International Journal of Innovative Computing, Information and Control
Volume **9**, Number **5**, May **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 1935–1949

A PRACTICAL SECURE CHAOS-BASED GROUP KEY AGREEMENT PROTOCOL SUITABLE FOR DISTRIBUTED NETWORK ENVIRONMENT

Zi-Yao Cheng[1], Yun Liu[1,*], Chin-Chen Chang[2,3] and Shih-Chang Chang[4]

[1]Department of Electronic and Information Engineering
Key Laboratory of Communication and Information Systems
Beijing Municipal Commission of Education
Beijing Jiaotong University
No. 3, Shang Yuan Cun, Hai Dian District, Beijing 100044, P. R. China
{ 09111024; 09111001 }@bjtu.edu.cn; *Corresponding author: liuyun@bjtu.edu.cn

[2]Department of Information Engineering and Computer Science
Feng Chia University
No. 100, Wenhwa Rd., Seatwen, Taichung 40724, Taiwan
alan3c@gmail.com

[3]Department of Computer Science and Information Engineering
Asia University
No. 500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan

[4]Department of Computer Science and Information Engineering
National Chung Cheng University
No. 168, University Rd., Minhsiung Township, Chiayi County 62102, Taiwan
chang.coby@gmail.com

Received February 2012; revised June 2012

Abstract. *Group key agreement is an important security mechanism for distributed communications, in which it can provide a specific environment for people located at different places to contribute each one's secret to establish a final session key over an insecure channel. Recently, Guo and Zhang proposed a new protocol that satisfied the requirements of a group key agreement in terms of a chaotic cryptosystem. However, Guo and Zhang's protocol cannot comply with the mechanism for group key agreement even if it has a contributory nature inside, and it cannot protect the security when off-line guessing attacks occur. To overcome these drawbacks, we propose a novel, practical protocol that meets the realistic requirements of group key agreement based on Chebyshev chaotic maps and resists several types of attacks. We believe that our proposed protocol is efficient, secure, and suitable for distributed networks for collaborative group communications.*

Keywords: Group key agreement, Chaos, Chebyshev, Security

1. Introduction.

In the last few years, the security and privacy designs have made advances that the distributed communications such as distributed simulations, multi-media conferences and decentralized time applications service for people to get various information conveniently and quickly in any place. According to the requirements of practical cryptographic mechanisms on distributed multi-case communications, secure infrastructures for collaborative groups have been widely studied. In general, protocols for the establishment of group keys have been developed so that groups of participants can negotiate a common secret key, called a group key, thereafter allowing the group members to communicate with each other securely over an open network. The protocol for establishing the group key consists of group key distribution [1-4] and group key agreement [5-10].

In group key distribution, a key distributor, who must be a trusted party, establishes a specific key for the use of all involved participants and distributes it to them with secure protection. However, in the group key agreement protocol, each participant in the group contributes to the establishment of the secret key for the session. Although the group key distribution protocol is inherently simple, the group key agreement protocol has the advantage of contributory property, which means that there is no need for the existence of a trusted distributor because the participants can execute processes over an insecure channel, and none of the group members can predetermine the secret information of the group key.

In recent years, cryptosystems based on chaos theory have been discussed extensively [11-17]. A chaotic cryptosystem is suitable for deriving a common secret key from the collaborative contributions of two or more parties, but none of the parties can predict the resulting value because the protocol for a chaotic cryptosystem is characterized by sensitive dependence on initial conditions and designing secure communications. In 2005, Xiao et al. [11] proposed a specific, entire-denial, authentication scheme based on a chaotic system that utilized a chaotic public key cryptosystem. However, it was insecure against Bergamo et al.'s developed attack [12] and the man-in-the-middle attack [13]. Consequently, enhancements for dealing with the aforementioned drawbacks have been presented; Xiao et al. [14] proposed a novel, chaos-based key agreement that can improve the security level. Later, in 2008, Han [15] discovered that Xiao et al.'s protocol was still vulnerable to the replay attack. Accordingly, several research efforts [15-17] were developed to investigate the use of effective time stamps or nonces.

Although the previous works [15-17] had attractive attributes without either a synchronization clock or replaying attempts, it was claimed that those protocols had difficulty achieving the contributory property of key agreement. More specifically, Guo and Zhang [18] first presented a server spoofing attack and a denial-of-service attack on Xiao et al.'s protocol and analyzed the situation for reasons. With the goal of conquering the weaknesses that result from security flaws and the hiatus of the contributory nature, Guo and Zhang recently proposed a group key agreement protocol based on a chaotic hash function [18]. By exploiting the chaotic cryptosystem, the proposed protocol has advantages in the contributory property, and it also resists obvious attacks. Unfortunately, we can illustrate that Guo and Zhang's protocol is still susceptible to the off-line password guessing attack. Based on our cryptanalysis in Subsection 3.2, their protocol cannot actually satisfy the standard that requires a realistic group key agreement achievement.

In this article, we propose a novel protocol that consists of the actual requirements of the group key agreement and efficient Chebyshev chaotic maps. Compared with Guo and Zhang's protocol, our proposed protocol has several advantages in both functionality and security requirements. To describe our contributions in detail, we demonstrate the characteristic features as follows:

- Functionality concerns. We provide evidence that our proposed protocol offers the contributory nature of group key agreement as does Guo and Zhang's protocol. In addition, we show that mutual authentication can be achieved in our realistic group communication without the use of a synchronized clock. Also, our protocol has a dynamic setting in which individual group participants can leave or join the process arbitrarily, depending on personal needs or preferences.
- Security requirements. It can be proven that the proposed group key agreement protocol can successfully withstand several kinds of attacks, such as impersonation attacks, replay attacks, off-line password guessing attacks, and denial-of-service attacks. The security requirements of our protocol ensure forward secrecy in that the

derivation of the group key caused by joining the process does not result in compromising the previously established secret key information. Moreover, it should protect the property of backward secrecy, which is a phenomenon in which a participant can leave the group without compromising private group keys in the future.

- Practical framework for multi-party use. Each legal participant in our proposed group key agreement protocol is associated with the tree-based structure, which is organized for all members of the group to perform key computations. A detailed introduction is presented in Section 4.

The rest of this article is described below. In the next section, we briefly introduce the Chebyshev chaotic map and the specific enhanced version utilized in our protocol. In Section 3, we review Guo and Zhang's protocol step by step and provide our cryptanalysis of the drawbacks of their protocol. We propose our group key agreement protocol in Section 4, and the security analysis and comprehensive discussion are presented in Sections 5 and 6, respectively. Our conclusions are given in Section 7.

2. Preliminaries.

In this section, we will present the description of Chebyshev chaotic map, in which it contains several main definitions to obtain the semi-group property.

2.1. Normal Chebyshev polynomial.

As we know, Chebyshev polynomials of degree n can be defined as:

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x), \tag{1}$$

where the integer $n \geq 2$, $T_0(x) = 1$ and $T_1(x) = x$.

So the rest of the Chebyshev polynomials are satisfied:

$$T_2(x) = 2x^2 - 1, \quad T_3(x) = 4x^3 - 3x, \ \ldots, \ T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x).$$

If the internal of x is restricted to $[-1, 1]$, the corresponding Chebyshev polynomial $T_n(x)$ belongs to $[-1, 1]$ which is a standard chaotic map $T_n(x) : [-1, 1] \to [-1, 1]$. It can be easily defined that

$$T_n(x) = \cos(n \cdot \arccos(x)). \tag{2}$$

By virtue of trigonometric and cosine attributes, the eligible, semi-group property [19] can be achieved as follows:

$$T_r(T_s(x)) = T_{r \cdot s}(x). \tag{3}$$

An immediate derivation can make the Chebyshev polynomials commute under composition. It implies that

$$T_r(T_s(x)) = T_s(T_r(x)). \tag{4}$$

2.2. Enhanced Chebyshev polynomial.

Due to the periodicity of the cosine function, encryption schemes that use the normal Chebyshev chaotic map are always insecure. To fill the gaps, Zhang [20] enhanced the Chebyshev chaotic map and proved that it can still obtain the semi-group property; even the internal of x is defined for $(-\infty, +\infty)$. To avoid the weakness of cosine periodicity, we utilize the enhanced Chebyshev polynomials in our protocol.

The semi-group property holds for Chebyshev polynomials such as:

$$T_n(x) = (2xT_{n-1}(x) - T_{n-2}(x))(\text{mod} N), \tag{5}$$

for any integer $n \geq 2$, where N is a large prime number and x represents variable values over the internal $(-\infty, +\infty)$.

Consequently, the enhanced Chebyshev polynomials also can achieve the equivalent properties of the normal examples.

$$T_r(T_s(x)) = T_{r \cdot s}(x) = T_s(T_r(x)). \tag{6}$$

3. **Review of Guo and Zhang's Group Key Agreement Protocol.**

In this section, we will illustrate Guo and Zhang's group key agreement protocol [18]. It is claimed that they utilized the semi-group property of the Chebyshev chaotic map and the one-way property of the chaotic hash function [21] to satisfy security concerns, thereby overcoming the vulnerabilities of Xiao et al.'s scheme and achieving the contributory property of the group key agreement protocol. After the protocol is reviewed below, we will demonstrate our related cryptanalysis of Guo and Zhang's protocol.

3.1. **Guo and Zhang's group key agreement protocol.**

Guo and Zhang's protocol assumes that user A and server B share the A's password value $h_{PW} = H(ID_A, PW_A)$, where ID_A is A's identity and $H(\cdot)$ is one-way chaotic hash function.

3.1.1. *Authentication phase.*

Step 1: User A chooses a random number $ra \in [-1, 1]$, then sends the authentication message $AU_A = \{ID_A, ra, H(h_{PW}, ra)\}$ to server B, where A juxtaposes ID_A, ra and $H(h_{PW}, ra)$ from left to right.

Step 2: After receiving the authentication message AU_A from user A, the server B can verify the validity of A by means of her or his own h_{PW} and received ra to compute $H'(h_{PW}, ra)$ and then checks whether the value of $H'(h_{PW}, ra)$ is equal to the received $H(h_{PW}, ra)$. If so, the validity of A is authenticated, and B sends a message $AU_B = \{rb, H(h_{PW}, ra, rb)\}$ back to A, where rb is a random number selected by B; otherwise, B rejects the authentication request.

Step 3: As the user A receiving the message AU_B, he or she verifies the validity of B by using her or his own h_{PW} and ra to compute $H'(h_{PW}, ra, rb)$ and then checks whether the value of $H'(h_{PW}, ra, rb)$ is equal to the received $H(h_{PW}, ra, rb)$. If so, the server B is validated and mutual authentication is done. A continues to calculate an acknowledgement message $ACK = H(rb \oplus h_{PW})$ and sends it to B.

Step 4: After receiving the message $ACK = H(rb \oplus h_{PW})$, the server B computes $ACK' = H(rb \oplus h_{PW})$ and compares it with the received acknowledgement message. If the two values are same, B can prove that the acknowledgement message was truly sent by A.

3.1.2. *Key agreement phase.*

Step 1: The user A continues to transmit message T_1 to server B, where $T_1 = H(h_{PW}) \oplus H(T_r(ra))$, $T_r(ra)$ is the Chebyshev polynomial of degree r, and r is chosen randomly by user A.

Step 2: After the message T_1 is received, B derives $X = T_1 \oplus H(h_{PW}) = H(T_r(ra))$ and responds with message $T_2 = \{H(H(h_{PW}) \oplus X) \oplus T_s(ra), H(T_s(ra))\}$, where $T_s(ra)$ is the Chebyshev polynomial of degree s, and s is a random integer that server B selected.

Step 3: Upon receiving the message T_2, A derives $T'_s(ra) = H(H(h_{PW}) \oplus H(T_r(ra))) \oplus (H(H(h_{PW}) \oplus X) \oplus T_s(ra))$ and checks whether the $H(T'_s(ra))$ is equal to the received $H(T_s(ra))$. If so, it proves that the message T_2 was truly sent by server B and that the corresponding Chebyshev polynomial $T_s(ra)$ is valid. Thus, A calculates the message $T_3 = H(h_{PW} \oplus T_s(ra)) \oplus T_r(ra)$ and transmits it to B.

Step 4: After receiving the message T_3, server B computes $T'_r(ra) = H(h_{PW} \oplus T_s(ra)) \oplus T_3$ and checks whether $H(T'_r(ra))$ is equal to the received $H(T_r(ra))$. If so, B confirms that $T_r(ra)$ is valid and keeps it as a secret.

Step 5: From the above, both the user A and server B can negotiate a session key k, where $k = T_r(T_s(ra)) = T_s(T_r(ra)) = T_{rs}(ra)$.

3.2. **Cryptanalysis of Guo and Zhang's protocol.** Although Guo and Zhang's protocol [18] can overcome the security weaknesses of previous work [14] and achieve the contributory nature of key agreement, there are hidden security problems that can be attacked easily. Hence, we will demonstrate our corresponding cryptanalysis in this subsection. It can be seen that Guo and Zhang's protocol cannot achieve the property of key agreement in a group, and it is still vulnerable to off-line password guessing attack. We introduce the details in Subsection 3.2.1 and Subsection 3.2.2, respectively.

3.2.1. *Group key agreement property.* The purpose of group key agreement is to negotiate a common key among group participants, and this common secret key is also called the group key, which can be utilized by all participants to encrypt or decrypt secret information for their private communications. As we know, the group key agreement protocol is generally for a multi-party case, which requires more than two participants to establish the key.

Unlike group key distribution, the main characteristic of group key agreement is contributory property, in which all participants take part in the generation of the key and guarantee that the resulting key is fresh. That means no individual member of the group can predetermine the group key successfully.

After reviewing Guo and Zhang's protocol, we conclude that their protocol only does a good job of guaranteeing the contributory property. However, it is still a specific key agreement protocol between the user and server; it cannot operate in the multi-party scenario in which each group participant can contribute equally to the derivation of the group key. Therefore, Guo and Zhang's protocol is actually a two-party case application for key agreement scheme and not a real multi-party case to satisfy group key agreement requirements.

3.2.2. *Off-line password guessing attack.* In Guo and Zhang's protocol, we assume that an active adversary can intercept the authentication message $AU_A = \{ID_A, ra, H(h_{PW}, ra)\}$ in the authentication phase, since all these parameters $\{ID_A, ra, H(h_{PW}, ra)\}$ come from the user A's selection. Thus, the adversary can use the intercepted message over the insecure channel and achieve a guessing attack as follows:

1. The adversary can forge a fake password value of user A, where $h^*_{PW} = H(ID_A, PW^*_A)$ is derived by using the intercepted parameter ID_A and assumed PW^*_A. Then, the adversary can also compute hash value $H(h^*_{PW}, ra)$.
2. After that, adversary can check whether the $H(h^*_{PW}, ra)$ is equal to the intercepted $H(h_{PW}, ra)$. If so, $H(h^*_{PW}, ra) = H(h_{PW}, ra)$, and the adversary has guessed legitimate user A's password successfully.

Therefore, this protocol incurs the risk of this off-line password guessing attack, and an adversary can easily masquerade a legal user and communicate with the server.

4. **Our Proposed Group Key Agreement Protocol.** In this section, we give a detailed description of our proposed protocol based on the Chebyshev chaotic map. The proposed protocol can achieve mutual authentication between a legal user and a reliable server as well as overcome the weaknesses of the previous work identified in the analysis above. In addition, it does not lose the essential contributory nature for group key agreement protocol. Our protocol consists of five phases, i.e., 1) the initialization phase, 2) the registration phase, 3) the authentication phase, 4) the sub-key contribution and communication phase, and 5) the group key computation phase. We also take into consideration group membership adjustments by including the joining and leaving processes. All the details are provided below.

4.1. **Initialization phase.** As we know, almost all previous group key management research efforts [22-25] exploited a kind of binary-key tree structure because of its inherent efficiency. Our proposed protocol also utilizes this kind of structure, as illustrated in Figure 1. Since the form of our binary-key tree structure has been modified and extended [26], the related technologies achieve the purpose for group key generation and membership adjustments. The notations used throughout this paper are presented below:

- n: number of group participants (current members)
- S: server, which provides participant to be legitimate
- U: set of group participants, i.e., $U = \{U_1, U_2, \ldots, U_n\}$; each U_i for $1 \le i \le n$
- ID_i: identity of the participant U_i
- PW_i: password of the participant U_i
- $h(\cdot)$: one way hash function
- V_{lj}: tree node j at level l
- H: height of the tree structure
- $T_K(\cdot)$: enhanced Chebyshev polynomial of degree K
- TR: tree structure in advance of group membership exchanges
- TR^*: modified tree after group membership exchanges
- M : set of each node's Chebyshev polynomial on TR
- M': set of each node's Chebyshev polynomial on TR^*

As shown in Figure 1, there are n participants, and all of them are associated with a node denoted V_{lj}, in which each level of the binary tree structure can hold 2^l nodes at most and j should satisfy that $0 \le j \le 2^l - 1$. In essence, Figure 1 actually presents an example in which the root node V_{00} is located as level 0 and can be split into two leaf nodes; it is clear that the lowest leaves are located as level 3.

4.2. **Registration phase.** At the beginning, all the participants should register with server S to be legitimate member of the group. In brief, we use U_i as an example member of the group to introduce the specific procedures.

Step 1: U_i first transmits his identity ID_i and PW_i to server S through a secure channel.

Step 2: After receiving the registration request message $\{ID_i, PW_i\}$, S calculates $V = h(ID_i \parallel x) \parallel h(x)$ and $B = V \oplus PW_i$, where x is a secret long-term key chosen by S. Then, S transmits B to U_i over a secure channel.

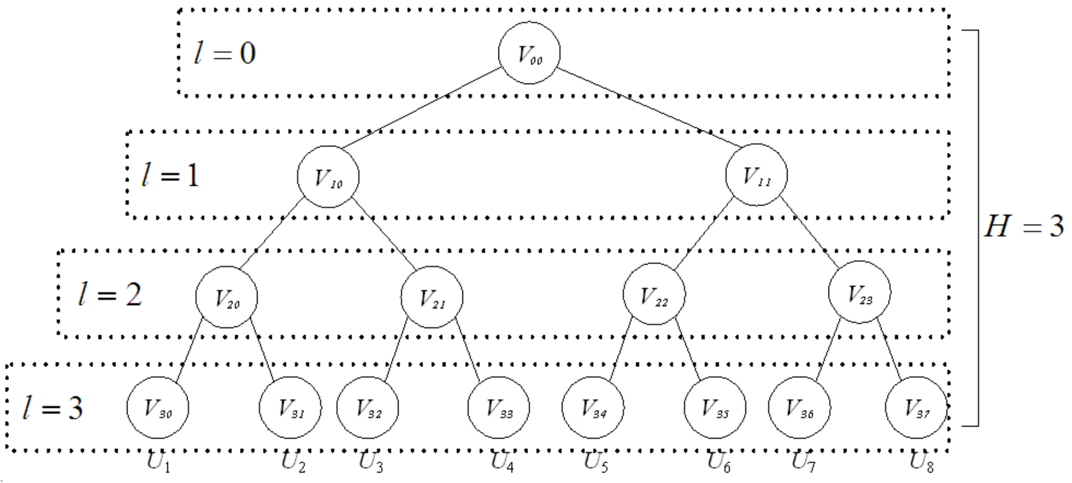

FIGURE 1. Illustration of the binary tree structure

4.3. **Authentication phase.** To undertake the authentication phase, we again use U_i as an example. After the response message is received, U_i and server S execute the following procedures.

Step 1: Upon receiving the response message, U_i can utilize the received B to compute $V' = B \oplus PW_i$ and $D = h(V' \parallel ra)$, and then he or she sends message $T_1 = \{ID_i, ra, D\}$ to S, where ra is a random integer kept by U_i.

Step 2: After the message T_1 is received, S checks the format of ID_i, and uses its secret long-term key x and U_i's identity ID_i to calculate $V = h(ID_i \parallel x) \parallel h(x)$. Then, S can check whether the received parameter $D = h(V' \parallel ra)$ is equal to the hash value $h(V \parallel ra)$ by using the received ra and private V. If not, S stops here; otherwise, U_i is authenticated, and S computes $P = h((ra+1) \parallel V) \oplus rb$ and $Q = h(h((ra+2) \parallel V) \parallel rb)$, where rb is selected randomly by S. Then, S returns the message $T_2 = \{P, Q\}$ to U_i.

Step 3: While receiving the message T_2, U_i uses her or his own random integer ra and parameter V' to calculate $rb' = P \oplus h((ra+1) \parallel V')$ and checks whether the received Q is equal to $h(h((ra+2) \parallel V') \parallel rb')$. If it is, S is authenticated. Then, U_i computes $ACK = h(V' \parallel rb' + 2)$ as an acknowledgement message and sends it to S.

Step 4: S receives the acknowledgement message from U_i and uses its own rb, V to compute $ACK' = h(V \parallel rb+2)$. After checking the equation $ACK \overset{?}{=} ACK'$ stratified, S confirms that the acknowledgement message was sent from U_i.

Therefore, in our proposed protocol, mutual authentication is achieved effectively between the legal user and the authentication server.

4.4. **Sub-key contribution and communication phase.** Without loss of generality, the group key agreement protocol can help group participants to negotiate a common key by initializing $U = \{U_1, U_2, \dots, U_n\}$. Since we utilize the binary tree structure to implement group members' communication, each participant is associated with a tree node. In this phase, each participant must contribute her or his own sub-key for group key generation. For instance, U_i executes the following procedures:

Step 1: U_i, who is located at node of the tree, has her or his own sub-key K_{U_i}, where K_{U_i} is chosen randomly by each group participant and also is named by her or his located node. Then he or she computes $CK_{U_i} = f(K_{U_i}) = T_{K_{U_i}}(rb') = T_{K_{U_i}}(rb)$, where $T_{K_{U_i}}(rb)$ is the enhanced Chebyshev polynomial described in Subsection 2.2, and rb is chosen by the authentication server S.

Step 2: U_i broadcasts her or his CK_{U_i}. In this way, each node V_{lj} that belongs to tree structure should publish respective Chebyshev polynomial CK_{lj} after executing the procedure in Step 1.

Step 3: U_i can derive the key information along the path from her or his node to the root node V_{00}, since the computing key of each node along the path only needs to comply with following recursive function by using her or his one child node key value and the Chebyshev polynomial of the other. The recursive function arises from:

$$\begin{aligned} K_{lj} &= T_{K_{(l+1)2j}}(CK_{(l+1)(2j+1)}) = T_{K_{(l+1)(2j+1)}}(CK_{(l+1)2j}) \\ &= f(K_{(l+1)2j} K_{(l+1)(2j+1)}) \\ &= T_{K_{(l+1)2j} K_{(l+1)(2j+1)}}(rb). \end{aligned}$$

4.5. **Group key computation phase.** According to the preceding phase execution, the involved participant has a relationship with his key-path nodes, which are referred to as the related nodes from his located node to the root directly. Consequently, the root node key K_{00} is the group secret shared among all members to which all involved participants contributed.

As shown in Figure 2, to simplify the situation, we assume a three-height tree structure there is. It is clear that the legal participant U_3 can derive all the keys of dotted curve nodes along the key-path from V_{32} to V_{00}. It is obvious that he or she knows every node's Chebyshev polynomial in the set $M = \{CK_{10}, CK_{11}, \ldots, CK_{37}\}$ and derives the key set $I_{U_2} = \{K_{32}, K_{21}, K_{10}, K_{00}\}$ along the key-path. More specifically, participant U_3 only requires the Chebyshev polynomial set of sibling nodes along her or his key-path $M^*_{U_2} = \{CK_{33}, CK_{20}, CK_{11}\}$ and her or his own sub-key K_{32} (so-called K_{U_3}) to compute the group secret K_{00}.

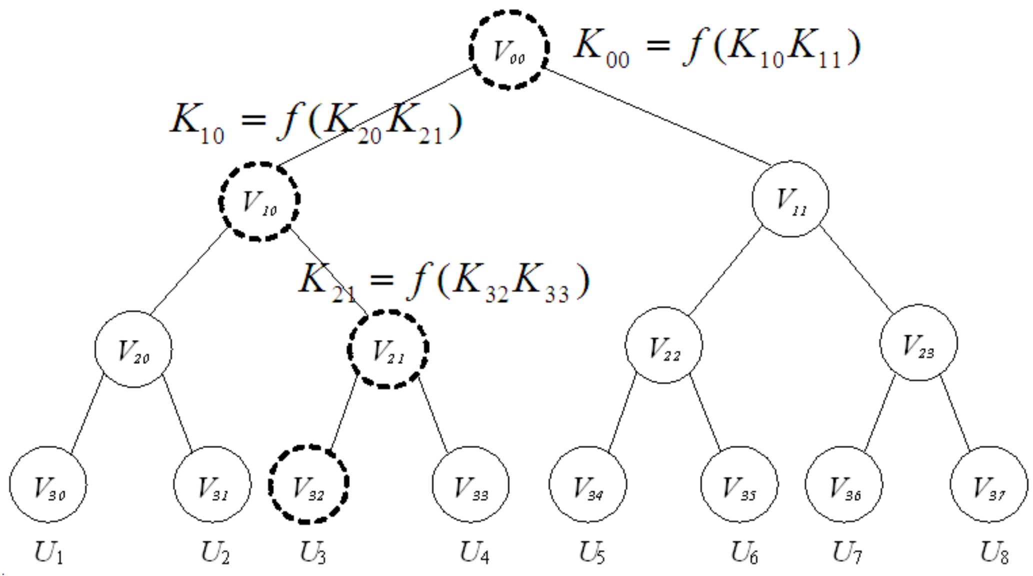

FIGURE 2. Example of U_3's view of the tree structure

We note that the group key $SK = h(K_{00})$, in which it can satisfy a function that

$$SK = h(K_{00}) = h\left(T_{T_{T_{K_{U_1}K_{U_2}}(rb)\cdot T_{K_{U_3}K_{U_4}}(rb)}(rb)\cdot T_{T_{K_{U_5}K_{U_6}}(rb)\cdot T_{K_{U_7}K_{U_8}}(rb)}(rb)}(rb)\right),$$

where $E = \{K_{U_1}, K_{U_2}, \ldots, K_{U_8}\}$, denotes the set of respective sub-keys for current group members of example. In summary, each legal participant owns her or his sub-key and intermediate keys on the corresponding key-path, and eventually he or she can compute the group key.

4.6. **Group membership adjustments.**

One of the main characteristics of our protocol is that it includes group key adjustments according to specific membership exchanges. To present the aforementioned, our proposed protocol supports participants who arbitrarily join or leave the group. Hence, all involved participants can follow the protocol operation correctly to execute group key refreshment to deal with concerns about security requirements. Additionally, an essential feature of our protocol is the so-called sponsor; it can significantly handle the separate joining or leaving process whenever group membership exchanges occur. Next, there is a need to emphasize its other features, which are introduced below.

4.6.1. *Joining process.* It is assumed that a new participant U_j attempts to join the group communication. In practice, this participant U_j must take the following procedures in our protocol as specified below. An illustration of the joining process is provided in Figure 3.

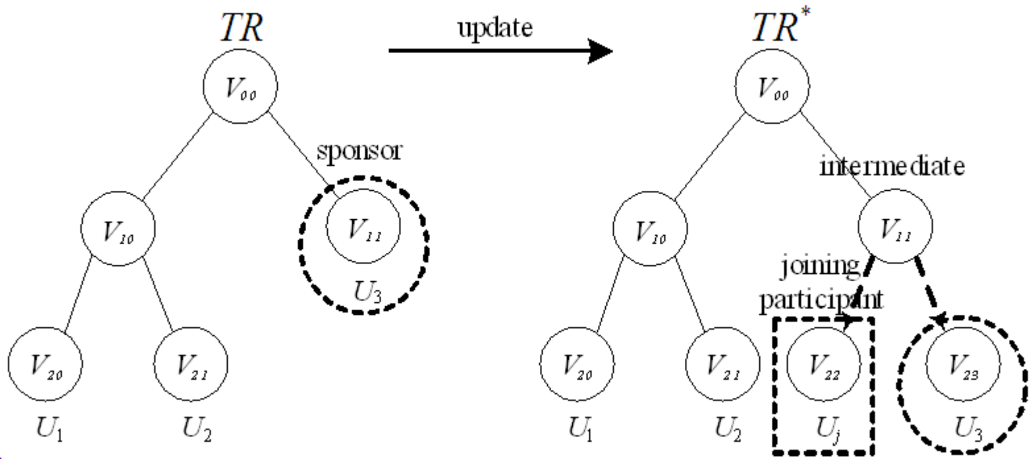

Figure 3. Participant joining process illustration

Step 1: U_j is required to pass through the above registration and authentication phases to be a legal member. Subsequently, he or she broadcasts a joining request message that includes her or his Chebyshev polynomial.

Step 2: After receiving the new participant's message requesting to join the group, there is a requirement for the group to confirm the location of the adaptive insertion node to ensure that the tree height remains the same. And, as always, the insertion node is arranged for the new group member and settled in the subtree of the shallowest rightmost node.

Step 3: In Figure 3, the sponsor U_3, who is normally in the rightmost node, generates an insertion node V_{22} and becomes the sibling of U_j by resetting her or his location from V_{11} to V_{23}.

Step 4: Consequently, V_{11} is promoted to be a new intermediate node by the sponsor, where the intermediate node is the parent of both sponsor and the new participant's node.

Step 5: After updating the tree structure, the sponsor firstly recomputes the group key by using his all known Chebyshev polynomials, which includes U_j's broadcasted one, but all original members currently cannot compute the new group key SK' because they cannot execute the procedure in Step 4.

Step 6: After the sponsor publishes the set M' that contains all necessary Chebyshev polynomials on the updated tree TR^*, other participants can achieve key refreshment according to the new group key computation.

4.6.2. *Leaving process.* It is assumed that a participant U_l who is a member of current set U leaves the group. The process of leaving is illustrated in Figure 4. In essence, the following specific steps are necessary:

In this case, U_l leaves the group with her or his broadcast leaving request message, and it makes membership exchanges of the group infrastructure. The sponsor U_4 is still the rightmost leaf node, where it belongs to the subtree of U_l's sibling node.

Step 1: The fact of membership exchanges is that the sponsor must update the tree TR by deleting node V_{22} and related parent node V_{11}. And the sponsor promotes U_l's preceding sibling node to replace her or his parent.

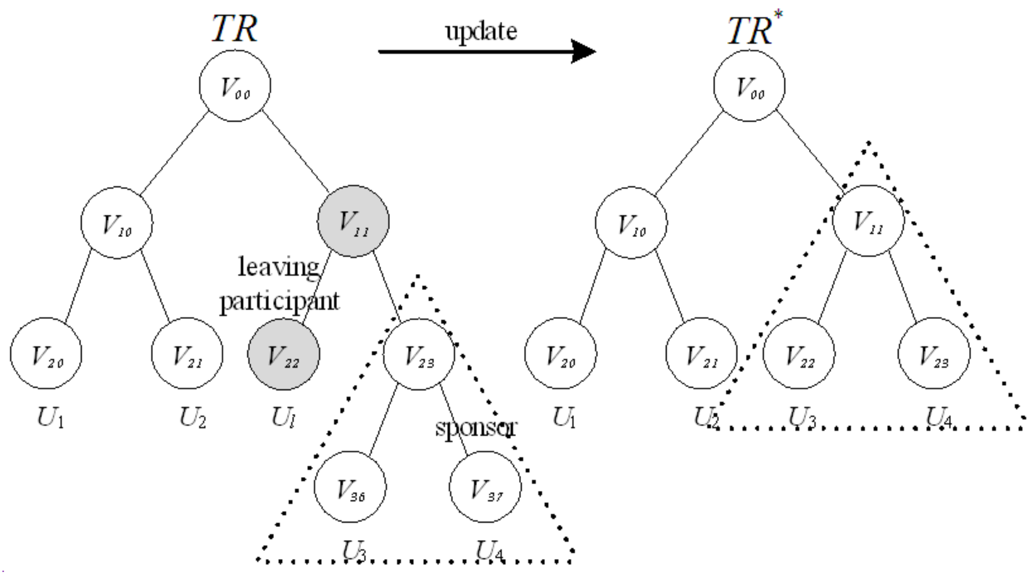

FIGURE 4. Participant leaving process illustration

Step 2: Furthermore, the sponsor selects a new sub-key K'_{U_4}, makes use of the updated tree TR^* to compute all keys along her or his key-path, and broadcasts the new set M' that includes all necessary Chebyshev polynomials of the group.

Step 3: All other participants can update the tree structure accordingly after receiving the sponsor's broadcast message, and then they compute the new group key SK' immediately.

Owing to the above procedures, all group participants accomplish group key refreshment with membership adjustments.

5. **Security Analysis.**

No matter how the functions change, the security requirement is the core feature of protocol design. We can provide in this section the basic analysis and evidence that are necessary for understanding the subsequent results. Meanwhile, we are careful to present all possible attacks and to explain how our proposed protocol can resist them to reach the security demands.

5.1. **Contributory property.**

Owing to the precise conclusion [27], it is claimed that there are three properties that can guarantee the contributory key agreement in group communication. We demonstrate those three properties as follows:

Theorem 5.1. *Each participant contributes her or his respective secret key share to compute the group key.*

Proof: In our protocol, each group member U_i executes formula process to compute corresponding Chebyshev polynomial CK_{U_i} with sub-key K_{U_i} in Step 1 of Subsection 4.4. The broadcasting method of CK_{U_i} securely services for contributing respective key share of legal participant in subsequent Step 2.

Theorem 5.2. *The group key computation cannot be achieved without the information of at least one participant's key share.*

Proof: All involved participants can derive the root node key K_{00} by virtue of the tree-based infrastructure. As seen in Subsection 4.5, our proposed recursive function can help members compute the group key correctly.

Theorem 5.3. *All shared keys are secret and protected securely.*

Proof: It is obvious that each sub-key K_{U_i} is protected by the function of Chebyshev polynomial. In addition, we utilized the enhanced Chebyshev polynomial to improve the instinctive drawbacks of the normal style, in which several Chebyshev polynomials can pass through same point according to the inherent periodicity of the cosine function. It is implied that an adversary is unable to compute a similar sub-key of any legal participant in our proposed protocol indeed, since the periodicity is avoided by extending the interval from $[-1, 1]$ to $(-\infty, +\infty)$ in Section 2.

5.2. **Withstanding impersonation attack.** Assume that an adversary desires to impersonate the message T_1 and convinces the server S to believe her or his validity. We show that the adversary will fail, we explain why. The adversary first forges the message $T_1^* = \{ID_i, ra, D^*\}$ and sends it to S. Upon receiving the forged message, S first checks the format of his identity ID_i. If it is valid, S obtains $h(V \parallel ra)$ by computing $V = h(ID_i \parallel x) \parallel h(x)$, where the x is a secret long-term key selected by S. There is certainty that the adversary cannot solve the equation $D^* = h(V \parallel ra)$. For the same reason, if the adversary wants to dupe the user into believing he or she is a legal server, he or she will fail. In fact, the adversary cannot forge a correct message T_2^*, since the included parameters should satisfy equations, where $P = h((ra + 1) \parallel V) \oplus rb$ and $Q = h(h((ra+2) \parallel V) \parallel rb)$, respectively. Next, he or she cannot pass Step 3 in authentication phase without knowing the private parameter V. Therefore, our proposed protocol can resist this kind of impersonation attack.

5.3. **Withstanding replay attack.** Suppose an adversary intercepts the message T_1 in Step 1 of the authentication phase and then he transmits this message to the server S. Due to the process in Step 2 of the same phase, S will respond the message T_2. Nevertheless, the adversary cannot derive the correct rb to compute the acknowledgement message in Step 3, since he or she has no way to derive parameter V, which only belongs to the legal user. It also leads to the failure of verification in Step 4, such that $ACK \neq ACK'$. In addition, the numbers ra and rb are selected differently in every session. Thus, according to the above analysis, our proposed protocol can withstand the replay attack.

5.4. **Withstanding off-line password guessing attack.** If an adversary obtains all the communication messages, such that $T_1 = \{ID_i, ra, D\}$, $T_2 = \{P, Q\}$ between the legal user and the authentication server, we can prove that he or she will fail to guess the legal user's password from all these intercepted messages. Although the adversary can utilize the intercepted ra and D in message T_1 to guess essential parameter V', he or she still cannot derive the correct value of user's password, because only the legal user can derive the equation $V' = B \oplus PW_i$ by using private B in Step 1 of the authentication phase. Besides, the message T_2 contains unknown random number rb for the adversary, and the secret information is protected by secure, one-way hash function. Hence, our proposed protocol can overcome Guo and Zhang's security weakness with respect to an off-line password guessing attack.

5.5. **Withstanding denial-of-service (Dos) attack.** When the adversary intercepts the communication messages that contains $\{ID_i, ra, D\}$, $\{P, Q\}$ and $\{ACK\}$ to make such a Dos attack, he or she will fail, because the adversary's forged messages can be exposed by the verification in the authentication phase. This is the result of the collision resistance of using one-way hash function in our protocol. Furthermore, we can also be on guard against an attempt in which an adversary wants to block the communication networks by transmitting large numbers of forged messages. Any member in our group key agreement environment has chance to publish messages at most two times: the first time is to broadcast the message of Chebyshev polynomial in Step 2 of the sub-key distribution and communication phase; the second time is to broadcast the leaving request message due to her or his leaving process activates. Therefore, our proposed protocol can withstand the denial-of-service attack.

5.6. **Forward secrecy.** This is an important security concern that avoids leakage happening in case of membership exchanges. More precisely, we state that a new participant who joins the group obtains current group key, but this will not compromise the preceding group key established by original participants. If any participant U_j wants to join the group after he or she can be authenticated, he needs to choose a secret key K_{U_j}, protected it by using Chebyshev polynomial, and then broadcasts $T_{K_{U_j}}(rb)$ as a request to join the group. Upon receiving required set M', U_j can calculate all secret keys along her or his key-path up to the group secret K_{00}. It is evident that all related keys include U_j's contribution part. Therefore, U_j really cannot obtain any information of previously established group keys, since all of them are distinguishingly contributed by independent secret keys on that path. Our proposed protocol can achieve the forward secrecy.

5.7. **Backward secrecy.** There is another essential requirement and that is to make sure that a participant who leaves the group cannot derive any secret information about the new group. Recall that, in the leaving process, once a member U_l leaves the group, the sponsor deletes her or his located node and replaces the corresponding parent node with U_l's sibling. To update key information, the sponsor accordingly selects a new key secret. Hence, it leads to a clear exclusion of the departed member's contribution along her or his original key-path. Thus, our protocol will prevent any participant who leaves the group from discovering subsequent group keys, so we also achieve backward secrecy in our protocol.

6. **Discussions.**

In this section, we offer three kinds of comparisons between our proposed protocol and the previous work [18] in order to address our major concerns. The three kinds of comparisons are 1) performance comparison, 2) functionality comparison, and 3) security requirements comparison. We can demonstrate the details as follows.

First, we explain the details of the performance comparison, in which we compare the proposed scheme with Guo and Zhang's protocol [18] based on the computation cost associated with one-way hash function (*Hash*), chaotic cryptographic operation (*Cha*) in the authentication procedure, and the group key generation procedure. The computation cost of the two protocols are presented in Table 1.

Table 1. Performance comparisons

Items	Guo and Zhang's protocol [18]	Our proposed protocol
Authentication procedure	$6Hash$	$10Hash$
Group key generation procedure	$7Hash + 3Cha$	$1Hash + (\lceil \log_2 n \rceil + 1)Cha$

As shown in Table 1, we present the actual computation cost for the two protocols. In our proposed group key agreement protocol, we consider that each participant should comply with the recursive function to derive the final group key. Thus, every member in the proposed protocol must execute $(H + 1)$ chaotic cryptographic operations, where H means the height of the tree structure, derived from function $H = \lceil \log_2 n \rceil$. In fact, Guo and Zhang's protocol is really a two-party case that relates to the scheme for the user and the server. To simplify the situation, if our assumption is two-party environment, the total computation cost has $11Hash + 2Cha$, which is more efficient than $13Hash + 3Cha$ in Guo and Zhang's protocol.

The results for the functionality comparisons are depicted in Table 2. As we know, the achievement of mutual authentication affects the generation of the session key between the user and the server, ensuring that their future communication will be private. In addition, due to the use of timestamps, there is a need for both the user and the server to harmonize a synchronized clock, and the synchronization problem in the system will require the support of additional hardware. Meanwhile, the contributory protocol is definitely a main attribute for the group key agreement protocol. As shown in Table 2, all of the aforementioned functions can be satisfied with practical requirements. The main point to emphasize is that our proposal includes criteria to ensure that the proposed protocol attains the goals of enforcing realistic group key generation and use and making group membership adjustments, neither of which the previous work could achieve. Hence, our proposed protocol has significant practical advantages in its functionality comparison with Guo and Zhang's protocol.

Last, we describe the comparison of security requirements in Table 3. It is obvious that our proposed protocol is more secure than Guo and Zhang's protocol, as expressed in fact that our proposed protocol can withstand a series of attacks. From the perspective of security requirements, our proposed protocol is superior to the previous work on the basis of resistance to off-line password guessing attack, protection of forward secrecy, and protection of backward secrecy. The ultimate objectives of our remedy in security concerns and enhancing functionality are both practical and prescriptive.

7. Conclusions.

In this article, we demonstrated that Guo and Zhang's group key agreement protocol is not as secure as has been claimed. In addition, there is an infrastructure

Table 2. Functionality comparisons

Items	Guo and Zhang's protocol [18]	Our proposed protocol
Mutual authentication	Yes	Yes
Without synchronized problem	Yes	Yes
Realistic group key agreement achievement	No	Yes
Contributory property	Yes	Yes
Group membership adjustments	No	Yes

Table 3. Security requirements comparisons

Items	Guo and Zhang's protocol [18]	Our proposed protocol
Resistance to impersonation attack	Yes	Yes
Resistance to replay attack	Yes	Yes
Resistance to off-line password guessing attack	No	Yes
Resistance to denial of service attack	Yes	Yes
Protection of forward secrecy	No	Yes
Protection of backward secrecy	No	Yes

leakage that prevents participants from cooperating simultaneously for group communications. To the contrary, we have presented a practical group key agreement protocol, based on chaos theory, that offers greater security and better functionality compared with protocols developed in previous research. Also, we solved the weaknesses of Guo and Zhang's protocol and improved the group key agreement for the dynamic case. Furthermore, we actually propose a more efficient authentication scheme with collaborated members and compare our protocol with the published achievements. Due to its functionality advantages reflected in the contributory property without synchronized time design, our proposed protocol can be employed for distributed communication networks even though the users are located at any places with different time zones. Therefore, our novel method is most suitable for distributed networks with group communications.

Acknowledgment. This work is partially supported by National High Technology and Development Program (863 Program) of China under Grant No. 2011AA010104-2, National Natural Science Foundation of China under Grant 61071076, the Academic Discipline and Postgraduate Education Project of Beijing Municipal Commission of Education, the Fundamental Research Funds for the Central Universities under Grant 2012YJS023. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] I. Ingemarsson, D. Tang and C. Wong, A conference key distribution system, *IEEE Transactions on Information Theory*, vol.28, no.5, pp.714-720, 1982.

[2] C. K. Wong, M. Gouda and S. S. Lam, Secure communications using key graphs, *IEEE/ACM Transactions on Networking*, vol.8, no.1, pp.16-30, 2000.

[3] Y. Cai and Y. Wang, Identity-based conference key distribution protocol with user anonymity, *Chinese Journal of Electronics*, vol.16, no.1, pp.179-181, 2007.

[4] L. Harn and C. Lin, Authenticated group key transfer protocol based on secret sharing, *IEEE Transactions on Computers*, vol.59, no.6, pp.842-846, 2010.

[5] Y. Kim, A. Perrig and G. Tsudik, Group key agreement efficient in communication, *IEEE Transactions on Computers*, vol.53, no.7, pp.905-921, 2004.

[6] Y. Mao, Y. Sun, M. Wu and K. J. R. Liu, Dynamic join-exit amortization and scheduling for time-efficient group key agreement, *Proc. of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, pp.2617-2627, 2004.

[7] W. Kim, E. Ryu, J. Im and K. Yoo, New conference key agreement protocol with user anonymity, *Computer Standards and Interfaces*, vol.27, no.2, pp.185-190, 2005.

[8] B. Jung, An efficient group key agreement protocol, *IEEE Communications Letters*, vol.10, no.2, pp.106-107, 2006.

[9] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, Asymmetric group key agreement, *Advances in Cryptology – EUROCRYPT*, Berlin, pp.153-170, 2009.

[10] S. Jarecki, J. Kim and G. Tsudik, Flexible robust group key agreement, *IEEE Transactions on Parallel and Distributed Systems*, vol.22, no.5, pp.879-886, 2011.

[11] D. Xiao, X. Liao and K. Wong, An efficient entire chaos-based scheme for deniable authentication, *Chaos, Solitons & Fractals*, vol.23, no.4, pp.1327-1331, 2005.

[12] P. Bergamo, P. D'Arco, A. Santis and L. Kocarev, Security of public-key cryptosystems based on Chebyshev polynomials, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol.52, no.7, pp.1382-1393, 2005.

[13] G. Alvarez, Security problems with a chaos-based deniable authentication scheme, *Chaos, Solitons & Fractals*, vol.26, no.1, pp.7-11, 2005.

[14] D. Xiao, X. Liao and S. Deng, A novel key agreement protocol based on chaotic maps, *Inform. Sci.*, vol.177, no.4, pp.1136-1142, 2007.

[15] S. Han, Security of a key agreement protocol based on chaotic maps, *Chao. Solit. Fract.*, vol.38, no.3, pp.764-768, 2008.

[16] D. Xiao, X. Liao and S. Deng, Using time-stamp to improve the security of a chaotic maps-based key agreement protocol, *Inform. Sci.*, vol.17, no.6, pp.1598-1602, 2008.

[17] S. Han and E. Chang, Chaotic map based key agreement with/out clock synchronization, *Chaos, Solitons & Fractals*, vol.39, no.3, pp.1283-1289, 2009.

[18] X. Guo and J. Zhang, Secure group key agreement protocol based on chaotic hash, *Information Sciences*, vol.180, no.20, pp.4069-4074, 2010.

[19] L. Kocarev and Z. Tasev, Public-key encryption based on Chebyshev maps, *Proc. of International Symposium on Circuits and Systems*, pp.28-31, 2003.

[20] L. Zhang, Cryptanalysis of the public key encryption based on multiple chaotic systems, *Chaos, Solitons & Fractals*, vol.37, no.3, pp.669-674, 2008.

[21] D. Xiao, X. Liao and S. Deng, One-way hash function based on the chaotic map with changeable-parameter, *Chaos, Solitons & Fractals*, vol.24, no.1, pp.65-71, 2005.

[22] Y. Kim, A. Perrig and G. Tsudik, Tree-based group key agreement, *ACM Transactions on Information and System Security*, vol.7, no.1, pp.60-96, 2004.

[23] R. Dutta, R. Barua and P. Sarkar, Provably secure authenticated tree based group key agreement, *Proc. of the Conference on Information and Communications Security*, Berlin, pp.45-51, 2004.

[24] R. Dutta and R. Barua, Dynamic group key agreement in tree-based setting, *Proc. of the Conference on Information Security and Privacy*, Berlin, pp.101-112, 2005.

[25] Y. Desmedt, T. Lange and M. Burmester, Scalable authenticated tree based group key exchange for ad-hoc groups, *Proc. of the 11th International Conference on Financial Cryptography and the 1st International Conference on Usable Security*, pp.104-118, 2007.

[26] Y. Kim, A. Perrig and G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups, *Proc. of the 7th ACM Conference on Computer and Communications Security*, pp.235-244, 2000.

[27] M. Steiner, G. Tsudik and M. Waidner, Key agreement in dynamic peer groups, *IEEE Transactions on Parallel and Distributed Systems*, vol.11, no.8, pp.769-780, 2000.