Yet Another Attack On the Chinese Remainder Theorem Based Hierarchical Access Control Scheme

Niu Liu[1], Shaohua Tang[1⋆], and Lingling Xu[1]

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

Abstract. The hierarchical access control scheme based on Chinese Reminder Theorem [49] (CRTHACS) was supposed to be capable of hiding hierarchical structure, but Geiselmann et al. [18] showed practical attacks on CRTHACS to reveal the hierarchies it hides. Then, Zou et al. modified it, and gave a new CRTHACS [50] to resist those attacks. Nevertheless, we find that the modified version is still defective if it permits changes of structure, i.e. the scheme works in a dynamic scenario. In this paper, we describe our attack on the modified version of CRTHACS. We extend the description of the CRTHACS in a more proper form making it easier for us to look into the problem it has. We find the key character of the vulnerability which we name as *double-invariance*. We generalize our attack in an algebraic form and apply it to a series of hierarchical cryptographic access control schemes that share the same vulnerability with CRTHACS. We also give the countermeasure to fix this vulnerability.

Keywords: communication security, CRTHACS, Chinese remainder theorem, hierarchical access control, secure group communication, formal security

1 Introduction

The hierarchical access control (HAC) refers to a scenario in which the users or groups of members of a computer (or communication) system are divided into a number of disjoint security classes. One class may have more privilege than another, and thus it gives rise to a hierarchical structure of the group. Usually, we describe those relations by *partial orders* or a *directed acyclic graph* (DAG).

Since users or members in a same security class form a set, we call it a subgroup. In an HAC scheme, if subgroup G_i has access to G_j's messages or data encrypted by his key, we say there is a partial order $G_j \preceq G_i$. A partial ordering $\preceq$ on a set of subgroups $G = \{G_1, \cdots, G_n\}$ is a binary relation on $S \times S$ that for all G_i, G_j and G_k in G, we have:

1. *Reflexive*: $G_i \preceq G_i$;

[⋆] Corresponding Author (email: shtang@IEEE.org)

2. *Transitive*: if $G_i \preceq G_j$ and $G_j \preceq G_k$ then $G_i \preceq G_k$
3. *Antisymmetric*: if $G_i \preceq G_j$ and $G_j \preceq G_i$ then $G_i = G_j$

If $G_j \preceq G_i$ holds, we may say G_i is an ancestor of G_j, or equally, G_j is a descendant of G_i. To describe the access hierarchy with a DAG, we can draw a directed line from G_i to G_j for every partial order $G_j \preceq G_i$. For simplicity, we may omit those lines that reflect an indirect partial order, i.e. if $G_3 \preceq G_1$ but there is a subgroup G_2 such that $G_3 \preceq G_2$ and $G_2 \preceq G_1$, then we can omit the line from G_1 to G_3 as shown in Fig. 1.

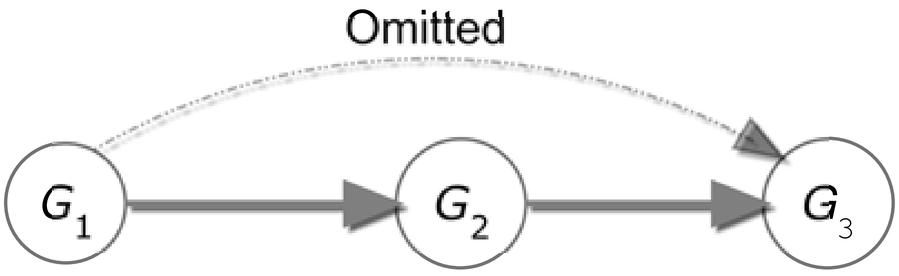

Fig. 1. Omit the line of the indirect partial order $G_3 \preceq G_1$.

Generally, HAC schemes focus on two kinds of structures: general structures as shown in Fig. 2, and as special cases of them, the tree-like structures as shown in Fig. 3. Mostly, there is a trusted third party as the group controller who controls the hierarchical dynamic changes and may help to compute and distribute subgroup keys.

The first cryptographic solution to HAC problems was proposed by Akl et al. [1] Their scheme uses discrete logarithm to help ancestral subgroup derive descendants' data encryption keys by using public information and his private knowledge. Afterwards, a large number of cryptographic HAC schemes have been proposed. Some of them use one way function to derive keys, in which keys in lower level depend on keys in higher level, such as Sandhu's [33], Chick's [12], Lin's [32], Yeh's [47], Hwang's [24], Wu's [43], Kayem's [30], Atallah's scheme [3] and etc. Some other HAC schemes use different ways to derive keys so as to make keys independent from each other and lower the cost in the dynamic changes. Such schemes include Liaw's [31], Zou's [48,49], Wu's [44], Shen's [36], Das's [15], Tzeng's scheme [40] and etc.

Goldwassor et al. once described the status quo of designing key distribution schemes: '*The problem is* deceptively simple. *It is easy to propose protocols in which subtle security problems later emerge.*' [19] So are HAC schemes which form a subclass of key distribution schemes. Many schemes mentioned above suffered various attacks: attack [22] on Liaw's [31], attack [18] on Zou's [49], attack [23] on Yeh's [47], attack [20,41] on Shen's [36] and Wu's [44], attack [42]

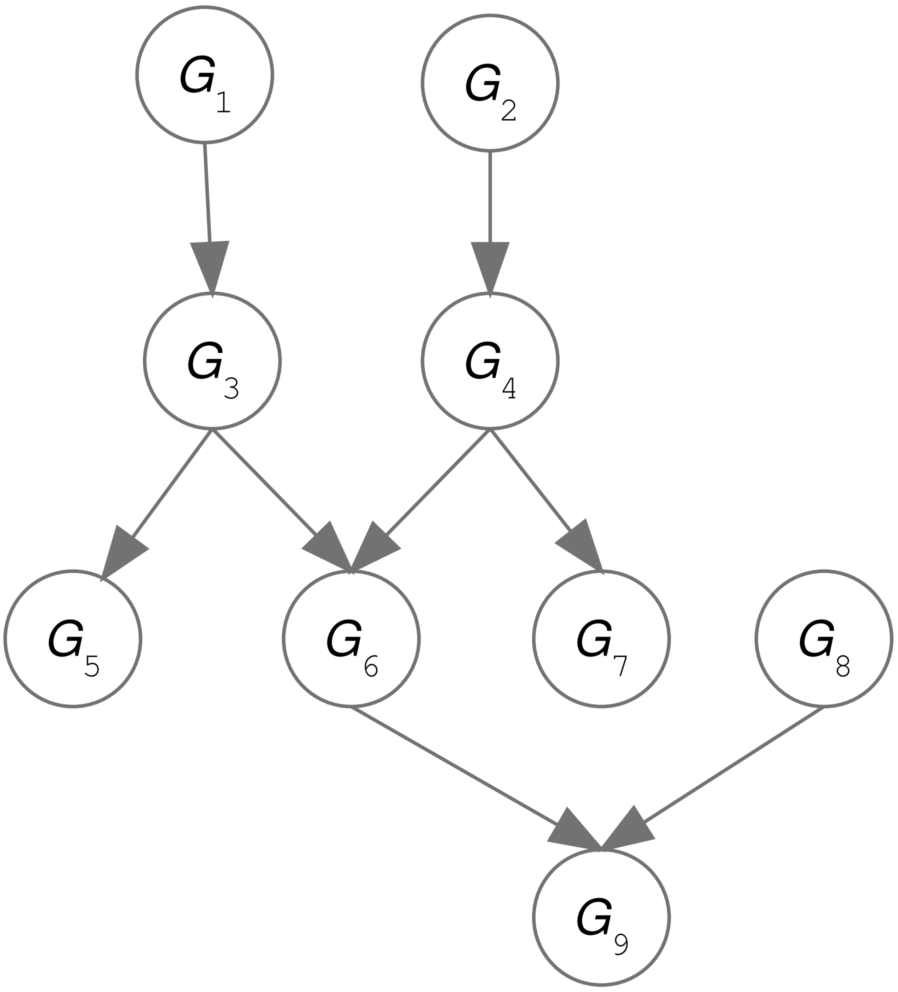

Fig. 2. A case of general hierarchy.

on Hwang's scheme [24] and etc. And some of them still have security flaws after amendment.

In this paper, we find a concrete attack on the improved scheme [50] of Zou's hierarchical access control scheme based on Chinese Reminder Theorem(CRTHACS) [49]. Though Zou's improved version is not the most efficient HAC scheme in the classification of Crampton [14], we pay attention to it and feel worthy and interesting to study it mainly because of the following special reasons:

1. To our knowledge, it's the sole HAC scheme that claims to be able to hide the hierarchical structure and the receivers in communication. In this case,

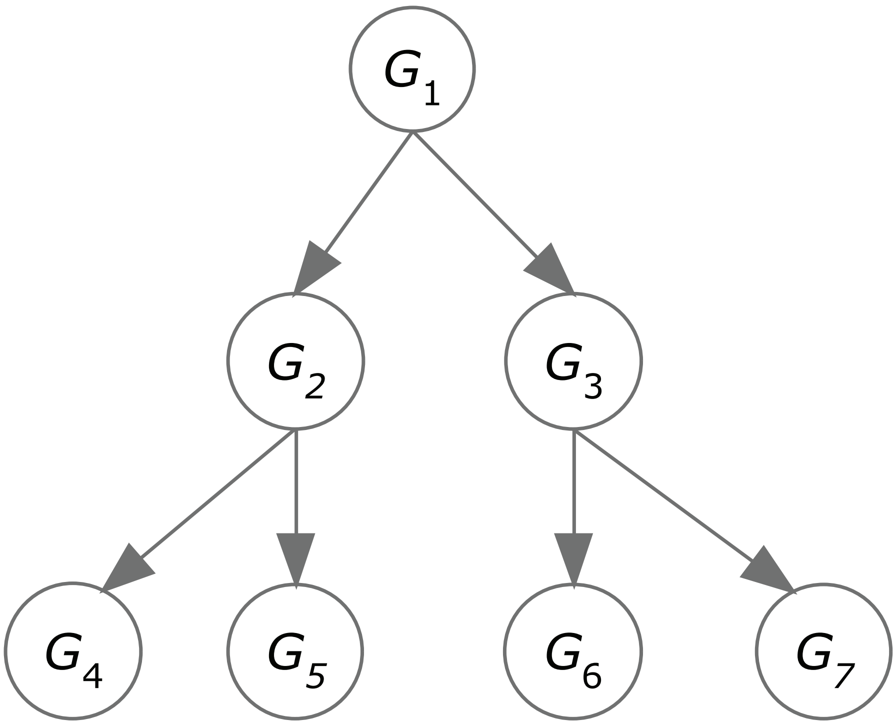

Fig. 3. A case of tree-like hierarchy.

it sets an example for researchers and engineers who are looking for such properties. Thus, it's worthy of a better reexamination.

2. It's based on the Chinese Reminder Theorem, which can be deemed as the general form of many other methods. Analysis on CRTHACS is also critical to other HAC schemes which make use of them. Details see Section 3.
3. While some analyses [25, 46] on group-oriented protocols pay explicit attention to *"inter-session"* analysis in their formation of security notions, CRTHACS doesn't show much concern about this idea. It may result in security flaws even in a "formally proved" scheme [13,17] when designers ignore it unconsciously. Our analysis may present a substantive example of showing how this kind of neglect imperils a protocol. This *"inter-session"* idea has a not very short history. The pioneering work of Bellare etc. [4, 5] formalized the notion of "sessions" of key establishment schemes and the security notion of "session key" in which the consideration of "inter-session" analysis is contained. In this security notion, it calls for a reductive proof on the independence of different sessions. Then lots of researches [3, 7, 11, 25, 29, 37, 46] followed this idea to give the formal definition of different types of group key management protocols. In Atallah's full length paper [3], he discussed the

security notion of indistinguishability and incomputability of HAC schemes under this framework. It's seems inevitable to dive into the inter-session relations if we are to seriously examine the security of a group key management scheme. When designers ignore it unconsciously, security flaws could emerge even in a "formally proved" scheme [13, 17]. However, CRTHACS doesn't show much concern about this idea. Our analysis may present a substantive example of showing how this kind of neglect imperils a protocol.

These points guided us in our research. In our study, we find that the scheme does indeed not fulfil the task of hiding the hierarchical structure. We also find that a series of HAC schemes share the similar vulnerability with CRTHACS. One [40] of them was even proposed in 2011, which showed an ignorance of the problem. Thus, we feel necessary to study the problem in depth.

The rest of this paper is organized as follows. In Section 2, we give our attack on CRTHACS. In Section 2.1, we briefly show how CRTHACS works; In Section 2.2, we give an exemplary attack on a concrete case of CRTHACS; In Section 2.2, we give a toy example; In Section 2.3, we generalize our attack in the form of algebra, and introduce time into the description of schemes; In Section 2.4, we show how to fix a scheme that is vulnerable to our attack. In Section 3, we apply our attack to some other HAC schemes and briefly describe our attack on one [15] of them. Section 4 concludes our paper.

2 Our Attack on CRTHACS

Zou et al. [49] proposed a Chinese Remainder Theory (CRT) based Hierarchical Access Control Scheme (CRTHACS) aiming to hide the access hierarchy. Nevertheless, Geiselmann et al. [18] presented three practical attacks based on the greatest common divisor (GCD) that allowed for revealing at least parts of the hierarchy. Then, Zou et al. [50, 51] modified the scheme to resist those GCD attacks and maintain the property of hiding the hierarchy. Unfortunately, here we find another attack in dynamic scenarios to reveal the hierarchy in their modified CRTHACS.

Some notations are given in Table 1 for reference.

2.1 Review of CRTHACS

CRTHACS [51] is applicable to general hierarchies. There is a trusted party as a group controller (GC). The entire group is divided into n disjoint subgroups $G_1, \cdots, G_n$. Users in these subgroups are managed by a subgroup controller, which is also denoted by $G_1, \cdots, G_n$. Any proper group key management protocol can be used in a subgroup. The η ancestors of subgroup G_i are denoted by $G_{i_1}, \cdots, G_{i_\eta}$, and the set of them is denoted by AG_i.

Initialization: The GC has a pair $(P_{\text{GC}}, S_{\text{GC}})$ of public and private keys. Every subgroup G_i has a pair (P_i, S_i) of public and private keys respectively, and every member U_l in subgroup G_i has a pair (p_l, s_l) of public and private keys. The GC chooses $n+1$ pair-wise relatively prime numbers $N_0, N_1, \cdots, N_n$, then makes N_0 public and keeps N_i in secret. G_i randomly chooses his own data encryption key k_i and sends $E_{P_{\mathrm{GC}}}(E_{S_i}(k_i))$ to the GC.

The GC decrypts all data encryption keys. For subgroup G_i, the GC determines all of his η ancestors $G_{i_1}, \cdots, G_{i_\eta}$. Then the GC establishes the following system of $\eta+1$ congruences and gets the solution of CRT_i:

$$\begin{cases} CRT_i \equiv \{k_i\}_{k_{i_1}} \bmod N_{i_1}, \\ CRT_i \equiv \{k_i\}_{k_{i_2}} \bmod N_{i_2}, \\ \cdots \\ CRT_i \equiv \{k_i\}_{k_{i_\eta}} \bmod N_{i_\eta}, \\ CRT_i \equiv \{k_i\}_{k_i} \bmod N_0. \end{cases} \tag{1}$$

Then, the GC sends $E_{P_i}(E_{S_{\mathrm{GC}}}(N_i, CRT_i))$ to subgroup controller G_i. The subgroup controller decrypts them and sends them along with k_i to users in the subgroup securely. Thus, subgroup Controller G_i holds $(P_i, S_i, k_i, N_i, CRT_i)$, and member l in subgroup G_i holds $(p_l, s_l, k_i, N_i, CRT_i)$.

Data Communication: When a user $U_l \in G_i$, with identity ID_l, wants to send a message M, he sends $(\mathrm{ID}_l, CRT_i, Signed\_MAC, \{M\}_{k_i})$, where $Signed\_MAC = E_{s_l}(MAC_{k_i}(\{M\}_{k_i}))$. Any user $U \in G_{i_j}$, where G_{i_j} is an ancestor of G_i, is able to figure out k_i in congruence $CRT_i \equiv \{k_i\}_{k_{i_j}} \bmod N_{i_j}$, and then the message M. See Fig. 4 as an example. U_{72} in group G_7 is sending a message M to users that have access to his level, i. e. users in group G_2, G_4 and G_7.

We notice that, when a user U_l in subgroup G_i sends a message M encrypted with k_i, he actually publishes the CRT_i of his own subgroup by sending $(\mathrm{ID}_l, CRT_i, Signed\_MAC, \{M\}_{k_i})$.

Dynamic Key Management: There are two levels of dynamics in CRTHACS, one is *member dynamics* such as members' joining, leaving, etc. in a subgroup. The other one is *subgroup dynamics* such as a subgroup's leaving or joining in the entire group. Member dynamic changes are dependent on the subgroup key management protocol, which are not considered here.

Subgroup dynamic changes include: adding, inserting, deleting, splitting a subgroup and modifying the key of a subgroup etc. For example, when adding a new subgroup G_i into the group, the GC determines its ancestors. Then, after G_i sends its secret data encryption key k_i to GC securely, GC computes CRT_i according to (1) and sends $E_{P_i}(E_{S_{\mathrm{GC}}}(N_i, CRT_i))$ to G_i. If G_i has any descendant G_j, GC shall recompute CRT_j and send it to G_j securely.

2.2 A Heuristic Example of Attack

Our attack on CRTHACS is in a dynamic scenario. After several dynamic operations, an outsider attacker, who is even not in any security class, is probably able to find out at least part of the hierarchy which, according to the claim of the scheme, was supposed to be invisible to him. We give a heuristic example of attack as follows. We take the hierarchy in Fig. 2 as an example.

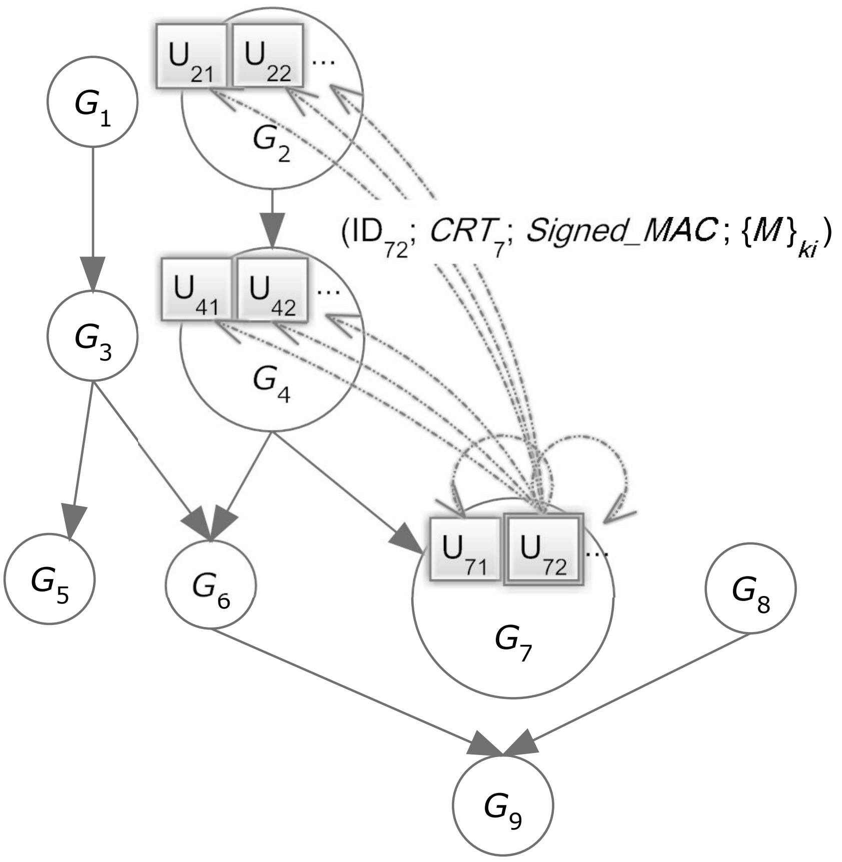

Fig. 4. U_{72} is broadcasting a message to authorized users.

We assume that the adversary $\mathcal{A}$ is able to wiretap all communications in the group, but unable to decrypt or sign a message with a key that he does not possess.

Since users in a same subgroup G_i always share a common CRT_i no matter whether it changes, $\mathcal{A}$ can roughly distinguish subgroups by their CRT_i. Despite the fact that a user may move from one subgroup to another, $\mathcal{A}$ can identify the new group that the user belongs to with his new CRT_i later on. $\mathcal{A}$ sees as much number of subgroups as the number of CRT_is used in the communications of the group.

When the protocol is initiated, the GC computes CRT_9 according to (2), and sends $E_{P_9}(E_{S_{\mathrm{GC}}}(N_9, CRT_9))$ to G_9. When a user U_l in G_9 wants to send a message M to others, he sends $(\mathrm{ID}_l, CRT_9, Signed\_MAC, \{M\}_{k_9})$ to them, the CRT_9 is thereby made public to any adversary who has been listening to their public communications. Consequently, $\mathcal{A}$ knows that the user U_l belongs

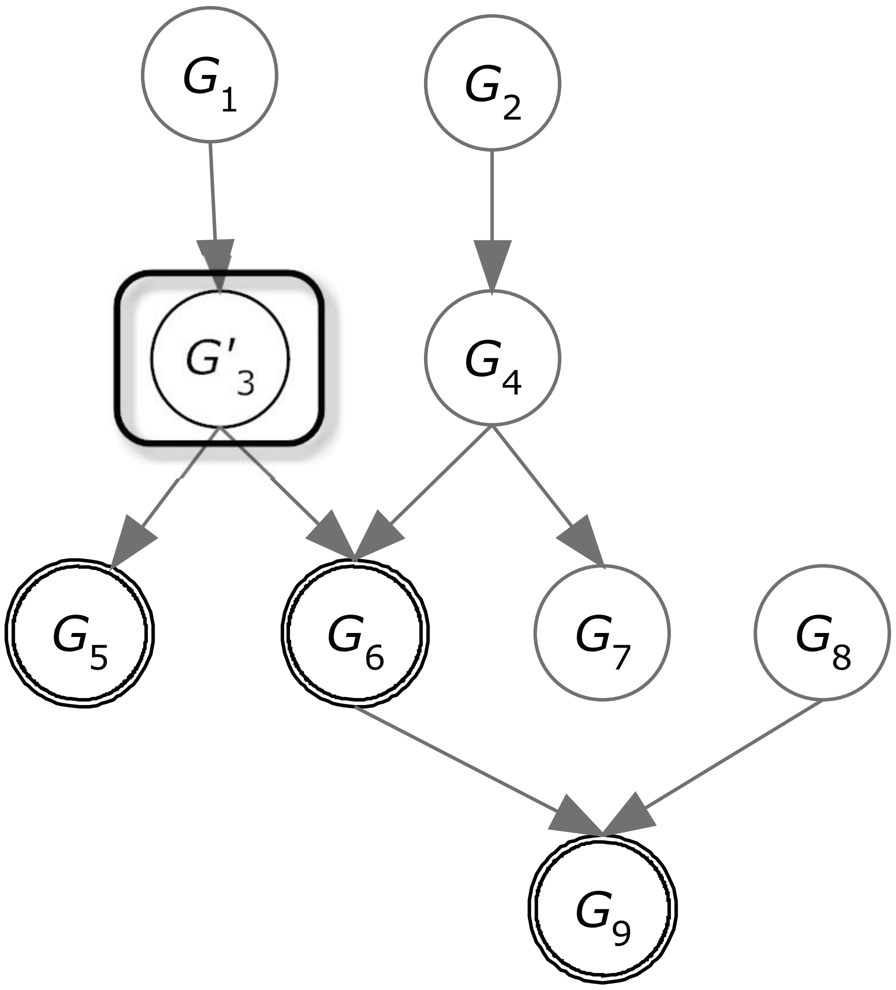

Fig. 5. Modify the data encryption key k_3 of G_3.

to a subgroup corresponding to CRT_9. Similarly, if a user $U_v \in G_i$ sends out a message M' to others, he sends out $(\mathrm{ID}_v, CRT_i, Signed\_MAC, \{M'\}_{k_i})$, then $\mathcal{A}$ knows that U_v's subgroup owns CRT_i.

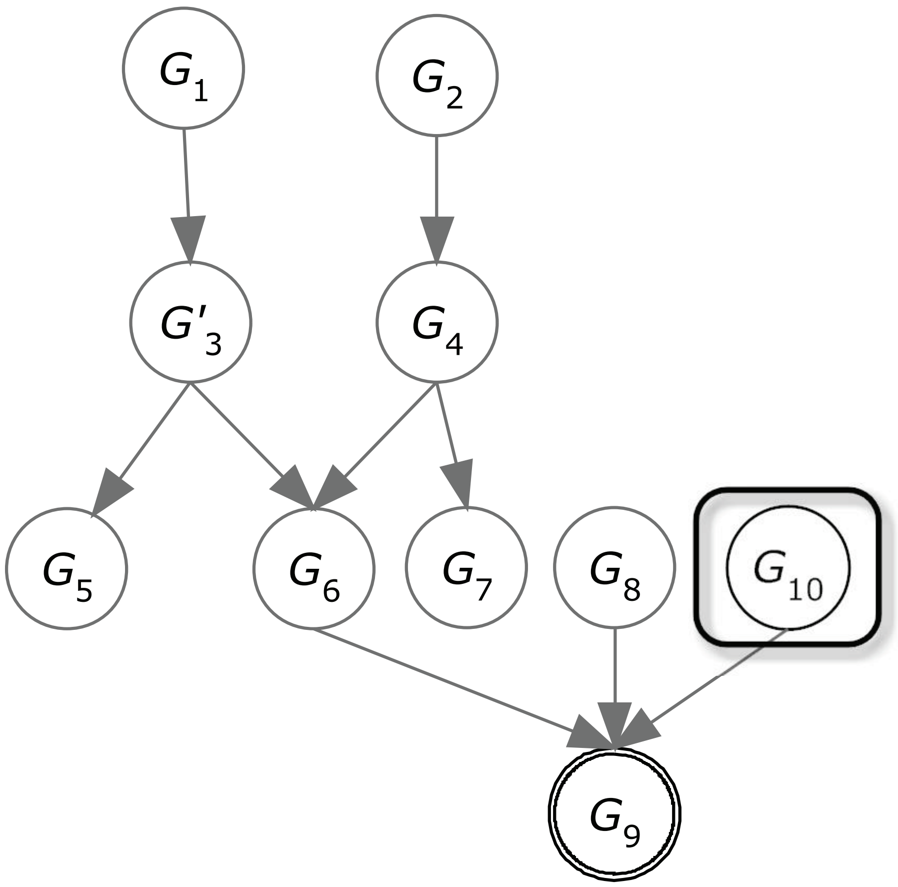

Fig. 6. Add a new subgroup G_{10}.

$$\begin{cases} CRT_9 \equiv \{k_9\}_{k_1} \bmod N_1, \\ CRT_9 \equiv \{k_9\}_{k_2} \bmod N_2, \\ CRT_9 \equiv \{k_9\}_{k_3} \bmod N_3, \\ CRT_9 \equiv \{k_9\}_{k_4} \bmod N_4, \\ CRT_9 \equiv \{k_9\}_{k_6} \bmod N_6, \\ CRT_9 \equiv \{k_9\}_{k_8} \bmod N_8, \\ CRT_9 \equiv \{k_9\}_{k_9} \bmod N_0. \end{cases} \tag{2}$$

Afterwards, we assume that G_3 modifies its data encryption key k_3 to k'_3, as shown in Fig. 5. Then, G_3, G_5,G_6 and G_9 have to update their CRT_i. The GC has to recompute CRT_9 using (3) by substituting k_3 in (2) with k'_3 since G_9 is a descendant of G_3.

$$\begin{cases} CRT'_9 \equiv \{k_9\}_{k_1} \bmod N_1, \\ CRT'_9 \equiv \{k_9\}_{k_2} \bmod N_2, \\ CRT'_9 \equiv \{k_9\}_{k'_3} \bmod N_3, \\ CRT'_9 \equiv \{k_9\}_{k_4} \bmod N_4, \\ CRT'_9 \equiv \{k_9\}_{k_6} \bmod N_6, \\ CRT'_9 \equiv \{k_9\}_{k_8} \bmod N_8, \\ CRT'_9 \equiv \{k_9\}_{k_9} \bmod N_0. \end{cases} \tag{3}$$

Comparing (2) and (3), we get (4) which shows that N_0, N_1, N_2, N_4, N_6 and N_8 are factors of $(CRT'_9 - CRT_9)$, i.e. $N_0N_1N_2N_4N_6N_8|(CRT'_9 - CRT_9)$.

$$\begin{cases} CRT'_9 - CRT_9 \equiv 0 \bmod N_1, \\ CRT'_9 - CRT_9 \equiv 0 \bmod N_2, \\ CRT'_9 - CRT_9 \equiv 0 \bmod N_4, \\ CRT'_9 - CRT_9 \equiv 0 \bmod N_6, \\ CRT'_9 - CRT_9 \equiv 0 \bmod N_8, \\ CRT'_9 - CRT_9 \equiv 0 \bmod N_0. \end{cases} \tag{4}$$

Similarly, the GC also has to recompute CRT'_3, CRT'_5 and CRT'_6. Thus, we have $N_0N_1|(CRT'_5 - CRT_5)$ and $N_0N_1N_2N_4|(CRT'_6 - CRT_6)$. $\mathcal{A}$ will find out the common factors of $(CRT'_5 - CRT_5)$ and $(CRT'_6 - CRT_6)$, i.e. N_0, N_1 and perhaps a few irrelevant numbers.

More dynamic changes will then leak out more information of the probable structure of the group hierarchy. For example as shown in Fig. 6, on the morrow of the modification of G_3's data key, the GC adds a new subgroup G_{10} into the group as one of G_9's immediate superior. Only CRT_9 should be updated. The GC should then recompute CRT''_9 as shown in (5).

$$\begin{cases} CRT''_9 \equiv \{k_9\}_{k_1} \bmod N_1, \\ CRT''_9 \equiv \{k_9\}_{k_2} \bmod N_2, \\ CRT''_9 \equiv \{k_9\}_{k'_3} \bmod N_3, \\ CRT''_9 \equiv \{k_9\}_{k_4} \bmod N_4, \\ CRT''_9 \equiv \{k_9\}_{k_6} \bmod N_6, \\ CRT''_9 \equiv \{k_9\}_{k_8} \bmod N_8, \\ CRT''_9 \equiv \{k_9\}_{k_{10}} \bmod N_{10}, \\ CRT''_9 \equiv \{k_9\}_{k_9} \bmod N_0. \end{cases} \tag{5}$$

Comparing (3) with (5), we'll have (6) which is similar to (4). We get that $N_0N_1N_2N_3N_4N_6N_8|(CRT''_9 - CRT'_9)$.

$$\begin{cases} CRT''_9 - CRT'_9 \equiv 0 \bmod N_1, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_2, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_3, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_4, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_6, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_8, \\ CRT''_9 - CRT'_9 \equiv 0 \bmod N_0. \end{cases} \tag{6}$$

The adversary $\mathcal{A}$ who eavesdrops on the group communications, now finds out that a user's CRT_i changed, from CRT_9 to CRT_9' and then CRT_9''. $\mathcal{A}$ gets $(CRT_9'' - CRT_9')$, $(CRT_9' - CRT_9)$. He also notices the change of a user's CRT_i from CRT_6 to CRT_6', so he gets $(CRT_6' - CRT_6)$. Through computing their greatest common divisors and then factoring them, $\mathcal{A}$ has a good chance to figure out the shared moduli like N_0, N_1, N_2, N_4, N_6, etc. Then, he may infer the hierarchical relation from moduli and CRT_is.

We shall make it clear that $\mathcal{A}$ identify G_i with its CRT_i, but he doesn't know what N_i of a subgroup G_i is, and which N_i corresponds to which G_i at the beginning. A subgroup's CRT_i changes whenever there is a subgroup dynamic change at any of its ancestors. We show as follows what $\mathcal{A}$ may know after these two moves.

After the first move, the data encryption key k_3's modification, CRT_6 and CRT_9 change to CRT_6' and CRT_9'. Both $(CRT_6' - CRT_6)$ and $(CRT_9' - CRT_9)$ share factors N_0, N_1, N_2 and N_4. $\mathcal{A}$ infers that they probably share some same ancestral subgroups. Similarly, $(CRT_5' - CRT_5)$ and $(CRT_6' - CRT_6)$ share the factor N_0 and N_1. So do $(CRT_5' - CRT_5)$ and $(CRT_9' - CRT_9)$. $\mathcal{A}$ may infer that some ancestors of G_6 and G_9, which corresponding to N_2 and N_4, are not ancestors of G_5.

After the second move, only CRT_9' changes, and CRT_6 doesn't. Then, $\mathcal{A}$ could infer that G_9 is either a sibling or a descendant of G_6. $(CRT_6' - CRT_6)$ and $(CRT_9'' - CRT_9)$ also share factors N_0, N_1, N_2 and N_4, which is same as that of $(CRT_6' - CRT_6)$ and $(CRT_9' - CRT_9)$. $\mathcal{A}$ may know that the newly added subgroup is the ancestor of G_9, but not G_6's. $(CRT_9' - CRT_9)$ and $(CRT_9'' - CRT_9')$ share N_0, N_1, N_2, N_4 and N_6, N_8. $\mathcal{A}$ infers that it's possible that one of N_6 and N_8 corresponds to G_6, and the worst case is that both of them belong to siblings of G_6.

To get more clear of this, $\mathcal{A}$ should wait for more subgroup dynamic changes, especially of G_6 or his ancestors, to take place. After a few more subgroup dynamic changes, $\mathcal{A}$ may see some relational patterns emerge repeatedly, and thus be able to draw numbers of hierarchical relations between different subgroups.

Since the modulus N_i of every leaf node subgroup G_i (such as N_5 and N_9) has never shown up in the whole procedure of the protocol , and all that $\mathcal{A}$ can do is to infer from the moduli he gets in subgroup dynamic changes, he may not be able to figure out the exact structure of the hierarchy entirely. Despite this uncertainty, our attack of revealing the hierarchy protected by CRTHACS is still worth noting.

Toy Example Here, we give a toy example of above. We set the length of the security parameter to be 10 bits. That is to say, the symmetric encryption $\{x\}_k$ is assumed to be a mapping whose range is between 1 and 2^{10}. The moduli N_i is restricted between $2^{10} + 1$ and 2^{11}.

Since the algorithm of symmetric encryption $\{x\}_k$ is irrelevant to our method of attack, we use a random table of the mapping to simulate it. We randomly choose those eleven pair-wise relatively prime numbers:

$$\begin{cases} N_1 = 1992, N_2 = 1195, N_3 = 1499, \\ N_4 = 1681, N_5 = 1037, N_6 = 1471, \\ N_7 = 1919, N_8 = 1237, N_9 = 1157, \\ N_{10} = 1087. N_0 = 1267. \end{cases} \tag{7}$$

In description of CRTHACS, N_0 is public, and all of the other N_i are kept secret.

We have encryption table of $\{k_i\}_{k_j}$ before and after k_3's change, see Table 2. We just list $\{k_i\}_{k_j}$ that are used in calculating CRT_i and leave a block empty if the $\{k_i\}_{k_j}$ is not used.

We have CRT_i, CRT'_i and CRT''_i in Table 3, and their differences in the same table.

An adversary may record the CRT_i, so he has Table 3. He tests whether a $(CRT'_i - CRT_i)$ or a $(CRT''_i - CRT'_i)$ can be divided by N_0. If not, that means k_i has changed during that session, and the corresponding difference $((CRT'_i - CRT_i)$ or $(CRT''_i - CRT'_i))$ can't be used to calculate moduli. In Table 3, we have $N_0 \nmid (CRT'_3 - CRT_3)$, $N_0|(CRT'_5 - CRT_5)$, $N_0|(CRT'_6 - CRT_6)$, $N_0|(CRT'_9 - CRT_9)$ and $N_0|(CRT''_9 - CRT'_9)$. Then, the adversary knows k_3 has changed, and he calculates:

$$\begin{cases} \mathrm{GCD}(\frac{CRT'_5 - CRT_5}{N_0}, \frac{CRT'_6 - CRT_6}{N_0}) = 7968, \\ \mathrm{GCD}(\frac{CRT'_5 - CRT_5}{N_0}, \frac{CRT'_6 - CRT_6}{N_0}, \frac{CRT'_9 - CRT_9}{N_0}) \\ = 1992, \\ \mathrm{GCD}(\frac{CRT'_9 - CRT_9}{N_0}, \frac{CRT''_9 - CRT'_9}{N_0}) \\ = 7281273177974280. \end{cases} \tag{8}$$

The adversary hereby gets "1992". If he knows that N_i is restricted between $2^{10}+1$ and 2^{11}, which is normally public, he'll be sure that G_5, G_6 and G_9 share a same ancestor whose modulus is 1992 , i. e. N_1. He also may infer that G_3 is an ancestor of G_5, G_6 and G_9. Because $N_0 \nmid (CRT'_3 - CRT_3)$, he may detect the change of G_3's data key, which results in the changes of CRT_5, CRT_6 and CRT_9.

Furthermore, the adversary also gets a number 7281273177974280, which is the very product of N_1, N_2, N_4, N_6 and N_8. It's a very useful clue for further reasoning on G_9's ancestors and the whole structure.

2.3 The Theoretical Analysis

In this section, we analyze the problem of CRTHACS. We generalize the core problem into an algebraic form, which may help us find out the similarity of those schemes that our attack can be adopted. To help with the analysis, we introduce time t into the description of CRTHACS as session ID. We know that in a group key management (GKM) protocol, we always identify a change with a session identity. These session identities are actually related with different time periods. But, in the description of the CRTHACS, there is no session identity.

Adding it into the scheme helps us describe the relations among sessions, and it is also the key to find out the vulnerability of this scheme.

Firstly, we use a more general and algebraic form to restate the problem. The general form of the Chinese Remainder Theorem can be formulated in rings and ideals. Hungerford has proved the following Theorem 1, i.e. Corollary 2.27 in his book Algebra [21].

Theorem 1 (Chinese Remainder Theorem [21]). *If $I_1, \cdots, I_\zeta$ are ideals in a ring R, then there is a monomorphism of rings*

$$\begin{aligned}&\theta : R/(I_1 \cap I_2 \cap \cdots \cap I_\zeta) \to R/I_1 \times R/I_2 \times \\ &\cdots \times R/I_\zeta.\end{aligned} \tag{9}$$

If $R^2 + I_u = R$ for all u, and $I_u + I_v = R$ (or say they are coprime) for all $u \neq v$, then θ is an isomorphism of rings.

Remark 1. If R has an identity, then $R^2 = R$, whence $R^2 + I = R$ for every ideal I of R.

In the CRTHACS, the integer ring $\mathbb{Z}$ is the instantiation of the ring R in Theorem 1. There are η ancestors $G_{i_1}, \cdots, G_{i_\eta}$ of G_i, then ζ in Theorem 1 corresponds to $\eta + 1$ in this instance. $\langle N_{i_1} \rangle, \cdots, \langle N_{i_\eta} \rangle$, and $\langle N_0 \rangle$ are $\eta + 1$ ideals $I_1, \cdots, I_{\eta+1}$ of $\mathbb{Z}$, where $\langle N_j \rangle$ stands for an ideal generated by N_j. Because $\mathbb{Z}$ is commutative, $\langle N_{i_1} \rangle \cap \cdots \cap \langle N_{i_\eta} \rangle \cap \langle N_0 \rangle = \langle N_{i_1} \cdots N_{i_\eta} N_0 \rangle$. The isomorphism is thus

$$\begin{aligned}&\theta : \mathbb{Z}/\langle N_{i_1} \cdots N_{i_\eta} N_0 \rangle \to \mathbb{Z}/\langle N_{i_1} \rangle \times \cdots \\ &\times \mathbb{Z}/\langle N_{i_\eta} \rangle \times \mathbb{Z}/\langle N_0 \rangle.\end{aligned} \tag{10}$$

That is,

$$\begin{aligned}&\theta(CRT_i + \langle N_{i_1} \cdots N_{i_\eta} N_0 \rangle) \\ &= (CRT_i + \langle N_{i_1} \rangle, \cdots, CRT_i + \langle N_{i_\eta} \rangle, CRT_i + \langle N_0 \rangle) \\ &= (rem_{i_1} + \langle N_{i_1} \rangle, \cdots, rem_{i_\eta} + \langle N_{i_\eta} \rangle, rem_{i_0} + \langle N_0 \rangle).\end{aligned} \tag{11}$$

To describe the subgroup dynamic changes in CRTHACS more precisely, we introduce time t into the scheme. We extend our notations as in Table 4. We denote a value V at time t by $V[t]$. We use the notion 'determinant subgroups' instead of ancestors in CRTHACS. After some subgroup dynamic changes, some determinant subgroups of G_i may have modified their data encryption key, left or joined in the group and etc., and certain determinant subgroups of G_i may still keep *unchanged.* We denote the set of those unchanged subgroups as $UG_i[t_1, t_2]$.

Therefore, we rewrite (11) at time t to be (12),

$$\begin{aligned}&\theta(CRT_i[t] + \langle N[t]_{i_1} \cdots N[t]_{i_{\eta[t]_i}} N_0 \rangle) \\ &= (CRT_i[t] + \langle N[t]_{i_1} \rangle, \cdots, CRT_i[t] + \langle N[t]_{i_{\eta[t]_i}} \rangle, \\ &\quad CRT_i[t] + \langle N_0 \rangle) \\ &= (rem[t]_{i_1} + \langle N[t]_{i_1} \rangle, \cdots, rem[t]_{i_{\eta[t]_i}} + \langle N[t]_{i_{\eta[t]_i}} \rangle, \\ &\quad rem[t]_{i_0} + \langle N_0 \rangle).\end{aligned} \tag{12}$$

For any determinant subgroup G_{i_l} of G_i, if $G_{i_l} \in UG_i[t_1, t_2]$, then $N[t_1]_{i_l} = N[t_2]_{i_l}$ and $rem[t_1]_{i_l} = rem[t_2]_{i_l}$. Hence we have

$$\begin{aligned} &CRT_i[t_1] + \langle N[t_1]_{i_l} \rangle = rem[t_1]_{i_l} + \langle N[t_1]_{i_l} \rangle \\ &= rem[t_2]_{i_l} + \langle N[t_2]_{i_l} \rangle = CRT_i[t_2] + \langle N[t_2]_{i_l} \rangle. \end{aligned} \tag{13}$$

(13) informs that $(CRT_i[t_2] - CRT_i[t_1]) \in \langle N[t_1]_{i_l} \rangle$, for any $G_{i_l} \in UG_i[t_1, t_2]$. As a result, we have

$$(CRT_i[t_2] - CRT_i[t_1]) \in \langle \textstyle\prod_{G_{i_l} \in UG_i[t_1,t_2]} N[t_1]_{i_l} \rangle. \tag{14}$$

Now, we may reexamine the problem of CRTHACS in the form of algebra.

We suppose that R is a unique factorization domain (UFD) in the following text. $I[t]_0, I[t]_1, \cdots, I[t]_{\eta[t]_i}$ are $\eta[t]_i + 1$ ideals of R. $I[t]_u + I[t]_v = R$ for all $u \neq v$. Therefore, they satisfy the conditions in Theorem 1. We have

$$\begin{aligned} &\theta : R/(I[t]_0 I[t]_1 \cdots I[t]_{\eta[t]_i}) \to R/I[t]_0 \times R/I[t]_1 \times \\ &\cdots \times R/I[t]_{\eta[t]_i}. \end{aligned} \tag{15}$$

Now, we substitute $\mathbb{Z}$ and $\langle N[t]_i \rangle$ in (10–14) with R and $I[t]_i$, and retain other notations in Table 4. We give as follows an exact definition of the critical feature of CRTHACS that makes our attack feasible.

Definition 1. *We say a scheme* PT *is* double-invariant *if it holds* $rem[t_1]_{i_l} = rem[t_2]_{i_l}$ *and* $I[t_1]_{i_l} = I[t_2]_{i_l}$ *when* $G_{i_l} \in UG_i[t_1, t_2]$.

CRTHACS is an example of *double-invariant* schemes. Now, we generalize our attack to *double-invariant* schemes in the form of the following theories.

Lemma 1. *In a double-invariant scheme* PT, *R is a unique factorization domain. $I[t]_i$, $i = 0, \cdots, \eta[t]_i$ are $\eta[t]_i$ pair-wise coprime ideals of R. θ is a monomorphism described by* (15)*. If there exists a nonempty set $UG_i[t_1, t_2]$, then*

$$(CRT_i[t_2] - CRT_i[t_1]) \in \textstyle\prod_{G_{i_l} \in UG_i[t_1,t_2]} I[t_1]_{i_l}, \tag{16}$$

where t_1 and t_2 are two distinct time points, satisfying $t_1 < t_2$.

Proof. Follow the deduction above.

If in R factorization can be practically solved, then our attack will succeed in some trivial steps. But usually it is not. Moreover, (16) means that $(CRT_i[t_2] - CRT_i[t_1])$ has the factors that are relevant to ideals corresponding to subgroups in it, but still could have some factors that are random and irrelevant. Then, without further steps, we can't recognize any proper relation, ie. $(CRT_i[t_2] - CRT_i[t_1])$ and $(CRT_j[t_4] - CRT_j[t_3])$ for any i, j and time t may not exhibit any explicit relation such as divisibility etc. Thus, we shall go further at least to figure out those factors or the exact product of them that consist of no irrelevant elements.

Definition 2. *We call* $CoIt[a(t_1,t_2),b(t_3,t_4)]$ *a* conditional intersection*, which is defined to be the set* $\{G_j : G_j \in UG_a[t_1,t_2] \bigcap UG_b[t_3,t_4], I[t_1]_j = I[t_3]_j\}$*, where* $t_1 < t_2$*,* $t_3 < t_4$*,* U_a *and* U_b *are two subgroups.*

This definition helps us identify subgroups whose corresponding ideals $\mathcal{A}$ may figure out. It can be applied to more than two UGs in a similar way.

$CoIt[a(t_1,t_2),b(t_3,t_4)]$ is slightly different from $UG_a[t_1,t_2] \bigcap UG_b[t_3,t_4]$. There may exist a subgroup U_j keeping unchanged during t_1 to t_2 and t_3 to t_4, but having changed in time between t_2 and t_3. That is to say, though $I[t_1]_j = I[t_2]_j$ and $I[t_3]_j = I[t_4]_j$, $I[t_1]_j \neq I[t_3]_j$. As a result, $j \in UG_a[t_1,t_2] \bigcap UG_b[t_3,t_4]$, but $j \notin CoIt[a(t_1,t_2),b(t_3,t_4)]$. $CoIt[a(t_1,t_2),b(t_3,t_4)]$ is also different from $UG_a[t_1,t_4] \bigcap UG_b[t_1,t_4]$. For $U_j \in CoIt[a(t_1,t_2),b(t_3,t_4)]$, $rem[t]_j$ may change during time t_2 to t_3, but $I[t]_j$ shall not. But for $U_j \in UG_a[t_1,t_4] \bigcap UG_b[t_1,t_4]$, both $rem[t]_j$ and $I[t]_j$ keep unchanged.

We know that in a unique factorization domain R, any two elements $u, v \in R$ have a greatest common divisor (gcd) d. If R is a principal ideal domain (PID), d can be represented by a linear combination of u and v. Furthermore, if R is a Euclidean domain, d can be efficiently computed by Euclidean Algorithm.

Lemma 2. *In a double-invariant scheme* PT*,* R *is a Euclidean domain.* $I[t]_i$*,* $i = 0, \cdots, \eta[t]_i$ *are* $\eta[t]_i$ *pair-wise coprime ideals of* R*.* θ *is a monomorphism described by* (15)*. If there exists a nonempty set* $CoIt[a(t_1,t_2),b(t_3,t_4)]$*, then the greatest common divisor* d *as*

$$\begin{aligned} &d_{a(t_1,t_2),b(t_3,t_4)} \\ &= \mathrm{GCD}(CRT_a[t_1] - CRT_a[t_2], CRT_b[t_3] - CRT_b[t_4]) \end{aligned} \tag{17}$$

can be worked out efficiently, where $\mathrm{GCD}(\alpha,\beta)$ *means the greatest common divisor of* α *and* β*.*

Proof. Follow the deduction above.

When $a = b$ and $t_2 = t_3$ in $CoIt[a(t_1,t_2),b(t_3,t_4)]$, as a special case of Lemma 2, we have

Lemma 3. *In a double-invariant scheme* PT*,* R *is a Euclidean domain.* $I[t]_i$*,* $i = 0, \cdots, \eta[t]_i$ *are* $\eta[t]_i$ *pair-wise coprime ideals of* R*.* θ *is a monomorphism described by* (15)*. If there exists a nonempty set* $UG_a[t_1,t_4]$*, where* t_1*,* t_2 *and* t_4 *are three distinct time points,* $t_1 < t_2 < t_4$*, then the greatest common divisor* d *as*

$$\begin{aligned} &d_{a(t_1,t_4)} = \\ &\mathrm{GCD}(CRT_a[t_2] - CRT_a[t_1], CRT_a[t_4] - CRT_a[t_2]) \end{aligned} \tag{18}$$

can be worked out efficiently.

Here, $UG_a[t_1,t_4] = CoIt[a(t_1,t_2),b(t_3,t_4)]$. Now, we have our main statement as follows.

Theorem 2. *In a double-invariant scheme* PT*, R is a Euclidean domain. $I[t]_i$, $i = 0, \cdots, \eta[t]_i$ are $\eta[t]_i$ pair-wise coprime ideals of R. If the dynamic changes of determinant subgroups of a subgroup happen not only once, then we have a good chance to figure out the product of moduli of some of its determinant subgroups during dynamic changes (or the product of generators of ideals corresponding to those determinant subgroups).*

Proof. Because every Euclidean domain R is also a principal ideal domain, the generator of every ideal $I[t]_i$ is an element $\xi[t]_i \in R$. Since the subgroup dynamic changes usually happen once on a determinant subgroup of a subgroup, nonempty UGs and $CoIt$s emerge continuously in dynamic scenarios. We assume that there are μ nonempty unchanged sets $UG_{a_i}[t_{2i-1}, t_{2i}]$, $i = 1, \cdots, \mu$, and a nonempty conditional intersection $CoIt[a_1(t_1, t_2), \cdots, a_\mu(t_{2\mu-1}, t_{2\mu})]$. Here we shall make clear that a_i is identifier of subgroup, two different a_is may possibly identify a same subgroup G_j.

By Lemma 1, we have

$$\begin{cases} \prod_{G_j \in UG_{a_1}[t_1,t_2]} \xi[t_1]_j | (CRT_{a_1}[t_2] - CRT_a[t_1]), \\ \cdots \\ \prod_{G_j \in UG_{a_\mu}[t_1,t_2]} \xi[t_{2\mu-1}]_j | (CRT_{a_\mu}[t_{2\mu}] \\ -CRT_{a_\mu}[t_{2\mu-1}]). \end{cases} \tag{19}$$

Then, by Lemma 2, we have

$$\begin{aligned} & d_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} \\ & = \mathrm{GCD}(CRT_{a_1}[t_2] - CRT_a[t_1], \cdots, CRT_{a_\mu}[t_{2\mu}] \\ & \quad -CRT_{a_\mu}[t_{2\mu-1}]) \\ & = \Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} \\ & \quad \times \prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j, \end{aligned} \tag{20}$$

where $\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})}$ denotes: $\frac{\mathrm{GCD}(CRT_{a_1}[t_2]-CRT_a[t_1],\cdots,CRT_{a_\mu}[t_{2\mu}]-CRT_{a_\mu}[t_{2\mu-1}])}{\prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j}$.

We may rewrite (19) with notion of $CoIt$ as

$$\begin{cases} CRT_{a_1}[t_2] - CRT_{a_1}[t_1] = \delta_{a_1(t_1,t_2)} \\ \times \prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j, \\ \cdots \\ CRT_{a_\mu}[t_{2\mu}] - CRT_{a_\mu}[t_{2\mu-1}] = \delta_{a_\mu(t_{2\mu-1},t_{2\mu})} \times \\ \prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j, \end{cases} \tag{21}$$

where $\delta_{a_i(t_{2i-1},t_{2i})}, i = 1, \cdots, \mu$ are μ numbers, each of which is a product of a few random numbers. For detail, see A.2. It holds that

$$\begin{aligned} & \Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} \\ & = \mathrm{GCD}(\delta_{a_1(t_1,t_2)}, \cdots, \delta_{a_\mu(t_{2\mu-1},t_{2\mu})}). \end{aligned} \tag{22}$$

If $\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} = 1$, then we get $\prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j$ from (20), the product of generators of the ideals corresponding to those subgroups in $CoIt$. In this case, $d_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})}$ is the very product we need, i.e. our attack succeeds.

The probability of $\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} = 1$ mainly depends on the following three factors:

1. what the ring R is;
2. the relations between UGs and $CoIt$;
3. the types of subgroup dynamic changes in it.

Different rings have different rules of coprime, and thus they result in different probabilities.

If R is the integer ring $\mathbb{Z}$, by the knowledge of probabilistic number theory [39], the probability of two random numbers α, β to be coprime is $p_z = \text{Prob}[\text{GCD}(\alpha, \beta) = 1 : \alpha, \beta \in \mathbb{Z}] = \frac{6}{\pi^2} \approx 0.6079$.

If R is a polynomial ring over a finite field F_q where q is a prime number, by Sugita's analysis [38], the probability of two random polynomial f, g to be coprime is

$$p_f = \text{Prob}[\text{GCD}(f, g) = 1 : f, g \in F_q[x]] = \tfrac{q-1}{q}. \tag{23}$$

For the latter two factors, we took the integer ring $\mathbb{Z}$ as our example, furthered our reasoning in detail. For the sake of length, we leave them in Appendix A. In A.2, we give the estimated ranges of possibilities of success in consideration of different types of subgroup dynamic changes and relations between UGs and $CoIt$. Generally, we find that the more the dynamic moves an adversary captures, the greater his probabilities of success are. The probabilities of success of our attack are pretty high.

Experiments in A.3 justify our analysis.

Theorem 2 is the main reason why our attack works. If the generators of ideals in a scheme are prime, we have the following corollary.

Corollary 1. *In a double-invariant scheme* PT*, R is a Euclidean domain. If all ideals $I[t]_i$ in (15) are* prime*, the factorization is practical in R and the dynamic changes of determinant subgroups of a subgroup happen not only once, then we have a good chance to figure out the generators of ideals of some of its unchanged determinant subgroups.*

2.4 Countermeasure

We find from the lemmas above and Theorem 2, that the *double-invariance* of CRTHACS is the critical cause of its security vulnerability. So, removing it is the key to protect schemes from our attack. We can fulfil this by adding some session/time-related 'salt' into $rem[t]_i$ or $I[t]_i$ to make them distinct in each session/time.

In CRTHACS, for a subgroup G_i, GC computes CRT_i through (1), which can be represented as (24) at t_1.

$$\begin{cases} CRT_i[t_1] \equiv rem[t_1]_{i_1} \bmod N[t_1]_{i_1}, \\ CRT_i[t_1] \equiv rem[t_1]_{i_2} \bmod N[t_1]_{i_2}, \\ \cdots \\ CRT_i[t_1] \equiv rem[t_1]_{i_{i_\eta}} \bmod N[t_1]_{i_\eta}, \\ CRT_i[t_1] \equiv \{k_i\}_{k_i} \bmod N_0. \end{cases} \tag{24}$$

In CRTHACS, if a subgroup change of an ancestor of G_i takes place at time t_2, to any other ancestors G_{i_l} of G_i, it still holds that $rem[t_1]_{i_l} = rem[t_2]_{i_l} = \{k_i\}_{k_{i_l}}$ and $N[t_1]_{i_l} = N[t_2]_{i_l} = N_{i_l}$. This is the property we defined in Definition 1, i.e. *double-invariance*. We can remove this property by letting $rem[t]_{i_l}$ or $N[t]_{i_l}$ be different at every time point t. To avoid coprime tests of $N[t]_{i_l}$ for different i_l, we can merely change $rem[t]_{i_l}$ at each time t. One method is to add a hash of time as a time stamp into $rem[t]_{i_l}$, that is, to let $rem[t]_{i_l} = \{k_i||hash(t)\}_{k_{i_l}}$, and when someone sends $CRT_i[t]$ he always sends the time t or $hash(t)$ with it. The *hash* function is a public cryptographic hash function with a negligible probability of collision. We omit the formal proof here for the sake of length.

3 Our Attack on Other Schemes

The general form of Chinese Remainder Theorem, i.e. Theorem 1, plays a very important part in modern cryptography.

We notice that instances of general form of CRT that are applied to build secret sharing schemes, could also be used to build key management protocols for a group with hierarchy. For example, Chinese Reminder Theorem is applied to build a secret sharing scheme [2] and a key management protocol for HAC i.e. CRTHACS [49, 50]. Other types of secret sharing schemes such as Shamir's [35], Blakeley's [8], Karnin's [28], Brickell's [10], Wu's scheme [45], could also be transformed to group key management protocols for HAC. All the methods of these secret sharing schemes are the very instances of general form of CRT. Shamir's secret sharing scheme [35] may probably be the best known example of the above. His scheme makes use of polynomial interpolation over a finite field. We represent the interpolation method in a form similar to (9) as follows.

In a finite filed F, given ζ points $(x_1, y_1), \cdots, (x_\zeta, y_\zeta)$ with distinct x_i's, we are able to build up a polynomial $f(x)$ of degree $(\zeta - 1)$ by using following equations,

$$\begin{cases} f(x) \equiv y_1 \bmod (x - x_1), \\ f(x) \equiv y_2 \bmod (x - x_2), \\ \cdots \\ f(x) \equiv y_\zeta \bmod (x - x_\zeta). \end{cases} \tag{25}$$

In fact, this is an isomorphism described by

$$\begin{aligned} &\theta : F[x]/\langle (x - x_1) \cdots (x - x_\zeta) \rangle \\ &\to F[x]/\langle x - x_1 \rangle \times \cdots \times F[x]/\langle x - x_\zeta \rangle, \end{aligned} \tag{26}$$

where $F[x]$ is a polynomial ring over F, $\langle x - x_i \rangle$ is an ideal generated by $(x - x_i)$, $F[x]/\langle x - x_i \rangle$ is a quotient ring. Since F is a field, $F[x]$ is a Euclidean domain.

Many schemes of group key management for HAC make use of this, such as Shen's [36], Wu's scheme [44] and etc., and the modified versions [15, 20, 40] of them. Unfortunately, all of the above are *double-invariant* schemes and hence vulnerable to our attack.

Take Das's scheme [15] as an example. Das's scheme is a hierarchical group key management scheme using Newton's interpolation function [34]. It's applied to a tree-like hierarchy as shown in Fig. 7.

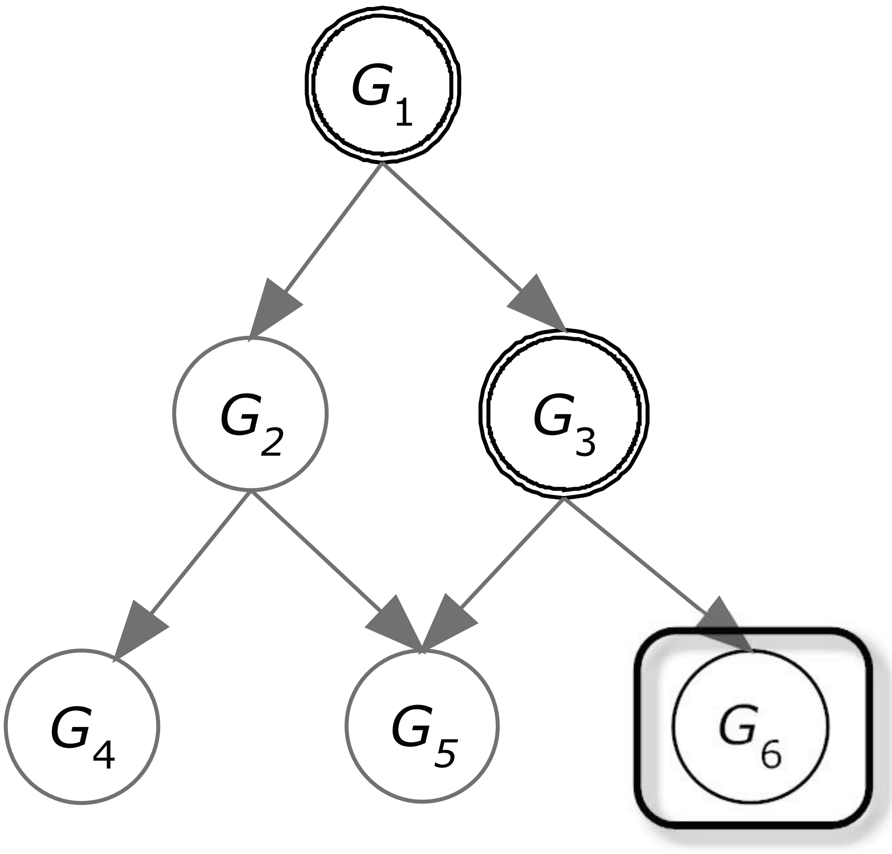

Fig. 7. A subgroup hierarchy for Das's scheme.

All computations run over a finite field F. In his scheme, k_i is G_i's data encryption key, ID_i is G_i's unique identity. h is a cryptographic hash function. GC computes Newton's interpolation function $H_i(x)$ at points $(h(\mathrm{ID}_{i_l}||k_i), k_{i_l})$, for all i_l such that $G_{i_l} \preceq G_i$. Then GC publishes $H_i(x)$.

$$\begin{cases} H_i(x) \equiv k_{i_1} \bmod (x - h(\mathrm{ID}_{i_1}||k_i)), \\ H_i(x) \equiv k_{i_2} \bmod (x - h(\mathrm{ID}_{i_2}||k_i)), \\ \cdots \\ H_i(x) \equiv k_{i_\eta} \bmod (x - h(\mathrm{ID}_{i_\eta}||k_i)). \\ H_i(x) \equiv k_i \bmod (x - h(\mathrm{ID}_i||k_i)). \end{cases} \tag{27}$$

Then G_i is able to derive his descendant G_{i_l}'s data encryption key k_{i_l} as $k_{i_l} = H_i(h(\mathrm{ID}_{i_l}||k_i))$.

Compare it with the notations in Table 4, in Das's scheme at time t, $H_i(x)$ corresponds to $CRT_i[t]$, $\langle(x - h(\mathrm{ID}_{i_l}||k_i))\rangle$ corresponds to $I[t]_{i_l}$, and k_{i_l} corresponds to $rem[t]_{i_l}$. $AG_i[t] = \{G_{i_l} : G_{i_l} \preceq G_i\}$. $UG_i[t_1, t_2]$ is the set of unchanged descendants of G_i (plus G_i itself) from time t_1 to t_2.

Since if a subgroup dynamic change takes place at a descendant G_{i_c} of G_i during time between t_1 and t_2, we can see that both k_{i_l} and $h(\mathrm{ID}_{i_l}||k_i)$ of G_i's descendants $G_{i_l} \in UG_i[t_1, t_1]$ keep unchanged. Thus, this scheme is *double-invariant*. By Theorem 2, Das's scheme is not secure in dynamic scenarios.

For example, we assume that at time t_1 Das's scheme is applied to a group with the hierarchy in Fig. 7, and G_6 happens to leave the group on the morrow of that at t_2. For G_3, We have

$$\begin{cases} H_3(x) \equiv k_5 \bmod (x - h(\mathrm{ID}_5||k_3)), \\ H_3(x) \equiv k_6 \bmod (x - h(\mathrm{ID}_6||k_3)), \\ H_3(x) \equiv k_3 \bmod (x - h(\mathrm{ID}_3||k_3)), \end{cases} \tag{28}$$

at time t_1, and

$$\begin{cases} H'_3(x) \equiv k_5 \bmod (x - h(\mathrm{ID}_5||k_3)), \\ H'_3(x) \equiv k_3 \bmod (x - h(\mathrm{ID}_3||k_3)), \end{cases} \tag{29}$$

at time t_2.

By Lemma 1, we have $(x-h(\mathrm{ID}_5||k_3))(x-h(\mathrm{ID}_3||k_3))|(H'_3(x)-H_3(x))$. Since finding roots of a polynomial of degree 3 over a finite field can be practically solved by algorithms [6, 9, 27], $h(\mathrm{ID}_5||k_3)$ and $h(\mathrm{ID}_3||k_3)$ can be worked out as roots of $(H'_3(x) - H_3(x))$. Then, we can get data encryption keys of G_3 and G_5: $k_3 = H_3(h(\mathrm{ID}_3||k_3))$, $k_5 = H_3(h(\mathrm{ID}_5||k_3))$. Actually, all data encryption keys of the group except k_6 will be revealed in the same way if our object of analysis is G_1 instead of G_3. Furthermore, by Lemma 3 and Theorem 2, it'll be much easier to break the scheme if there are two or more dynamic changes taking place in the group. We use the special form of Lemma 2, which is Lemma 3, since in Das's scheme whenever $a \neq b$, $CoIt[a(t_1, t_2), b(t_3, t_4)]$ is an empty set.

We shall mention that our attack on Das's scheme is very similar to Wang's attack [41] on Shen's scheme [36]. It's easy to see that Wang's attack is another instance of our Theorem 2.

The countermeasure to fix Das's scheme is very similar to that of CRTHACS shown in Section 2.4. So, we omit it here. Zou's another scheme [48] is also an HAC scheme that makes use of polynomial interpolation, and it successfully avoided *double-invariance* by adding a random integer z into $I[t]_i$ as a session identity (session-related 'salt'). However, we would like to remind readers that it's not the randomness but the uniqueness of z because of which the property of *double-invariance* is removed off.

4 Conclusion

In this paper, we gave an attack to reveal the structures of the hierarchy that the modified CRTHACS [50] intended to hide. We generalized the attack in the form of rings and their isomorphisms. We redefined the problem in a more proper form represented in time, as shown in Table 4 and (12), (13), (15). We defined the key vulnerability of this scheme to our attack as *double-invariance* and showed how to resist the attack in the general form.

We also pointed out that a series of HAC schemes share the same problem with CRTHACS, and gave an exemplary attack on one of them, i.e. Das's scheme [15], so as to remind readers that this vulnerability doesn't only belong to CRTHACS but is of some generality in certain types of HAC schemes.

To design HAC protocols is "deceptively easy", we shall be more careful than ever. "Provable security" and "reductionist" tradition offers a body of work to follow, including models and definitions as well as examples [5, 11, 26]. Some studies [3, 16, 43] have already taken use of them to design provably secure HAC schemes.

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. C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29(2):208–210, 1983.
3. Mikhail J. Atallah, Marina Blanton, Nelly Fazio, and Keith B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12(3):1–43, 2009. 1455531.
4. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *STOC '95, Proceedings of the 27 annual ACM symposium on Theory of computing*, 1995.
5. Mihir Bellare, Phillip Rogaway, and Douglas Stinson. Entity authentication and key distribution. In *Advances in Cryptology, CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin / Heidelberg, 1994.
6. E. R. Berlekamp. Factoring polynomials over finite fields. *Bell system technical journal*, 46(1853-1859):3, 1967.
7. Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *Crytography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer Berlin Heidelberg, 1997.
8. G. R. Blakley. Safeguarding cryptographic keys. In *FIPS Conference Proceedings*, volume 48, pages 313–317. AFIPS Press, 1979. Proc. NCC.

9. W. Bosma, J. Cannon, C. Playoust, and A. Steel. *Solving problems with MAGMA*, chapter 3. 1999.
10. E. F. Brickell. Some ideal secret sharing schemes. In *EUROCRYPT '89*, volume 434 of *Advances in cryptology: proceedings*, page 468. Springer, 1989.
11. Ran Canetti, Hugo Krawczyk, and Birgit Pfitzmann. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology ,EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer Berlin / Heidelberg, 2001.
12. Gerald Chick, Stafford Tavares, and Gilles Brassard. Flexible access control with master keys. In *Advances in Cryptology CRYPTO 89*, volume 435 of *Lecture Notes in Computer Science*, pages 316–322. Springer Berlin / Heidelberg, 1990.
13. Tan Chik-How and Yang Guomin. Comments on "Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting". *IEEE Transactions on Information Theory*, 56(11):5887–5888, 2010.
14. J. Crampton, K. Martin, and P. Wild. On key assignment for hierarchical access control. In *19th IEEE Computer Security Foundations Workshop*, pages 14 pp.–111, 2006.
15. Manik Lal Das, Ashutosh Saxena, Ved P. Gulati, and Deepak B. Phatak. Hierarchical key management scheme using polynomial interpolation. *SIGOPS Oper. Syst. Rev.*, 39(1):40–47, 2005. 1044556.
16. A. De Santis, A. Ferrara, and B. Masucci. Efficient provably-secure hierarchical key assignment schemes. In *Mathematical Foundations of Computer Science*, pages 371–382, 2007.
17. R. Dutta and R. Barua. Provably secure constant round contributory group key agreement in dynamic setting. *IEEE Transactions on Information Theory*, 54(5):2007–2025, 2008.
18. W. Geiselmann and R. Steinwandt. Attacks on a secure group communication scheme with hierarchical access control. In *International Symposium on Information Theory (ISIT 2004)*, page 14, 2004.
19. S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography.* 2008. p210.
20. Chien-Lung Hsu and Tzong-Sun Wu. Cryptanalyses and improvements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy. *Computers & Security*, 22(5):453–456, 2003. doi: 10.1016/S0167-4048(03)00514-5.
21. T. W. Hungerford. Algebra, volume 73 of graduate texts in mathematics, 1980.
22. Min-Shiang Hwang. An improvement of a dynamic cryptographic key assignment scheme in a tree hierarchy. *Computers & Mathematics with Applications*, 37(3):19–22, 1999. doi: 10.1016/S0898-1221(99)00042-5.
23. Min-Shiang Hwang. Cryptanalysis of YCN key assignment scheme in a hierarchy. *Inf. Process. Lett.*, 73(3-4):97–101, 2000. 343065.
24. Min-Shiang Hwang and Wei-Pang Yang. Controlling access in large partially ordered hierarchies using cryptographic keys. *Journal of Systems and Software*, 67(2):99–107, 2003. doi: 10.1016/S0164-1212(02)00091-2.
25. Mara Isabel Gonzlez Vasco Jens-Matthias Bohli and Rainer Steinwandt. Secure group key establishment revisited. *International Journal of Information Security*, 6(4):12, 2007.
26. Katz Jonathan and Shin Ji Sun. Modeling insider attacks on group key-exchange protocols, 2005. 1102146 180-189.
27. E. Kaltofen. Polynomial factorization: a success story. In *Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, Philadelphia, PA, USA., 2003. ACM. 860857 3-4.

28. E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.
29. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. *Advances in Cryptology-CRYPTO 2003*, pages 110–125, 2003.
30. A. V. D. Kayem, P. Martin, and S. G. Akl. Heuristics for improving cryptographic key assignment in a hierarchy. In *In 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07)*, volume 1, pages 531–536, 2007.
31. H. T. Liaw, S. J. Wang, and C. L. Lei. A dynamic cryptographic key assignment scheme in a tree structure. *Computers & Mathematics with Applications*, 25(6):109–114, 1993. doi: 10.1016/0898-1221(93)90305-F.
32. C. H. Lin. Dynamic key management schemes for access control in a hierarchy. *Computer communications*, 20(15):1381–1385, 1997. ISSN: 0140-3664.
33. R. S. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27:95–98, 1988.
34. J. B. Scarborough. *Numerical Mathematical Analysis.* John Hopkins Press, Baltimore, 4th edn edition, 1958.
35. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
36. Victor R. L. Shen and Tzer-Shyong Chen. A novel key management scheme based on discrete logarithms and polynomial interpolations. *Computers & Security*, 21(2):164–171, 2002. doi: 10.1016/S0167-4048(02)00211-0.
37. Victor Shoup and Avi Rubin. Session key distribution using smart cards. In Ueli Maurer, editor, *Advances in Cryptology, EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 321–331. Springer Berlin Heidelberg, 1996.
38. H. Sugita and S. Takanobu. The probability of two F_q-polynomials to be coprime. *Probability and number theory, Advanced Studies in Pure Mathematics*, 49:455–478, 2007.
39. G. Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 46. Cambridge Univ Pr, 1995.
40. Shiang-Feng Tzeng, Cheng-Chi Lee, and Tzu-Chun Lin. A novel key management scheme for dynamic access control in a hierarchy. *International Journal of Network Security*, 12(3), 2011.
41. Shyh-Yih Wang and Chi Sung Laih. Cryptanalyses of two key assignment schemes based on polynomial interpolations. *Computers & Security*, 24(2):134–138, 2005. doi: 10.1016/j.cose.2004.07.002.
42. Shyh-Yih Wang and Chi-Sung Laih. Cryptanalysis of Hwang-Yang scheme for controlling access in large partially ordered hierarchies. *Journal of Systems and Software*, 75(1-2):189–192, 2005. doi: 10.1016/j.jss.2004.04.015.
43. Jiang Wu, Ruizhong Wei, Bart Preneel, and Stafford Tavares. An access control scheme for partially ordered set hierarchy with provable security. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 221–232. Springer Berlin / Heidelberg, 2006.
44. T. C. Wu and C. C. Chang. Cryptographic key assignment scheme for hierarchical access control. *International Journal of Computer Systems Science and Engineering*, 16(1):25–28, 2001.
45. Tzong-Chen Wu and Wei-Hua He. A geometric approach for sharing secrets. *Computers & Security*, 14(2):135–145, 1995.
46. Shouhuai Xu. On the security of group communication schemes. *Journal of Computer Security*, 15(1/2007):40, 2007.

47. J. Yeh, R. Chow, and R. Newman. A key assignment for enforcing access control policy exceptions. In *International Symposium on Internet Technology*, pages 54–59, 1998.
48. 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.
49. X. Zou, B. Ramamurthy, and S. S. Magliveras. Chinese remainder theorem based hierarchical access control for secure group communication. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *Information and Communications Security*, volume 2229 of *Lecture Notes in Computer Science*, pages 381–385. Springer Berlin / Heidelberg, 2001.
50. X. Zou, B. Ramamurthy, and S. S. Magliveras. A GCD attack resistant CRTHACS for secure group communications. In *The International Conference on Information Technology: Coding and Computing (ITCC 2004)*, volume 2, pages 153–154 Vol.2, 2004.
51. X. Zou, B. Ramamurthy, and S. S. Magliveras. *Secure group communications over data networks.* Springer, 2005.

A Notes on theorem 2

In our Theorem 2, in order to estimate the probability of $\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} = 1$ in (22), we have to take an examination on the probability of coprimality of numbers, each of which is a product of several random numbers.

A.1 Probability of Coprimality

The probability $\Pr(1,1)$ of two random numbers A_1 and A_2 to be coprime can be deducted as follows.

For a prime number p, the probability of $p \nmid \mathrm{GCD}(A_1, A_2)$ is $1-\frac{1}{p^2}$. Therefore, we have $\Pr(1,1) = \prod_p (1 - \frac{1}{p^2})$. We may know from further deduction that $\frac{1}{\Pr(1,1)} = \sum_n \frac{1}{n^2} = \frac{\pi^2}{6}$, and thus $\Pr(1,1) = \frac{6}{\pi^2} \approx 0.6080$.

Assuming that we have i_1+i_2 random numbers $A_1,\cdots,A_{i_1}$ and $B_1,\cdots,B_{i_2}$, what is the probability $\Pr(i_1,i_2)$ of the two numbers $SA = A_1 \times \cdots \times A_{i_1}$ and $SB = B_1 \times \cdots \times B_{i_2}$ to be coprime? Deduction is similar to that of above. For any prime number p, the probability of $p \nmid \mathrm{GCD}(SA, SB)$ is $1 - [1-(1-\frac{1}{p})^{i_1}] \times [1-(1-\frac{1}{p})^{i_2}]$. Thus, we have:

$$\begin{aligned}&\Pr(i_1,i_2)\\ &= \textstyle\prod_p \{1 - [1-(1-\frac{1}{p})^{i_1}] \times [1-(1-\frac{1}{p})^{i_2}]\}.\end{aligned} \tag{30}$$

Similarly, we can get the probability $\Pr(i_1,\cdots,i_\nu)$ of ν numbers, each of which is a product of i_j random numbers, where $j = 1,\cdots,\nu$.

$$\begin{aligned}&\Pr(i_1,\cdots,i_\nu)\\ &= \textstyle\prod_p \{1 - [1-(1-\frac{1}{p})^{i_1}] \times \cdots \times [1-(1-\frac{1}{p})^{i_\nu}]\}.\end{aligned} \tag{31}$$

It's hard for us to give the exact analytical form of (31) like what we did to $\Pr(1,1)$, so we use the asymptotic result instead. For example, $\Pr(1,2) \approx 0.4283$ and $\Pr(2,3,5) \approx 0.1926$.

A.2 Different Cases in Subgroup Dynamics

There are several kinds of subgroup dynamic changes, mainly including a subgroup's leaving, joining the entire group, data key changes of it and etc. Different dynamic changes lead to different probability of $\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})} = 1$, which is because they form different types of products of random numbers, and thus related to the probability of coprimality we studied in A.1. We explain it in detail as follows.

Take CRTHACS as the example. In our heuristic attack in Section 2.2, we suppose that the system initialled at t_1. We put an imaginary subgroup G_0 corresponding to N_0. It happens two times of subgroup changes; one is the change of data key k_3 of G_3 at t_2, the other one is adding a new subgroup G_{10} at t_3.

By the solution of Chinese Remainder Theorem, after k_3's change, it holds that,

$$\begin{aligned} CRT_5 - CRT_5' &= Inv_{5,3} \times N_0N_1(\{k_5\}_{k_3} - \{k_5\}_{k_3'}), \\ CRT_6 - CRT_6' &= Inv_{6,3} \times N_0N_1N_2N_4 \\ &\qquad \times (\{k_6\}_{k_3} - \{k_6\}_{k_3'}), \\ CRT_9 - CRT_9' &= Inv_{9,3} \times N_0N_1N_2N_4N_6N_8 \\ &\qquad \times (\{k_9\}_{k_3} - \{k_9\}_{k_3'}), \end{aligned} \tag{32}$$

where $Inv_{5,3}, Inv_{6,3}$ and $Inv_{9,3}$ satisfy:

$$\begin{cases} Inv_{5,3} \times N_0N_1 \equiv 1 \bmod N_3, \\ Inv_{6,3} \times N_0N_1N_2N_4 \equiv 1 \bmod N_3. \\ Inv_{9,3} \times N_0N_1N_2N_4N_6N_8 \equiv 1 \bmod N_3. \end{cases} \tag{33}$$

In the new notations of Table 4, it holds that

$$\begin{cases} UG_5[t_1,t_2] = \{G_0, G_1\}, \\ UG_6[t_1,t_2] = \{G_0, G_1, G_2, G_4\}, \\ UG_9[t_1,t_2] = \{G_0, G_1, G_2, G_4, G_6, G_8\}, \\ CoIt[5(t_1,t_2), 6(t_1,t_2)] = \{G_0, G_1\}. \end{cases} \tag{34}$$

Comparing (34) to (21) and (22), we have

$$\begin{aligned} &\Delta_{5(t_1,t_2),6(t_1,t_2)} \\ &= \tfrac{\mathrm{GCD}(CRT_5 - CRT_5', CRT_6 - CRT_6')}{N_0N_1} \\ &= \mathrm{GCD}[Inv_{5,3} \times (\{k_5\}_{k_3} - \{k_5\}_{k_3'}), \\ &\qquad Inv_{6,3} \times N_2N_4 \times (\{k_6\}_{k_3} - \{k_6\}_{k_3'})]. \end{aligned} \tag{35}$$

We may treat N_i and $(\{k_j\}_{k_3} - \{k_j\}_{k_3'})$ as random numbers. Since $Inv_{5,3}$ and $Inv_{6,3}$ are determined by (33), they can't be considered as random. However, we can't treat them as fixed numbers that do not affect the probability. We may treat them as *semi-random.*

Though we don't know the exact probability of $\Delta_{5(t_1,t_2),6(t_1,t_2)} = 1$, we may estimate the range of it. It's larger than $\Pr(2,4)$ when we consider Inv as random, and smaller than $\Pr(1,3)$ when otherwise:

$$\Pr(2,4) < \mathrm{Prob}[\Delta_{5(t_1,t_2),6(t_1,t_2)} = 1] < \Pr(1,3), \tag{36}$$

i.e. it is between 0.0911 and 0.3371.

With slight difference, we have that the probability $\mathrm{Prob}[\varDelta_{5(t_1,t_2),6(t_1,t_2),9(t_1,t_2)} = 1]$ is in range

$$\begin{aligned}&[\Pr(2,4)\Pr(2,2) + (1 - \Pr(2,4))\Pr(2,3),\\ &\quad \Pr(1,3)\Pr(1,2) + (1 - \Pr(1,3))\Pr(1,3)],\end{aligned} \tag{37}$$

i.e. it is in range $[0.1391, 0.3679]$.

After t_3, a new subgroup G_{10} is added in the hierarchy. We denotes $M := N_0N_1N_2N_3N_4N_6N_8$. We have,

$$\begin{aligned}&CRT_9'' - CRT_9' = N_0N_1N_2N_4N_6N_8 \times N_3\\ &\quad\times[\tfrac{Inv_{9,0}''\times N_{10}-Inv_{9,0}'}{N_0} \times \{k_9\}_{k_9}\\ &\quad\ +\tfrac{Inv_{9,3}''\times N_{10}-Inv_{9,3}'}{N_3} \times \{k_9\}_{k_3'}\\ &\quad\ +Inv_{9,10}'' \times \{k_9\}_{k_{10}}\\ &\quad\ + \sum_{i=1,i\neq 3}^{9} (\tfrac{Inv_{9,i}''\times N_{10}-Inv_{9,i}'}{N_i} \times \{k_9\}_{k_i})],\\ &:= M \times TotalSuM,\end{aligned} \tag{38}$$

where $TotalSuM$ denotes what is in the square bracket in (38), $Inv_{9,i}''$ and $Inv_{9,i}'$ satisfy

$$\begin{cases} Inv_{9,i}'' \times \frac{M\times N_{10}}{N_i} \equiv 1 \bmod N_i,\\ Inv_{9,i}' \times \frac{M}{N_i} \equiv 1 \bmod N_i,\end{cases} \tag{39}$$

for $i = 0, \cdots, 9$. $Inv_{9,10}''$ satisfies

$$Inv_{9,10}'' \times M \equiv 1 \bmod N_{10}. \tag{40}$$

Thus by (39), $N_i|(Inv_{9,i}'' \times N_{10} - Inv_{9,i}')$, where $i = 0, \cdots, 9$. Thus, $TotalSuM$ is an integer. Since in it there are many $\{k_i\}_{k_j}$s, which we consider as random, we may treat $TotalSuM$ as a random integer.

Using the new notations, we have

$$\begin{cases} UG_9[t_2, t_3] = \{G_0, G_1, G_2, G_3, G_4, G_6, G_8\},\\ CoIt[9(t_1, t_2), 9(t_2, t_3)] = \{G_0, G_1, G_2, G_4, G_6, G_8\}.\end{cases} \tag{41}$$

Comparing to (21), we have

$$\begin{aligned}&\varDelta_{9(t_1,t_2),9(t_2,t_3)}\\ &= \tfrac{\mathrm{GCD}(CRT_9-CRT_9', CRT_9''-CRT_9')}{N_0N_1N_2N_4N_6N_8}\\ &= \mathrm{GCD}[Inv_{9,3} \times (\{k_9\}_{k_3} - \{k_9\}_{k_3'}),\\ &\qquad\quad N_3 \times TotalSum)].\end{aligned} \tag{42}$$

Similar to the deduction of (36), we have,

$$\Pr(2,2) < \mathrm{Prob}[\varDelta_{9(t_1,t_2),9(t_2,t_3)} = 1] < \Pr(1,2), \tag{43}$$

i.e. the probability is between 0.2178 and 0.4283.

We may further our explore on the structural relation between UGs and $CoIt$, and infer that $\mathrm{Prob}[\Delta_{5(t_1,t_2),6(t_1,t_2),9(t_1,t_2),9(t_2,t_3)} = 1]$ is in range $[0.1443, 0.3716]$. We also claim that the analysis of a subgroup's leaving is more or less the same as its joining, since these two moves are symmetric in terms of time. Details are omitted here.

We now see that $\mathrm{Prob}[\Delta = 1]$ indeed depends on the relations between UGs and $CoIt$. Different dynamic operations will result in different chances of success of our attack. Generally, the more the dynamic moves an adversary captures, the greater his chances of success are. The possibilities of success are too big to be ignored, which are exemplified by (36), (43) and etc.

A.3 Simulation of Attack

We simulate our attack in a similar way of our toy example, but set the security parameter to be 100 bits. We repeat our attack 10^5 times and then record the ratios of success. The results coincident with our analysis in A.2. The ratios are considered to be asymptotic to the real probabilities when the number of repetitions is large. We present them as follows.

$$\begin{cases} \mathrm{Prob}[\Delta_{5(t_1,t_2),6(t_1,t_2)} = 1] \approx 0.1531, \\ \mathrm{Prob}[\Delta_{5(t_1,t_2),5(t_1,t_2),9(t_1,t_2)} = 1] \approx 0.2522, \\ \mathrm{Prob}[\Delta_{9(t_1,t_2),9(t_2,t_3)} = 1] \approx 0.3802, \\ \mathrm{Prob}[\Delta_{5(t_1,t_2),6(t_1,t_2),9(t_1,t_2),9(t_2,t_3)} = 1] \approx 0.3534. \end{cases} \tag{44}$$

In A.2, our four ranges of them are: $[0.0911, 0.3371]$, $[0.1391, 0.3679]$, $[0.2178, 0.4283]$, $[0.1443, 0.3716]$.

We test a series of security parameters, from 10 to 250 bits, and find that it has no noticeable effect on the ratios of success. The number of repetitions also doesn't affect these ratios considerably. For example, in the test of 10 bits the length of security parameter, 10^7 the times of repetitions, the results are $(0.1599, 0.2587, 0.3842, 0.3540)$, which only show very little difference with (44).

Magma [9] programming language is adopted in simulations. Please feel free to contact authors for source codes.

Table 1. Common notations.

Notations	Meaning
$\{x\}_k$	Symmetric encryption under key k. x stands for one or several messages combined, different messages are separated by ",".
$E_k(x)$	public key encryption or signature under key k. x is defined in the same way above.
$\preceq$	Partial ordering.
G_i	Subgroup with identity i.
k_i	Data (symmetric) encryption key of G_i.
N_i	The modulus used by G_i.
n	The number of subgroups in the hierarchy.
η_i	The number of ancestors of subgroup G_i.
CRT_i	The solution of Chinese Remainder Theorem for G_i.
$\langle a \rangle$	It stands for an ideal generated by a in a ring.
$\mathrm{GCD}(a, b)$	The greatest common divisor of a and b.
$h()$	A cryptographic hash function.
$\|\|$	Concatenation of two messages.
$\|$	$a\|b$ represents that b is divisible by a. Similarly, $a \nmid b$ represents that b is not divisible by a.
$\Pr(i_1, \cdots, i_\nu)$	The probability of ν numbers, each of which is a product of i_j random numbers, to be coprime. $j \in \{1, \cdots, \nu\}$.

Table 2. $\{k_i\}_{k_j}$ in the toy example, Section 2.2.

	Initial									After k_3's change									Add G_{10}
$\{k_i\}_{k_j}$	$j = 1$	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	10
$i = 1$																			
2																			
3	596		306							478		593							
4		906		919							906		919						
5	4		449		976					4		401		976					
6	313	128	360	285		498				313	128	821	285		498				
7		800		307			581				800		307			581			
8																			
9	967	230	437	880		604		319	788	967	230	180	880		604		319	788	505

Table 3. CRT_i and their differences in the toy example.

Subgroup	CRT_i	CRT'_i	CRT''_i	$CRT'_i - CRT_i$	$CRT''_i - CRT'_i$
G_1	null	null	null	null	null
G_2	null	null	null	null	null
G_3	876285380	1455417430	1455417430	579132050	0
G_4	1390059561	1390059561	1390059561	0	0
G_5	4412627683684	4302143013220	4302143013220	-110484670464	0
G_6	5379838652695093273	5707051636970708473	5707051636970708473	327212984275615200	0
G_7	1617232194900	1617232194900	1617232194900	0	0
G_8	null	null	null	null	null
G_9	2331170126674	2034093739040	333783794992756	-297076387634	333580385618885
	9938357004855	3613086616895	28204966490055	6325270387960	224591879873160
G_{10}	null	null	null	null	null
note: The numbers of G_9 are too big, so we use two rows for each number.					

Table 4. New notations with time t.

Notations	Meaning
t	The temporal identity to identify a short period of time, in which no subgroup dynamic change takes place.
$CRT_i[t]$	The CRT_i of G_i computed by GC at time t.
$AG_i[t]$	The set of all determinant subgroups of G_i at time t, which may be ancestors or descendants or any other specific subgroups, depending on the scheme.
$\eta[t]_i$	The number of members in $AG_i[t]$, $\eta[t]_i = \|AG_i[t]\|$.
$G[t]_{i_l}$	A determinant subgroup of G_i at time t.
$N[t]_i$	The corresponding modulus of G_i at time t.
$I[t]_i$	The ideal of R corresponding to G_i at time t.
$\xi[t]_i$	The generator of an ideal $I[t]_i$ in a principal ideal domain.
$rem[t]_i$	The remainder class of $CRT_i[t]$ mod $N[t]_i$.
$UG_i[t_1, t_2]$	The set of unchanged determinant subgroups of G_i, from time t_1 to t_2, where $t_1 < t_2$.
$CoIt[a(t_1, t_2), b(t_3, t_4)]$	A conditional intersection. See Def. 2.
$\Delta_{a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})}$	It denotes $\frac{\mathrm{GCD}(CRT_{a_1}[t_2]-CRT_a[t_1],\cdots,CRT_{a_\mu}[t_{2\mu}]-CRT_{a_\mu}[t_{2\mu-1}])}{\prod_{G_j \in CoIt[a_1(t_1,t_2),\cdots,a_\mu(t_{2\mu-1},t_{2\mu})]} \xi[t_1]_j}$, see also (22).