Research Article

Analysis and Improvement of Key Distribution Scheme for Secure Group Communication

Jia Ning Luo¹ and Ming Hour Yang²

¹Department of Information and Telecommunication Engineering, Ming Chuan University, Taoyuan 333, Taiwan

²Department of Information and Computer Engineering, Chung Yuan Christian University, Taoyuan 320, Taiwan

Correspondence should be addressed to Jia Ning Luo; deer@mail.mcu.edu.tw

Received 16 November 2015; Accepted 2 March 2016

Academic Editor: Isao Echizen

Copyright © 2016 J. N. Luo and M. H. Yang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In a secure group communication, messages between a group coordinator and members are protected by session keys. If a group's membership changes, the session keys should be updated to insure forward secrecy and backward secrecy. Zhou and Huang proposed a key-updated scheme based on ciphertext-policy attribute encryption algorithm to improve the security of key-update mechanism, but their scheme is vulnerable: a malicious group member may send forged key-update messages to control the group. In this paper, we analyze the vulnerability in Zhou and Huang's scheme and propose an enhanced scheme. In our scheme, only the group initiator can update group keys and the verification of key-update mechanism is improved to prevent malicious insiders from controlling the group. We also give a security and performance analysis of our scheme.

1. Introduction

In recent years, social networks such as Google+, Facebook, and Twitter are receiving wide popularity and provide much convenience in people's daily life. In a social network, people can maintain their own social circles freely, such as adding or removing their friends and sharing messages to specific members within a group.

Social networks are based on cloud computing technology. In a cloud service, users' data and documents are not stored in their computers, but on cloud servers. In most cases, users' data is not encrypted and, therefore, is vulnerable to system vulnerabilities, unauthorized access, and privacy leak under government's censorship [1–3].

The secure multicast technique can be used to enhance users' privacy in social networks [4]. For secure multicast, the confidentiality of group communication is secured because group members share a session key to de/encrypt their communications. Once a group member leaves a group or a new member joins a group, their group key must be renewed. A group coordinator has to rekey the group. He needs to ensure that a new group key is delivered to every new member through a secure channel and that every old member's key is

updated simultaneously. By this, new members are not able to access previous messages, and those who have left a group cannot access the group's new messages. This guarantees the forward and backward secrecy in group communication and has made key updating an important issue for secure multicast.

Chang et al. [5] propose Flat Table (FT) for key management in a binary tree. A group coordinator only needs to store $\log N$ keys and each member can join $N - 1$ subgroups. To have higher flexibility of data encryption, Attribute-Based Encryption (ABE) [6–9] and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [10] are proposed. Bethencourt et al. [11] propose CP-ABE for multicast key management. It features complex access control on encrypted data. Zhou and Huang [12] combine FT and CP-ABE to present Efficient Group Keying (EGK). EGK is more efficient for a group coordinator to rekey his group, and the size of its ciphertext is fixed. No matter how its membership changes, the size of its ciphertext remains the same. EGK is also able to keep its storage within $O(\log N)$, where N denotes the maximum members. Huang et al. also come up with secure virtual trust routing and multicasting, aiming to apply EGK to networking

routing [13]. Jia et al. propose a layered EGK architecture [14]. Huang et al. proposed a cloud solution of EGK [15]. As each mobile device is seen as a service node, they propose to process mobile cloud data through trust management and private data isolation.

However, we have discovered certain security issues in the EGK model. For example, a malicious group member may launch the following attacks within EGK. (1) He may perform Man-in-the-Middle (MITM) attacks to intercept group communications and modify the group coordinator's messages. (2) He may masquerade as the coordinator and forge the key-update message and become the new coordinator. In Jia et al.'s EGK-based scheme [14], we have also found the threat of desynchronization because the group members cannot verify whether the messages are sent from the coordinator.

In this paper, we analyze the security issues in EGK and propose a new secure key management scheme. Our scheme resists the MITM attack and masquerade attack. The rest of this paper is organized as follows. In Section 2, we analyze the vulnerability of EGK. We describe our scheme in Section 3 and give a performance analysis in Section 4. A conclusion is drawn in Section 5.

2. EGK Scheme

Zhou and Huang [12] propose an Efficient Group Keying (EGK) method that is based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Following CP-ABE's policy, EGK takes each bit of group members' ID as an attribute. The group key is calculated by using the combination of specific group members.

When a new member joins a specific group, the coordinator generates a new group key and sends it to the new member by a secure channel and uses the old group key to encrypt the new one and sends it to the members. When a member leaves the group, the coordinator uses the Quine-McCluskey algorithm [16] to update the new group key.

The notations used in EGK are listed in Notations.

2.1. Initial Stage. At this stage, a group coordinator creates a group and the maximum membership of the group is N . The length of an ID is designed as $n = \log N$. The total number of L_B 's is $2n$, and the total number of P_B 's is $2n$ too. Consider $L_B = \{B_0, \overline{B_0}, B_1, \overline{B_1}, \dots, B_{n-1}, \overline{B_{n-1}}\}$; B_i denotes that the i th bit of ID is 1; $\overline{B_i}$ denotes that the i th bit of ID is 0. Consider $P_B = \{P_{B_0}, \overline{P_{B_0}}, P_{B_1}, \overline{P_{B_1}}, \dots, P_{B_{n-1}}, \overline{P_{B_{n-1}}}\}$; consider $P_{B_i}, \overline{P_{B_i}} \in Z_q$; P_B maps L_B , P_{B_0} maps B_0 , $\overline{P_{B_0}}$ maps $\overline{B_0}$, and so on. Then, the coordinator generates a master key $MK = \{\beta, g^\alpha, g^\beta, e(g, g)^\alpha, L_B, P_B\}$ and sets the parameters of a bilinear map function, such as q, G_0, G_1, g , and e . At last, he generates a hash function H and defines the public parameter $Params = \{G_0, e, g, H\}$.

2.2. Assign Private Key to Group Member. When a new member joins a group, the group coordinator gives to the member a unique ID and associates the member with an attribute A_{ID} . For example, there is a group whose maximum membership is 64, and the length of the ID is $\log_2 64$. The

coordinator assigns $ID = 001011$ to the new member and accordingly associates the member with the attribute $A_{ID} = \{\overline{B_5}, \overline{B_4}, B_3, \overline{B_2}, B_1, B_0\}$. Then, he runs the algorithm $KeyGen(MK, A_{ID})$ to generate a private key for the new member:

- (1) He generates a random value $\gamma \in Z_q^*$.
- (2) He calculates $D = g^{(\alpha+\gamma)/\beta}$.
- (3) According to A_{ID} , he calculates $\forall B_i \in A_{ID} : D_{B_i} = g^{\gamma B_i}$.
- (4) He performs the mapping of L_B and P_B , $\forall B_i \in A_{ID} : D_B = g^{\gamma P_{B_i}}$.
- (5) He generates a private key $SK_{ID} = \{D = g^{(\alpha+\gamma)/\beta}, \forall B_i \in A_{ID} : D_{B_i} = g^{\gamma P_{B_i}}\}$ for the new member.
- (6) At last, the coordinator sends ID, A_{ID}, SK_{ID} , and current session key SEK to the new member through a secure channel.

2.3. Encryption/Decryption Algorithms. When a group coordinator is going to multicast a message, he can use a different bit-assignment to assign specific members to decrypt the message. Then, he has to run the Quine-McCluskey algorithm to minimize all group members' IDs. The IDs are compared in pairs and reduced until there is no pair with only one bit different. The irreducible IDs are taken as attributes for encryption: S_i , where $1 \leq i \leq m$ and m denotes the number of irreducible IDs. After the coordinator runs the encryption algorithm for each attribute, he generates the encrypted messages for multicast: $ENC(Params, MK, S_1, M), ENC(Params, MK, S_2, M), \dots, ENC(Params, MK, S_m, M)$.

For example, a group coordinator is going to multicast three members whose IDs are 001001, 001011, and 111000, respectively. After minimization through Quine-McCluskey algorithm, the IDs are reduced as 111000 and 0010x1, where x denotes a negligible bit. Next, the coordinator maps the reduced IDs with L_B as attributes S_1 and S_2 . Consider $S_1 = \{B_0, \overline{B_2}, B_3, \overline{B_4}, \overline{B_5}\}$ and $S_2 = \{\overline{B_0}, \overline{B_1}, \overline{B_2}, B_3, B_4, B_5\}$.

With the attributes, the coordinator runs the encryption algorithm $ENC(Params, MK, S_i, M)$ in detailed steps as follows:

- (1) converting S_i 's L_B into P_B ,
- (2) calculating $Sum_{S_i} = \sum_{B_i \in S_i} P_{B_i}$,
- (3) generating a random value $t_{S_i} \in Z_q^*$,
- (4) calculating $C_{0S_i} = Me(g, g)^{\alpha t_{S_i} Sum_{S_i}}$,
- (5) calculating $C_{1S_i} = g^{\beta t_{S_i} Sum_{S_i}}$,
- (6) calculating $C_{2S_i} = g^{t_{S_i}}$,
- (7) generating a ciphertext $CT_{S_i} = \{S_i, C_{0S_i}, C_{1S_i}, C_{2S_i}\}$.

After group members receive the encrypted messages, they use their private key and the public parameter to run the decryption algorithm, $DEC(Params, SK, CT_{S_i})$:

- (1) The group member verifies whether CT_{S_i} 's S_i matches his own attribute A_{ID} . If the two match, the member

proceeds to the next step. If they do not match, the decryption algorithm ends.

- (2) He converts S_i 's L_B into P_B and calculates the sum: $F = \prod_{A_{ID} \in S_i} g^{Y_{P_{B_i}}} = g^{Y_{\sum_{A_{ID} \in S_i} P_{B_i}}} = g^{Y_{\text{Sum}_{S_i}}}$.
- (3) He calculates $A_1 = e(C_{1S_i}, D) = e(g^{\beta t_{S_i} \text{Sum}_{S_i}}, g^{(\alpha+\gamma)/\beta}) = e(g, g)^{(\alpha+\gamma)t_{S_i} \text{Sum}_{S_i}}$.
- (4) He calculates $A_2 = e(C_{2S_i}, F) = e(g^{t_{S_i}}, g^{Y_{\text{Sum}_{S_i}}}) = e(g, g)^{Y_{t_{S_i} \text{Sum}_{S_i}}}$.
- (5) He calculates $A_3 = A_1/A_2 = e(g, g)^{at_{S_i} \text{Sum}_{S_i}} \cdot C_{0S_i}/A_3 = Me(g, g)^{at_{S_i} \text{Sum}_{S_i}}/e(g, g)^{at_{S_i} \text{Sum}_{S_i}} = M$.

2.4. Key-Update Stage. When a new member joins a group, the group coordinator generates a new group key SEK' and uses the previous group key SEK to encrypt SEK' to the new member. Then, he sends ID, A_{ID} , SK_{ID} , and SEK to the new member through a secure channel. If a member leaves a group, the rekeying process will be divided into two parts: the group coordinator and the group member.

The group coordinator has to run the following steps:

- (1) Generating a random value $\alpha' \in Z_q^*$.
- (2) Updating his master key $\text{MK} = \{\beta, g^{\alpha'}, g^{\beta}, e(g, g)^{\alpha'}, L_B, P_B\}$.
- (3) Generating a rekeying message $\text{KU} = g^{(\alpha' - \alpha)/\beta}$.
- (4) Running Quine-McCluskey algorithm to minimize the remaining members' IDs until there are m irreducible IDs; these IDs are taken as attributes S_i , $1 \leq i \leq m$.
- (5) Using each attribute to run the encryption algorithm $\text{ENC}(\text{Params}, \text{MK}, S_i, M)$.

After receiving the ciphertext, the group members run the following steps:

- (1) Performing the decryption algorithm $\text{DEC}(\text{Params}, \text{SK}, \text{CT}_{S_i})$ and retrieving $\text{KU} = M$.
- (2) Multiplying KU by his private key's D and obtaining D' , $D' = \text{KU} \cdot D = g^{(\alpha' - \alpha)/\beta} \cdot g^{(\alpha + \gamma)/\beta} = g^{(\alpha' + \gamma)/\beta}$.
- (3) Updating his session key $\text{SEK}' = H_1(\text{KU})$.

3. Security Issues in EGK

3.1. Malicious Member and Man-in-the-Middle Attack. In EGK, a malicious member may decrypt the group coordinator's message and send the forged message to the members who are associated with the same attributes. The victim members may update their keys and consequently cause desynchronization with the coordinator. Thus, the group coordinator is replaced and the malicious member plays the man in the middle. He takes control of the group. Detailed steps are as follows.

A malicious member decrypts CT_{S_i} , a valid ciphertext of earlier sessions. He calculates F , A_1 , A_2 , and A_3 :

- (1) He retrieves $e(g, g)^{at_{S_i} \text{Sum}_{S_i}}$ from CT_{S_i} as A_3 ; that is, $A_3 = e(g, g)^{at_{S_i} \text{Sum}_{S_i}}$.
- (2) He generates a random value x and requires $\exists \alpha_2 \in Z_q^*$, so that $x = (\alpha_2 - \alpha)/\beta$; that is, $x\beta = \alpha_2 - \alpha$. He also forges a rekeying message $M' = g^x$.
- (3) He multiplies M' by A_3 and the result is $C'_0 = M'e(g, g)^{at_{S_i} \text{Sum}_{S_i}}$.
- (4) He sends a forged ciphertext $\text{CT}'_{S_i} : \{S_i, C'_{0S_i}, C_{1S_i}, C_{2S_i}\}$.

Therefore, the malicious member can bypass the encryption algorithm ENC and sends a forged ciphertext CT'_{S_i} without the group coordinator's master key MK . After receiving CT'_{S_i} , the members run the decryption algorithm DEC . Detailed steps are as follows:

- (1) Calculating $F = \prod_{A_{ID} \in S_i} g^{Y_{P_{B_i}}} = g^{Y_{\sum_{A_{ID} \in S_i} P_{B_i}}} = g^{Y_{\text{Sum}_{S_i}}}$.
- (2) Calculating $A_1 = e(C_{1S_i}, D) = e(g, g)^{(\alpha+\gamma)t_{S_i} \text{Sum}_{S_i}}$.
- (3) Calculating $A_2 = e(C_{2S_i}, F) = e(g, g)^{Y_{t_{S_i} \text{Sum}_{S_i}}}$.
- (4) Calculating $A_3 = A_1/A_2 = e(g, g)^{at_{S_i} \text{Sum}_{S_i}}$.
- (5) Calculating $C'_{0S_i}/A_3 = M'e(g, g)^{at_{S_i} \text{Sum}_{S_i}}/e(g, g)^{at_{S_i} \text{Sum}_{S_i}} = M'$.

After decryption, they retrieve the rekeying message $M' = g^{(\alpha_2 - \alpha')/\beta}$ and process to rekey themselves. They calculate $D' = D \cdot M' = g^{(\alpha_2 + \gamma)/\beta}$ and then calculate the current session key $\text{SEK}' = H(M')$.

Under such MITM attacks, the compromised members' private keys become $D = g^{(\alpha_2 + \gamma)/\beta}$, which is desynchronous with their group coordinator's $g^{\alpha'}$ in his master key MK . Here, we assume that the group coordinator sends a new rekeying message $\text{KU} = M = g^{(\alpha'' - \alpha')/\beta}$, uses Quine-McCluskey algorithm to minimize the members' IDs until there are n irreducible IDs and their attributes are S_i , $1 \leq i \leq n$, and multicasts the ciphertext to each member $\text{ENC}(\text{Params}, \text{MK}, S_1, M)$, $\text{ENC}(\text{Params}, \text{MK}, S_2, M), \dots, \text{ENC}(\text{Params}, \text{MK}, S_n, M)$. The compromised members receive their ciphertext and run the following steps for decryption:

- (1) Calculation of $F = \prod_{A_{ID} \in S_i} g^{Y_{P_{B_i}}} = g^{Y_{\sum_{A_{ID} \in S_i} P_{B_i}}} = g^{Y_{\text{Sum}_{S_i}}}$.
- (2) Calculation of $A_1 = e(C_{1S_i}, D) = e(g, g)^{(\alpha_2 + \gamma)t_{S_i} \text{Sum}_{S_i}}$.
- (3) Calculation of $A_2 = e(C_{2S_i}, F) = e(g, g)^{Y_{t_{S_i} \text{Sum}_{S_i}}}$.
- (4) Calculation of $A_3 = A_1/A_2 = e(g, g)^{\alpha_2 t_{S_i} \text{Sum}_{S_i}}$.
- (5) Calculation of $C_{0S_i}/A_3 = Me(g, g)^{\alpha' t_{S_i} \text{Sum}_{S_i}}/e(g, g)^{\alpha_2 t_{S_i} \text{Sum}_{S_i}}$.

After the division in step (5), the compromised members are unable to retrieve the key-update message M . However, the malicious member can repeat the same rekeying process

and successfully update the group's private keys and session key. Then, the malicious member is able to replace the coordinator and take control of the group.

3.2. Full Modification of Group Coordinator's Ciphertext. A malicious member may modify the whole ciphertext, except S_i , in the following steps:

- (1) Calculating $B_{1S_i} = e(C_{1S_i}, M) = e(g^{\beta t_{S_i} \text{Sum}_{S_i}}, g^{(\alpha' - \alpha)/\beta}) = e(g, g)^{(\alpha' - \alpha)t_{S_i} \text{Sum}_{S_i}}$.
- (2) Calculating $B_{2S_i} = C_{0S_i} \cdot B_1/M = e(g, g)^{\alpha t_{S_i} \text{Sum}_{S_i}} \cdot e(g, g)^{(\alpha' - \alpha)t_{S_i} \text{Sum}_{S_i}} = e(g, g)^{\alpha' t_{S_i} \text{Sum}_{S_i}}$.
- (3) Calculating $B_{3S_i} = (B_{2S_i})^x = e(g, g)^{\alpha' t_{S_i} \text{Sum}_{S_i} \cdot ((\alpha_2 - \alpha')/\beta)} = e(g, g)^{((\alpha_2 - \alpha')\alpha' t_{S_i} \text{Sum}_{S_i})/\beta}$.
- (4) Multiplying M' by B_{3S_i} to forge $C'_{0S_i} = M' \cdot B_{3S_i} = M' e(g, g)^{((\alpha_2 - \alpha')\alpha' t_{S_i} \text{Sum}_{S_i})/\beta}$.
- (5) Calculating $C'_{1S_i} = (C_{1S_i})^x = (C_{1S_i})^{(\alpha_2 - \alpha')/\beta} = g^{(\alpha_2 - \alpha')t_{S_i} \text{Sum}_{S_i}}$.
- (6) Calculating $C'_{2S_i} = (C_{2S_i})^x = (C_{2S_i})^{(\alpha_2 - \alpha')/\beta} = g^{((\alpha_2 - \alpha')t_{S_i})/\beta}$.
- (7) Sending the forged message $\text{CT}'_{S_i} = \{S_i, C'_{0S_i}, C'_{1S_i}, C'_{2S_i}\}$.

After receiving CT'_{S_i} , the group members run the decryption algorithm $\text{DEC}(\text{Params}, \text{SK}, \text{CT}'_{S_i})$:

- (1) The group members verify whether the received S_i matches their own A_{ID} . If the two are verified, they proceed to the next step. Otherwise, they end the algorithm.
- (2) They convert the S_i -associated L_B into P_B and then calculate the sum $F = \prod_{A_{\text{ID}} \in S_i} g^{\gamma P_{B_i}} = g^{\gamma \sum_{A_{\text{ID}} \in S_i} P_{B_i}} = g^{\gamma \text{Sum}_{S_i}}$.
- (3) They calculate $F = \prod_{A_{\text{ID}} \in S_i} g^{\gamma P_{B_i}} = g^{\gamma \sum_{A_{\text{ID}} \in S_i} P_{B_i}} = g^{\gamma \text{Sum}_{S_i}}$.
- (4) They calculate $A_1 = e(C'_{1S_i}, D) = e(g^{(\alpha_2 - \alpha')t_{S_i} \text{Sum}_{S_i}}, g^{(\alpha' + \gamma)/\beta}) = e(g, g)^{((\alpha_2 - \alpha')(\alpha' + \gamma)t_{S_i} \text{Sum}_{S_i})/\beta}$.
- (5) They calculate $A_2 = e(C'_{2S_i}, F) = e(g^{((\alpha_2 - \alpha')t_{S_i})/\beta}, g^{\gamma \text{Sum}_{S_i}}) = e(g, g)^{((\alpha_2 - \alpha')\gamma t_{S_i} \text{Sum}_{S_i})/\beta}$.
- (6) They calculate $A_3 = A_1/A_2 = e(g, g)^{((\alpha_2 - \alpha')(\alpha' + \gamma)t_{S_i} \text{Sum}_{S_i})/\beta} / e(g, g)^{((\alpha_2 - \alpha')\gamma t_{S_i} \text{Sum}_{S_i})/\beta} = e(g, g)^{((\alpha_2 - \alpha')(\alpha' + \gamma)t_{S_i} \text{Sum}_{S_i} - (\alpha_2 - \alpha')\gamma t_{S_i} \text{Sum}_{S_i})/\beta} = e(g, g)^{((\alpha_2 - \alpha')\alpha' t_{S_i} \text{Sum}_{S_i})/\beta}$.
- (7) They calculate $C'_{0S_i}/A_3 = M' e(g, g)^{((\alpha_2 - \alpha')\alpha' t_{S_i} \text{Sum}_{S_i})/\beta} / e(g, g)^{((\alpha_2 - \alpha')\alpha' t_{S_i} \text{Sum}_{S_i})/\beta} = M'$.

After the decryption procedure, they retrieve $M' = g^{(\alpha_2 - \alpha')/\beta}$, calculate $D' = D \cdot M' = g^{(\alpha_2 + \gamma)/\beta}$, and update their current session key $\text{SEK}' = H(M')$.

In EGK, even though a malicious member is unable to decrypt his group coordinator's ciphertext, he can forge a message and then try the steps of Sections 3.1 and 3.2. Maybe some members are able to decrypt two ciphertexts, for example, S_i and S_j , but they can still receive the forged message. Since they do not verify the sender, they just retrieve M' and update their keys and hence the desynchronization between them and their group coordinator.

4. Secure Key Distribution Scheme for Multicast Communication

4.1. Initial Stage. At this stage a group coordinator creates a multicast group with N maximum members, generates system parameters G_0, G_1, q, g , and e , defines two hash functions H_1 and H_2 , generates three random values α, β and t , requires $\alpha, \beta, t \in Z_q^*$, sets the length of ID as $n = \log N$, creates $2\log_2 N$ bit-assignment sets $L_B = \{B_0, \overline{B_0}, B_1, \overline{B_1}, \dots, B_{n-1}, \overline{B_{n-1}}\}$, where B_i denotes that the i th bit of ID is 1 and $\overline{B_i}$ denotes that the i th bit of ID is 0, and creates sets of bit-assignment secrets $P_B = \{p_{B_0}, \overline{p_{B_0}}, p_{B_1}, \overline{p_{B_1}}, \dots, p_{B_{n-1}}, \overline{p_{B_{n-1}}}\}$, where $p_{B_i}, \overline{p_{B_i}} \in Z_q$. P_B maps L_B . That is, p_{B_0} maps B_0 , $\overline{p_{B_0}}$ maps $\overline{B_0}$, and so on. Further, a group coordinator generates his master key $\text{MK} = \{\beta, t, g^\alpha, g^\beta, e(g, g)^\alpha, L_B, P_B\}$ and the public parameter $\text{Params} = \{G_0, e, g, P_{\text{pub}}, H_1, H_2\}$, where $P_{\text{pub}} = g^t$.

4.2. Key Distribution. When a new member joins the group, the group coordinator assigns a unique ID to the member and associates him with an attribute A_{ID} . Then, the group coordinator runs $\text{KeyGen}(\text{MK}, A_{\text{ID}})$ to generate a key:

- (1) He generates a random value $\gamma \in Z_q^*$.
- (2) He calculates $D = g^{(\alpha + \gamma)/\beta}$.
- (3) He uses A_{ID} to compute $\forall B_i \in A_{\text{ID}} : D_B = g^{\gamma B_i}$.
- (4) He maps P_B with $L_B, \forall B_i \in A_{\text{ID}} = g^{\gamma P_{B_i}}$.
- (5) He generates a private key $\text{SK}_{\text{ID}} = \{D = g^{(\alpha + \gamma)/\beta}, \forall B_i \in A_{\text{ID}} : D_B = g^{\gamma P_{B_i}}\}$.

At last, the group coordinator multicasts ID, A_{ID} , SK_{ID} , and current session key SEK to the new member through a secure channel.

4.3. Encryption and Decryption Algorithm. When a group coordinator needs to multicast his group, he assigns a unique set of bit-assignment to each group member, so that each member is able to decrypt their ciphertext. Also, the coordinator uses the Quine-McCluskey algorithm to minimize remaining members' IDs and then calculates attributes S_i . He runs ENC to encrypt S_i and members run DEC to decrypt it.

The group coordinator runs $\text{ENC}(\text{Params}, \text{MK}, S_i, M)$ that

- (1) calculates $h = H_2(M)$,
- (2) converts the S_i -associated L_B into P_B ,
- (3) calculates $\text{Sum}_{S_i} = \sum_{B_i \in S_i} p_{B_i}$,

- (4) calculates $C_{0S_i} = Me(h, g)^{\alpha t \text{Sum}_{S_i}}$,
- (5) calculates $C_{1S_i} = h^{\beta t \text{Sum}_{S_i}}$,
- (6) calculates $C_{2S_i} = h^t$,
- (7) generates $CT_{S_i} = \{S_i, C_{0S_i}, C_{1S_i}, C_{2S_i}\}$.

After receiving the ciphertext, group members verify whether their own A_{ID} matches S_i 's bit-assignment. If verified, they run the decryption algorithm $DEC(\text{Params}, SK, CT_{S_i})$. Besides, during decryption, they have to check if the sender of M is their group coordinator. Steps are as follows:

- (1) Conversion of the S_i -associated L_B into P_B .
- (2) Calculating $F = \prod_{A_{ID} \in S_i} g^{yP_{B_i}} = g^{y \sum_{A_{ID} \in S_i} P_{B_i}} = g^{y \text{Sum}_{S_i}}$.
- (3) Calculating $A_1 = e(C_{1S_i}, D) = e(h, g)^{(\alpha+\gamma)t \text{Sum}_{S_i}}$.
- (4) Calculating $A_2 = e(C_{2S_i}, F) = e(h, g)^{y t \text{Sum}_{S_i}}$.
- (5) Calculating $A_3 = A_1/A_2 = e(h, g)^{\alpha t \text{Sum}_{S_i}}$.
- (6) Dividing C_{0S_i} by A_3 and the result is $M = C_{0S_i}/A_3$.
- (7) Calculating $h = H_2(M)$.
- (8) Verifying if $e(h, P_{\text{pub}}) = e(h, g^t) = e(C_{2S_i}, g)$.

4.4. Key-Update Stage. If a new member joins a group, the group coordinator generates a new session key SEK' and uses the previous session key SEK to encrypt it to his group members. Also, through a secure channel, the coordinator multicasts ID, AID, SK_{ID} , and SEK to the members. If a member leaves a group, the coordinator and his members proceed to different steps.

Group coordinator performs the following:

- (1) He generates a random value $\alpha' \in Z_q^*$.
- (2) He updates his master key $MK = \{\beta, g^{\alpha'}, g^{\beta}, e(g, g)^{\alpha'}, L_B, P_B\}$.
- (3) He generates a rekeying message $KU = g^{(\alpha'-\alpha)/\beta}$.
- (4) He uses the Quine-McCluskey algorithm to minimize remaining members' IDs until there are m irreducible IDs and these IDs are taken as attributes S_i , $1 \leq i \leq m$.
- (5) He runs $ENC(\text{Params}, MK, S_i, M)$ for each attribute.

Group members perform the following:

- (1) After receiving the ciphertext, each group member runs $DEC(\text{Params}, SK, CT_{S_i})$ and retrieves $M = KU$.
- (2) They multiply KU by D and the result is $D' = KU \cdot D = g^{(\alpha'-\alpha)/\beta} \cdot g^{(\alpha+\gamma)/\beta} = g^{(\alpha'+\gamma)/\beta}$.
- (3) They update their session key $SEK' = H_1(KU)$.

5. Security Analysis

5.1. Malicious Member and MITM Attack. We assume a malicious member is in a group coordinator's communication group. He receives the latest rekeying message and then forges a fake one to other group members. The forged message may arrive before the group coordinator's rekeying message and successfully rekey the members. By this, the malicious member can cause asynchrony between group members and their coordinator.

After receiving group coordinator's CT_{S_i} , the malicious member retrieves S_i , C_{0S_i} , C_{1S_i} , and C_{2S_i} and calculates A_1 , A_2 , and A_3 . He tries to forge a message $C'_{0S_i} = M' \cdot A_3$ and sends the fake ciphertext $CT'_{S_i} = \{S_i, C'_{0S_i}, C_{1S_i}, C_{2S_i}\}$ to other members. Following the decryption algorithm, the members decrypt the fake ciphertext CT'_{S_i} and retrieve M' . According to the 7th step of decryption, they calculate $h = H_2(M')$. At the 8th step, they calculate $e(h, P_{\text{pub}}) = e(h, g^t)$, which does not match $e(C_{2S_i}, g)$, hence an unsuccessful attack.

5.2. Malicious Member and Asynchrony. We assume a malicious member belongs to a group coordinator's communication group. He is able to decrypt and retrieve the coordinator's rekeying message. He forges a rekeying message and sends it to other group members. CT_{S_i} is the previous ciphertext from the group coordinator:

$$CT_{S_i} = \{S_i, C_{0S_i} = Me(h, g)^{\alpha t \text{Sum}_{S_i}}, C_{1S_i} = h^{\beta t \text{Sum}_{S_i}}, C_{2S_i} = h^t\}, \quad (1)$$

$$M = KU = g^{(\alpha'+\alpha)/\beta}.$$

With CT_{S_i} and M , the group members have updated their private keys $D = g^{(\alpha'+\gamma)/\beta}$. The malicious member forges a rekeying message M' . Since he has had H_2 , he uses it to calculate $H_2(M')$ and tries to generate $C'_{0S_i} = M' e(H_2(M'), g)^{\alpha' t \text{Sum}_{S_i}}$. He has to use CT_{S_i} to run the following steps:

- (1) Summing up the previous attributes S_i ; thus, $F = \prod_{A_{ID} \in S_i} g^{yP_B} = g^{y \sum_{A_{ID} \in S_i} P_B} = g^{y \text{Sum}_{S_i}}$.
- (2) Calculating $A'_1 = e(C_{1S_i}, D) = e(H_2(M), g)^{(\alpha'+\gamma)t \text{Sum}_{S_i}}$.
- (3) Calculating $A'_2 = e(C_{2S_i}, F) = e(H_2(M), g)^{y t \text{Sum}_{S_i}}$.
- (4) Calculating $A'_3 = A'_1/A'_2 = e(H_2(M), g)^{\alpha' t \text{Sum}_{S_i}}$.

Since $A'_3 = e(H_2(M), g)^{\alpha' t \text{Sum}_{S_i}}$, it does not match $e(H_2(M'), g)^{\alpha' t \text{Sum}_{S_i}}$.

5.3. Backward Secrecy. If a group coordinator removes a member from the group, he updates his own master key and runs the encryption algorithm ENC to generate a rekeying message KU to the remaining members. After receiving KU , the members update their private keys $SK \cdot D' = SK \cdot D \cdot KU$

TABLE 1: Comparison of computational load.

	EGK		Our scheme	
	Encryption	Decryption	Encryption	Decryption
Coordinator	$(e + 3Ep)m$	0	$(h + e + 3Ep)m$	0
Member	0	$3e$	0	$h + 4e$

TABLE 2: Storage.

	Master key	Public parameter	Session key
Coordinator	1	0	1
Member	0	N	N
Total	1	N	$N + 1$

and calculate a new session key $SEK' = H_2(KU)$. Because the removed member does not have KU , he cannot calculate a new private key and session key.

5.4. Forward Secrecy. When a new member joins a group, the group coordinator generates a new session key SEK' and uses the previous session key SEK to encrypt SEK' to his group members. Then, he sends SEK' to the new member through a secure channel. In doing so, the new member is unable to access previous messages.

6. Performance Evaluation

In our secure multicast group communication protocol, a group coordinator has to store a set of a public parameter, a master key, and a session key, that is, $\{\text{Params}, \text{MK}, \text{SEK}\}$. It consists of five real numbers, two hash functions, one mapping function, and two sets of bit-assignments.

As for our computational load, a group coordinator calculates one hash function, one mapping, and three exponentials when running the encryption algorithm. When a group member decrypts a ciphertext, the decryption algorithm requires four mappings and one hash. As listed in Table 1, if there are m attributes after the minimization, our key-updating requires m times of encryption, that is, m hash functions, m mappings, and $3m$ exponentials.

Compared with EGK, our encryption algorithm requires sender verification; therefore, it needs to compute one more hash. And our decryption algorithm requires group members to verify messages, so it needs to compute one more hash and one more mapping.

Take Facebook as an example. We assume that it uses our scheme; its member limit is 5000 ($N = 5000$); each user is a group coordinator; each user is also a member of his friends' groups. Thus, each user has a master key, $(N + 1)$ public parameters, and $(N + 1)$ session keys, as shown in Table 2.

In our scheme, a group coordinator's communication load is in linear increase with the minimized attributes. If there are m attributes after minimization, he has to generate m rekeying messages. Each message consists of one S_i and three variables C_{0S_i} , C_{1S_i} , and C_{2S_i} (see Table 3).

Despite having one more hash and one more public parameter compared to EGK, our system's storage

TABLE 3: Communication loads for rekeying.

	EGK	Our scheme
	$(S_i, C_{0S_i}, C_{1S_i}, C_{2S_i})m$	$(S_i, C_{0S_i}, C_{1S_i}, C_{2S_i})m$
Coordinator		

requirement is the same as EGK's. And the two schemes' communication loads are also the same.

7. Conclusion

In this paper, we analyze Zhou and Huang's key distribution scheme for multicast group communication and find it prone to MITM and desynchronization attacks. These security issues can cause a group coordinator to lose the control over his group and fail to rekey his members. We propose a scheme to enhance the security of EGK. Our scheme guarantees forward and backward secrecy, prevents message modification and forgery during rekeying, requires sender verification, and therefore prevents MITM attacks in group communication.

We hope our secure multicast group communication scheme can be applied to cloud services and social networking sites, so that users can enjoy these services in a secure environment without further loads. And users' privacy and data integrity can also be secured.

Notations

q :	A large prime number
G_0 :	A cyclic additive group of prime order q and generator g
G_1 :	A cyclic multiplicative group of prime order q
g :	Generator, $g \in G_0$
e :	A bilinear map $e, G_0 \times G_0 \rightarrow G_1$
H :	One-way hash function
α, β, γ :	Random values, $\alpha, \beta, \gamma \in Z_q^*$
ID:	Identifier of group members, $ID = b_n b_{n-1} \dots b_0$
L_B :	A set of bit-assignments $L_B = \{B_0, \overline{B_0}, B_1, \overline{B_1}, \dots, B_{n-1}, \overline{B_{n-1}}\}$; B_i : the i th bit of ID is 1; $\overline{B_i}$: the i th bit of ID is 0.
P_B :	A set of bit-assignments secrets $P_B = \{p_{B_0}, \overline{p_{B_0}}, p_{B_1}, \overline{p_{B_1}}, \dots, p_{B_{n-1}}, \overline{p_{B_{n-1}}}\}$; $p_{B_i}, \overline{p_{B_i}} \in Z_q$; P_B maps L_B
Params:	Public parameter Params = $\{G_0, e, g, H\}$
MK:	Group coordinator's master key $MK = \{\beta, g^\alpha, g^\beta, e(g, g)^\alpha, L_B, S_B\}$
SK_{ID} :	Private keys that a group coordinator multicasts to members of specific ID, $SK_{ID} = \{D = g^{(\alpha+\gamma)/\beta}, \forall B_i \in A_{ID} : D_{B_i} = g^{r_{S_{B_i}}}\}$
SEK:	Session key
A_{ID} :	An attribute that a group coordinator gives to a member of a specific ID (each attribute is the bit-assignment of the ID).

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the National Science Council of Taiwan under Grant no. MOST104-2221-E-130-009.

References

- [1] X. Liang, K. Zhang, X. Shen, and X. Lin, "Security and privacy in mobile social networks: challenges and solutions," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 33–41, 2014.
- [2] L. A. Cuttillo, R. Molva, and T. Strufe, "Safebook: a privacy-preserving online social network leveraging on real-life trust," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 94–101, 2009.
- [3] H. Abelson, R. Anderson, S. M. Bellovin et al., "Keys under doormats: mandating insecurity by requiring government access to all data and communications," *Journal of Cybersecurity*, 2015.
- [4] S. Deering, "Host extensions for IP multicasting," *RFC 1112*, IETF, August 1989.
- [5] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure Internet multicast using Boolean function minimization techniques," in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM '99)*, pp. 689–698, Sharm El Sheik, Egypt, March 1999.
- [6] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.
- [7] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.
- [8] Z. Wan, J. Liu, and R. H. Deng, "HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2012.
- [9] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.
- [10] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76–88, 2013.
- [11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, Calif, USA, 2007.
- [12] Z. Zhou and D. Huang, "An optimal key distribution scheme for secure multicast group communication," in *Proceedings of the IEEE INFOCOM*, pp. 1–5, San Diego, Calif, USA, March 2010.
- [13] D. Huang, S. Ata, and D. Medhi, "Establishing secure virtual trust routing and provisioning domains for future internet," in *Proceedings of the 53rd IEEE Global Communications Conference*, pp. 1–6, IEEE, Miami, Fla, USA, December 2010.
- [14] H. Jia, Y. Chen, X. Mao, and R. Dou, "Efficient and scalable multicast key management using attribute based encryption," in *Proceedings of the IEEE International Conference on Information Theory and Information Security (ICITIS '10)*, pp. 426–429, Beijing, China, December 2010.
- [15] D. Huang, Z. Zhou, L. Xu, T. Xing, and Y. Zhong, "Secure data processing framework for mobile cloud computing," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM '11)*, pp. 614–618, Shanghai, China, April 2011.
- [16] E. McCluskey, "Minimization of Boolean functions," *The Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956.