show how to prevent these attacks. We gave a general method to remedy this kind of schemes in discussion [13]. We change  $M[t]_i$  in every session by adding a unique session related number  $u$  to it. For example, we can change  $x - k_i$  into  $x - h(k_i, u)$  and broadcast  $u$  in every session to remove the property of *double-invariance*, where  $h$  is a cryptographic hash function, and  $u$  is public and different in each session. Then (7) turns to

$$P = \prod_{i=1}^n (x - h(k_i, u)) + GK. \quad (12)$$

But like other *double-invariance* removed schemes, the security of this improved version lies on the hardness of finding a preimage of the cryptographic hash function and the avoidance of using a used hash value, instead of the problem of finding a root of a univariate polynomial. Actually, this problem is even not hard, we can't build a cryptographic protocol based on it. We discuss it in the following part.

#### 4.3 Comments on the hard problem of Piao et al.'s scheme.

Piao et al.'s intra-group KM scheme is based on the *univariate polynomial factorization* problem. Kipnis points out that over a finite field, a system of multivariate quadratic equations can be transformed to a certain form of super-sparse univariate polynomial in polynomial time (Lemma 3.3 in his article [12]). Since solving the system of multivariate quadratic equations is known to be NP-complete [5], it seems that the univariate polynomial factorization problem is also hard. Thus, one may argue that at least the polynomial factorization problem can be used as a core to design a secure KM scheme. However, this is not the case.

We refer readers to Geddes's book [7]. The following theorem in Chapter 8 gives us the estimated complexity of factoring a univariate polynomial.

*Theorem 8.12.* The big prime Berlekamp algorithm for factoring a polynomial  $a(x)$  of degree  $n$  in the domain  $GF(q)$  has complexity  $O(k \times n^2 \times \log(q) \times \log(k) + 2n^3)$  field operations.  $k$  represents the number of factors of  $a(x)$ , which on average is approximately  $\log(n)$ .

In Piao et al.'s scheme all factors of  $P$  are simple polynomials and square-free, thus  $k = n$ . The complexity is thus  $O(n^3 \times \log(q) \times \log(n))$ . This is polynomial in  $n$  and logarithmic in  $q$ , where  $n$  is the degree of the polynomial  $P$ , and  $q$  is the size of the finite field  $GF(q)$ . From the above theorem we can infer that factoring a normal polynomial is not a hard problem. In this scheme, since an insider attack  $\mathcal{A}$ , who we assume to be  $U_1$ , knows both the polynomial  $P$ , the session key  $GK$  and his own KEK  $k_1$ , it's pretty easy for him to factor the polynomial

$$\frac{P - GK}{x - k_1} = \prod_{j=2}^n (x - k_j) \quad (13)$$

by the big prime Berlekamp algorithm. Then,  $\mathcal{A}$  gets KEKs of other users who share this session with him.

The designers of this scheme have made a mistake in confusing problems of the super-sparse univariate polynomials with problems of normal univariate polynomials. They are different. Combinatorial properties are introduced into certain forms of super-sparse univariate polynomials while constructing them. That's why some problems of them become hard to solve. Plaisted has good discussions [15,16] on NP-hard problems of super-sparse univariate polynomials, in which we may easily be aware of this fact. For example, it is NP-hard to decide whether two sparse univariate polynomials are relatively prime or not, but it's easy to do that to two normal univariate polynomials.

The last flaw of this scheme is that how it make use of a hard problem goes against a fundamental principle of design. In the security analysis part, when Piao et al. discuss the hard problem on which their scheme is based, they consider  $n$  as the input of the problem. This violates the principle of designing a KM scheme.  $n$  is the degree of polynomial  $P$  in (7), and also the number of users that share a certain session. The number of participants in a session may be up to tens of thousands or just two. It has a large range to vary, and keeps changing over sessions. Therefore, we can't treat  $n$  as a security parameter which is determined beforehand, like the size  $q$  of the finite field. Even if the problem is indeed NP-hard in  $n$ , it's not secure when  $n$  is considerably small. To our knowledge, no KM scheme has ever constrained the lower bound of the number of users in the same way for the sake of security. In principle, the number  $n$  of users should not be considered as the input of a hard problem.

## 5 Redundant Design

Purushothama [17] proposed two group key management schemes, group and multi-layer group communication schemes, based on polynomial interpolation in 2012. To our opinion, his secure group communication scheme is very similar to the Logical Key Hierarchy (LKH) tree-based scheme [19]. The difference between Purushothama's scheme and the LKH scheme is that the former has one more step that uses polynomial interpolation to distribute users' keys. But we argue that this step is unnecessary.

### 5.1 An overview of the LKH scheme [19].

The LKH scheme and Purushothama's group scheme share a lot of attributes in common. In this section, we firstly have a brief look at the LKH scheme [19].

In the LKH scheme, there are  $n$  users and a group controller GC. The GC uses a virtual key-tree as shown in Fig. 5 to manage the dynamic changes in the group. There are two kinds of keys. One is Group Key (GK) on the root node, shared by all legitimate users of the group. When users communicate with each other secretly, they encrypt messages with the Group Key. The other one is Key Encryption Key (KEK) that is used for rekeying.

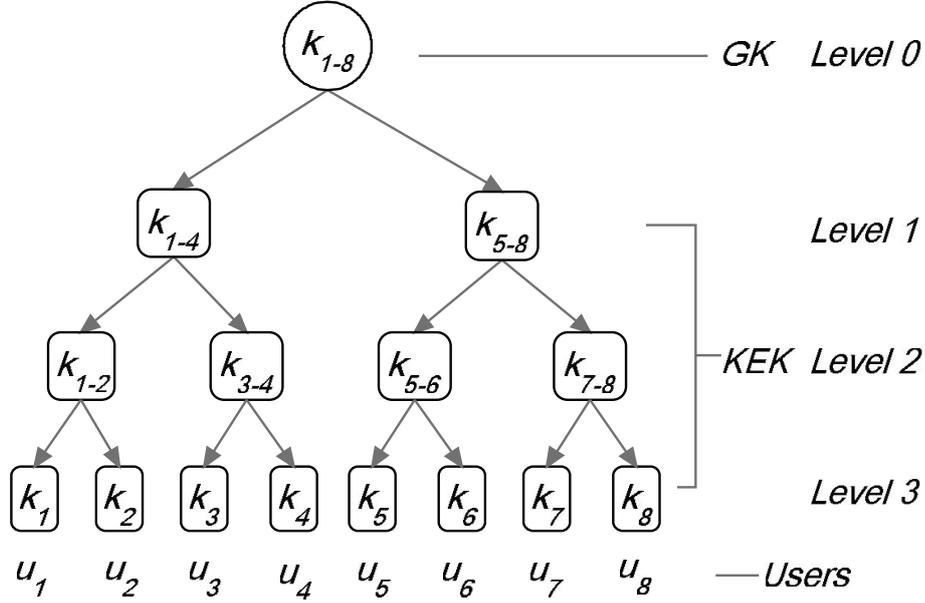


Fig. 5. The key tree of the LKH scheme.

A user  $U_i$  gets  $k_i$  from the GC secretly and directly after he is authenticated. When a user is in the group, he is assigned to a leaf node in the key tree and possesses all those keys from his leaf node to the root node. For example in Fig. 5, after the GC has done all his procedures to set up a Group Key  $k_{1-8}$ ,  $U_8$  shall have these keys:  $k_8$ ,  $k_{7-8}$ ,  $k_{5-8}$  and  $k_{1-8}$ .

When there happens a **dynamic change**, the GK and some of KEKs should be renewed. Every changed key is sent to a user or a group of users encrypted with its descendants' keys corresponding to them. For example, when  $U_8$  leaves the group, the GC should renew keys  $k_{7-8}$ ,  $k_{5-8}$  and  $k_{1-8}$  and delete  $k_8$ . The GC sends three encrypted messages to rekey them as follows,

$$\begin{aligned}
 GC &\rightarrow \{U_1, \dots, U_4\} : \{k'_{1-8}\}_{k_{1-4}}, \\
 GC &\rightarrow \{U_5, U_6\} : \{k'_{1-8}, k'_{5-8}\}_{k_{5-6}}, \\
 GC &\rightarrow U_7 : \{k'_{1-8}, k'_{5-8}, k'_{7-8}\}_{k_7},
 \end{aligned} \tag{14}$$

where  $k'_i$  is the changed keys.

Then all users except  $U_8$  can decrypt these messages to possess renewed keys. The operations for other dynamic events are very similar to the above. We refer readers to wong's paper [19] and Zou's book [21] (LKH: Member-Oriented Rekeying) for more details.

## 5.2 An overview of Purushothama's group scheme [17].

Purushothama's group scheme is similar to the LKH scheme, but there is a slight difference in the distributing step.

In the LKH scheme, the GC sends the encrypted KEKs and GK. In Purushothama's scheme, the GC sends the encrypted *key shares* that are used to compute the KEKs and GK by polynomial interpolations. A share  $S_i$  is a two dimensional pair  $(x_i, y_i)$ , where  $x_i, y_i \in GF(p)$ ,  $x_i \neq 0$  for all  $i$ ,  $x_i \neq x_j$  if  $i \neq j$ ,  $p$  is a large prime of length  $L$ . when a user has all  $m$  shares corresponding to a key  $k$ , he firstly computes a polynomial  $P(x)$  through Lagrange interpolation:

$$P(x) = \sum_{j=1}^m (y_j \prod_{i \neq j} \frac{x-x_i}{x_j-x_i}), \quad (15)$$

then he gets  $k = P(0)$ .

As shown in Fig. 6, the GC also maintain a same key tree as in the LKH scheme. Since the KEKs in the leaf nodes are predistributed to all the users as in the LKH scheme, they don't have to be computed through interpolations. There are two kinds of shares. One is key share, different in each key. The other is base share, different in each level. For example,  $k_{1-2}$  has a key share  $S_{1-2}$  and a set  $F_1$  of base shares as its corresponding shares, where  $F_1 = \{(x_1, y_1), \dots, (x_7, y_7)\}$  is the set of base shares related to level 2. Similarly,  $k_{5-8}$  has a key share  $S_{5-8}$  and a set  $F_2$  of base shares as its corresponding shares, where  $F_2 = \{(x_8, y_8), \dots, (x_{15}, y_{15})\}$  is the set of base shares related to level 1.  $k_{1-8}$  is the group key (GK), its corresponding shares include  $S_{1-8}$  and all base shares, ie, shares in set  $F_1, F_2$  and  $F_3$ , where  $F_3 = \{(x_{16}, y_{16}), \dots, (x_{21}, y_{21})\}$ .

When the distribution phase ends, user  $U_i$  should hold the GK, all the KEKs from his leaf node to the root and their corresponding key shares and all the base shares. For example,  $U_8$  should hold  $k_8, k_{7-8}, S_{7-8}, k_{5-8}, S_{5-8}, k_{1-8}, S_{1-8}, F_1, F_2$  and  $F_3$ .

When a **dynamic change** happens in the scheme, the GK, some KEKs and their key shares should be changed. The GC randomly picks new key shares for those keys that shall be renewed, computes corresponding polynomials through (15) and then the keys. In the LKH scheme, the GC sends those changed keys encrypted with their descendants' keys; In Purushothama's scheme, the GC sends those changed key shares encrypted with their descendants' keys.

For example, when the user  $U_8$  leaves the group, the GC should delete  $k_8$ , then renew key shares  $S_{7-8}, S_{5-8}, S_{1-8}$  and also keys  $k_{7-8}, k_{5-8}, k_{1-8}$ . It leaves all base shares unchanged. The GC sends to users the following messages to renew the shares:

$$\begin{aligned} GC \rightarrow \{U_1, \dots, U_4\} &: \{S'_{1-8}\}_{k_{1-4}}, \\ GC \rightarrow \{U_5, U_6\} &: \{S'_{1-8}, S'_{5-8}\}_{k_{5-6}}, \\ GC \rightarrow U_7 &: \{S'_{1-8}, S'_{5-8}, S'_{7-8}\}_{k_7}, \end{aligned} \quad (16)$$

where  $k'_i$  and  $S'_i$  are the changed key and changed key share respectively. Other dynamic changes work in a similar way.

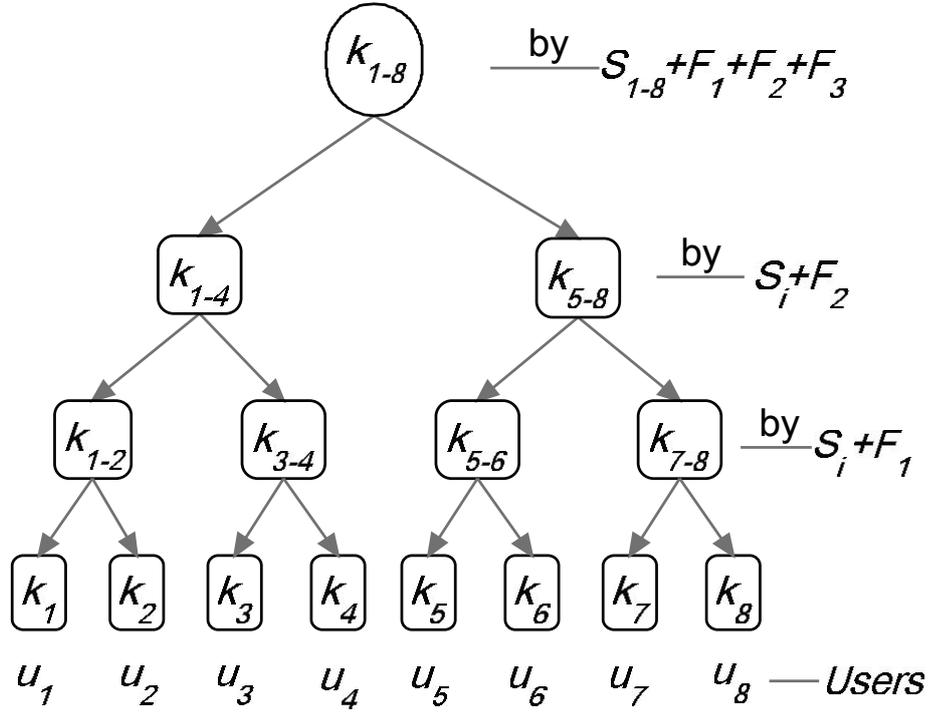


Fig. 6. The key tree of Purushothama’s group scheme.

5.3 Comparisons and comments.

The difference between the LKH scheme and Purushothama’s scheme is that there are one more step to compute the keys. In the former scheme, users decrypt messages and get renewed keys; but at the latter one, users should firstly decrypt messages, then compute a polynomial  $P(x)$  using (15) and finally get the key  $k = P(0)$ . We believe that this step does not improve the *security* of the LKH scheme, nor the *dynamic efficiency* of it.

**Security.** The security of the LKH scheme lies on the encryption of messages. On the other hand, the security of Purushothama’s scheme seems to lie on the encryption and the polynomial interpolation. But we notice a fact that whenever a dynamic change happens, all base shares shared by all users are not changed. When any user is removed from the group, he still knows all the base shares the system is using. Considering that the leaving of a user is almost inevitable, the base shares are not secret to adversaries. Thus the key share  $S_i$  is information theoretically equivalent to the corresponding key  $k_i$ . The security of Purushothama’s scheme actually *only* lies on the encryption of messages sent

by the GC. Anyone who has an advantage on guessing the keys of LKH scheme, has a same advantage on that of Purushothama's scheme.

**Dynamic Efficiency.** We compare (14) and (16), and conclude that the two schemes send out same number of messages when revoke a user. It holds true for all other dynamic changes. So the added step won't improve the dynamic efficiency.

We therefore conclude that the interpolation step of Purushothama's group scheme is a redundant design. It demands more resources of computing, storage and communication, but achieves exactly the same security level with the LKH scheme [19]. Polynomial interpolation in cryptography is mostly seen as a tool to share secret information in members by broadcasting public messages. It's a special form of Chinese Remainder Theorem [13]. The reason why the interpolation step of this scheme is redundant probably lies on that this function is misused.

The Purushothama's multi-layer group scheme holds a same problem. It's a redundant design compared to independent linear hierarchical key scheme, in which every subgroup holds all keys of the subgroups he has access to. We'd like to refer readers to Hassen's research paper [18]. His research focuses on the key management for linear hierarchies. He proposed a hybrid scheme that blend the dependent and independent schemes together. This scheme achieves a good dynamic efficiency. In our opinion, when we study *level-based* hierarchical access control problem in cryptographic way, it's better to adopt the linear model other than the tree-like model. It helps to simplify the problem. The design problem of Kayam's scheme [10] may also have something to do with it.

## 6 Conclusion

In this paper, we showed problems of three schemes proposed in recent years. Each one of them has different flaws, including vulnerability to collusion attack, faults on the hard problem, redundancy in design and etc. Designers of key management schemes are prone to those problems even they are careful. The KM schemes are usually very flexible and complex, thus intuition has an important influence on designing. It's also hard for designers to provide provable security for schemes with new hard problems or complex structures. These two reasons make the designing of the KM schemes a very fallible task to which we should pay more attention.

## References

1. Selim G. Akl and Peter D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983. 357372.

2. R. Aparna and B. B. Amberker. A key management scheme for secure group communication using binomial key trees. *International Journal of Network Management*, 20(6):383–418, 2010.
3. G. Chiou and W. T. Chen. Secure broadcasting using the secure lock. *IEEE Transactions on Software Engineering*, 15(8):929–934, 1989.
4. Jason Crampton. Cryptographically-enforced hierarchical access control with multiple keys. *Journal of Logic and Algebraic Programming*, 78(8):690–700, 2009. doi: 10.1016/j.jlap.2009.04.001.
5. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1(1):15–30, 1979.
6. S. Gao, M. van Hoeij, E. Kaltofen, and V. Shoup. The computational complexity of polynomial factorization. *American Institute of Mathematics*, page 364, 2006.
7. K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Springer, 1992.
8. S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography*. 2008. URL: <http://is.gd/xSuaE8>. p210.
9. A. A. Kamal. Cryptanalysis of a polynomial-based key management scheme for secure group communication. *International Journal of Network Security*, 15(1):59–61, 2013.
10. Anne V. D. M. Kayem, Selim G. Akl, and Patrick Martin. *Adaptive Cryptographic Access Control*, volume 48 of *Advances in Information Security*. Springer US, 2010.
11. Anne V. D. M. Kayem, Patrick Martin, and Selim G. Akl. Enhancing identity trust in cryptographic key management systems for dynamic environments. *Security and Communication Networks*, 4(1):79–94, 2010.
12. A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Advances in cryptology CRYPTO99*, pages 788–788. Springer, 1999.
13. Niu Liu, Shaohua Tang, and Lingling Xu. Yet another attack on the Chinese Remainder Theorem based hierarchical access control scheme. *Cryptology ePrint Archive*, Report 2013/132, 2013. <http://eprint.iacr.org/2013/132>.
14. Y. Piao, J. U. Kim, U. Tariq, and M. Hong. Polynomial-based key management for secure intra-group and inter-group communication. *Computers & Mathematics with Applications (2012)*, 2012.
15. D. A. Plaisted. New NP-hard and NP-complete polynomial and integer divisibility problems. *Theoretical Computer Science*, 31(1):125–138, 1984.
16. David Plaisted. Some polynomial and integer divisibility problems are np-hard. *SIAM Journal on Computing*, 7(4):458–464, 1978.
17. B. R. Purushothama and B. B. Amberker. Secure group and multi-layer group communication schemes based on polynomial interpolation. *Security and Communication Networks*, 2012.
18. H. Ragab Hassen, H. Bettahar, A. Bouadbdallah, and Y. Challal. An efficient key management scheme for content access control for linear hierarchies. *Computer Networks*, 2012.
19. C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM Computer Communication Review*, volume 28, pages 68–79. ACM, 1998. 4.
20. X. Zou, Y. Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. In *The 27th IEEE Conference on Computer Communications (INFOCOM 2008)*, pages 538–546, 2008.
21. X. Zou, B. Ramamurthy, and S. S. Magliveras. *Secure group communications over data networks*. Springer Science+Business Media Inc., 2005.