IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INFORMATION SECURITY SOLUTIONS FOR TELEMEDICINE APPLICATIONS

Received December 14, 2017, accepted January 16, 2018, date of publication January 26, 2018, date of current version May 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2799007

A Dynamic and Cross-Domain Authentication Asymmetric Group Key Agreement in Telemedicine Application

ZHANG QIKUN[1], GAN YONG[1], ZHANG QUANXIN[2], WANG RUIFANG[1], AND TAN YU-AN[2]

[1]Institute of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China
[2]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Zhang Quanxin (zhangqx@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1636213, Grant 61772477, Grant 61572445, and Grant 61501406, in part by the Ph.D. Research Fund of the Zhengzhou University of Light Industry, and in part by the Natural Science Foundation of Henan Province under Grant 162300410322.

ABSTRACT Telemedicine offers medical services remotely via telecommunications systems and physiological monitoring devices. Group-oriented communication is an important application for telemedicine. However, transmission of information over an insecure channel such as Internet or private data storing generates a security problem. Therefore, authentication, confidentiality, and privacy are important challenges in telemedicine. Therefore, developing suitable encryption communication protocol for group communication is quite important for modern medicine. Group key agreement is one way to ensure the security of group-oriented communication for telemedicine. In this paper, we propose a dynamic and cross-domain authenticated asymmetric group key agreement. The protocol adopts cross-domain authentication mechanism to avoid the security risks of key escrow and the complexity of certificate management. It supports the dynamic group key update of nodes for forward secrecy and backward security of group key, and also achieving the key self-certified, the member participated group key agreement can self-certify whether the calculated group keys are correct. The protocol is proven secure under the inverse computational Diffie-Hellman problem assumption, and the performance analysis shows that the proposed scheme is highly efficient. The proposed scheme is more suitable for security group communication in telemedicine.

INDEX TERMS Telemedicine, group key agreement, certificateless authentication, key self-certified.

I. INTRODUCTION

Recently, advances in computers and high-speed communication tools have led to enhancements in remote medical consultation research. Laws in some localities require hospitals to encrypt patient information before transferring the data over a network. Therefore, developing suitable encryption algorithms is quite important for modern medicine.

Murillo-Escobar *et al.* [1] proposed a novel symmetric encryption algorithm based on logistic map with double chaotic layer encryption (DCLE) in diffusion process and just one round of confusion-diffusion for the confidentiality and privacy of clinical information such as electrocardiograms (ECG), electroencephalograms (EEG), and blood pressure (BP) for applications in telemedicine. Yin *et al.* [2] demonstrated a digital image encryption algorithm based on chaotic mapping, which uses the no-period and no-convergence properties of a chaotic sequence to create image chaos and pixel averaging.

Group-oriented communication applications are a widely used in telemedicine. For the group security communication, studying group encryption communication algorithms is quite important for modern medicine. Group key agreement is one way to ensure the security of group communication for medical corps. Omid and Morteza [3] proposed an improved biometrics-based authentication and key agreement scheme. Asymmetric Group Key Agreement (ASGKA) was first introduced by Wu *et al.* [4] to solve the problem. Li *et al.* [5] proposed Dekey, an efficient and reliable convergent key management scheme for secure deduplication. Dekey applies deduplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. Xiaosong *et al.* [6] and Davis and Cogdell [7] proposed authenticated key exchange respectively, but these schemes depend on the third party, and these schemes are symmetric group key

agreement protocols. Ranjani [8] proposed a new cryptosystem termed as Asymmetric group key agreement cryptosystem, group users broadcast their contribution to other group members by keeping their own information secret. Each group user collects broadcast message from other group participants and derive a common group key.

Dynamic asymmetric group key agreement concerns about the scenarios such as the terminals in mobile cloud networks may join or leave at any given time. Yeun *et al*. [9] improved the GKA protocol by designing additional security requirements, making it is applicable to dynamic MANET environments. Li *et al*. [10] proposed a new Secure Outsourced ABE system, which supports both secure outsourced key-issuing and decryption. Xingwen *et al*. [11] proposed a dynamic AGKA protocol, the scheme improved the flexibility of AGKA, and it is suitable to be applied in ad hoc network, it adopts multi-signature scheme to achieve authentication of protocol. For the cost of computation and communication of signature and signature verification are very large, if the protocol is applied to the resource-limited mobile devices networks, the calculation and communication of this protocol must be further simplified. Bilal and Kang [12] propose a novel methodology for group key agreement which exploits the state vectors of group members. The state vector empowers the group member to generate multiple cryptographic keys independently. It supports the dynamic group key agreement.

Authenticated key agreement protocols authenticate the identities of users to ensure that only the intended group members can establish a session in which the group members can communicate with each other.

Farash [13] and Hongfei *et al.* [14] proposed an effective authenticated key exchange agreement respectively. It achieves mutual authentication without applying signature, which makes the protocol more practical. Lv *et al*. [15] proposed authenticated asymmetric group key agreement based on certificateless cryptosystem, which does not require certificates to guarantee the authenticity of public keys yet avoids the inherent escrow problem of identity-based cryptosystems. Zhu and Zhang [16] presented a novel group key agreement protocol with deniable authentication to against insider attack. The group participants unable to reveal the source of the messages to another party, any subgroup participants still can simulate the whole transcript process. Ranjani *et al*. [17] proposed an authenticated asymmetric group key agreement protocol; they used broadcast encryption mechanism without relying on the trusted dealer to distribute the secret key, which offers security against active as well as passive attacks. Zhang *et al*. [18] introduce an authenticated asymmetric group key agreement protocol which offers security against active attacks in open networks. Based on this protocol, they propose a broadcast encryption system without relying on a trusted dealer to distribute the secret keys to the users. Xu *et al*. [19] proposed a one-round affiliation-hiding authenticated asymmetric group key agreement with semi-trusted group authority, which can protect personal privacy, but the calculation of this protocol is very large, it is not suitable to use in energy-limited mobile devices of wireless network. Xu *et al*. [20] proposed a new affiliation-hiding protocol. The scheme not only exhibits the affiliation-hiding property, but also holds the properties of detectability and perfect forward secrecy. Qikun *et al*. [21], Wei *et al*. [22], and Zhang *et al.* [23] proposed authenticated AGKA protocol to improve the security of AGKA respectively. Li *et al*. [24] proposed a one-round authenticated dynamic protocol for symmetric group key agreement. They employed the identity-based public-key cryptography to authenticate users rather than the public key infrastructure and the certificate-less public-key cryptography.

This paper proposes a dynamic and cross-domain authenticated asymmetric group key agreement (DC-AAGKA) in telemedicine application. The main design goals of this paper are as follows:

(1) The DC-AAGKA protocol is a cross-domain authentication protocol and provably security key escrow freeness.

(2) The DC-AAGKA protocol makes the mobile terminals distributed different domains to securely exchange information and share secrets.

(3) We design an asymmetric key negotiation with key self-certified.

(4) We design an asymmetric key agreement with dynamic to allow the members to join and exit the group.

A. OUR CONTRIBUTION

Telemedicine networks have the some characteristics, such as members in medical corps that participated in collaborative computing and information sharing may be distributed in different secure domains, and the members leave one group or join another group frequently. In order to meet the security application need of remote medical system, we designed a dynamic and cross-domain authenticated asymmetric group key agreement protocol. The contribution is as follows: 1) cross-domain authenticated. DC-AAGKA requires the participants whether in different domain must to mutual authentication before negotiating the group keys, which to avoid illegal members to participate in group key agreement; 2) Key self-certified. All participants can verify the group keys correctness without any other additional communication; 3) Dynamic. DC-AAGKA protocol supports the members to join or exit the group, and it has the group key forward secrecy and backward secrecy security.

B. ORGANIZATION

In Section II, we describe the basic knowledge associated with this paper. In Section III, we describe the DC-AAGKA protocol and key update protocol. In Section IV, we analyze and prove the correctness and safety of the protocol. We further analyze the performance in Section V. Finally, we conclude the paper in Section VI.

II. PRELIMINARIES AND NOTATION

A. BILINEAR MAPS AND COMPLEXITY ASSUMPTIONS

This paper is based on the basic theory of bilinear mapping; some basic knowledge related to bilinear mapping will be described in this section.

Let G_1 be an additive group and G_2 is a multiplicative group. Both of them have the same prime order q, where $q \geq 2^k + 1$, and k is a security parameter. G_1 is generated by g_1, that means $G_1 = \langle g_1 \rangle$, and the discrete logarithm problems of G_1 and G_2 are difficult. We call e an admissible pairing, if $e : G_1 \times G_1 \to G_2$ satisfies the follow properties:

(1) bilinearty: For all $u, v \in G_1$, and $a, b \in \mathbb{Z}_p^*$, there is $e(au, bv) = e(u, v)^{ab}$;

(2) Non-degeneracy: There exits $u, v \in G_1$, such that $e(v, u) \neq 1$;

(3) Computability: For all $u, v \in G_1$, there exits a efficient way to calculate $e(v, u)$.

Inference1: For all $u_1, u_2, v \in G_1$, there is $e(u_1 + u_2, v) = e(u_1, v)e(u_2, v)$.

Definition 1: Discrete Logarithm problem (DLP). Given an equation $Y = aP$, and $Y, P \in G_1$,$a < q$. If a and P are given, it is easy to calculate Y. But if P and Y are given, it will be difficult to calculate a.

Definition 2: Inverse Computational Diffe-Hellman (ICDH) Problem: The IDH problem is given g_1, ag_1 and abg_1, for some $a, b \in \mathbb{Z}_p^*$ to compute $(ab/a)g_1$.

B. PARTICIPANTS AND NOTATIONS

Attributes of asymmetric group key negotiation protocol security model is given. And then, definition of formal definition of security model and Security objectives to be achieved are stated in this section.

Suppose U is the set of senor nodes participating in key agreement protocol, and the size of the set is polynomial bounded, each senor node in the set U has own identity information and corresponding public/private key pair. In any subset of U, $U_{sub} = \{u_{i,1}, u_{i,2}, \ldots, u_{i,n}\}$, the node can execute the key agreement protocol at any time Π. $\Pi_{u_{i,j}}^{\pi}$ is the πth instance that the node executes asymmetric key agreement with its partner nodes $\{u_{i,1}, \ldots, u_{i,j-1}, u_{i,j+1} \ldots, u_n\}$, $\pi \in mathbbZ_p^*$, and each instance of the node participating in $\Pi_{u_{i,j}}^{\pi}$ has various attributes, defined as follows:

$sid_{u_{i,j}}^{\pi}$: Session key identity of instance $\Pi_{u_{i,j}}^{\pi}$. All partners participating in this session key agreement will have the same session key identity.

$pid_{u_{i,j}}^{\pi}$: Partner identity label of instance $\Pi_{u_{i,j}}^{\pi}$ which is a identity set of all participants establishing session key with the $\Pi_{u_{i,j}}^{\pi}$, including $\Pi_{u_{i,j}}^{\pi}$ itself.

$ek_{u_{i,j}}^{\pi}$: Group session encryption key which is calculated by participants after instance $\Pi_{u_{i,j}}^{\pi}$, executes the key agreement protocol Π.

$dk_{u_{i,j}}^{\pi}$: Group session decryption key which is calculated after the instance $\Pi_{u_{i,j}}^{\pi}$ executes the key negotiation protocol.

$ms_{u_{i,j}}^{\pi}$: The link of all information sent and received during executing the key agreement.

$state_{u_{i,j}}^{\pi}$: The current status of instance $\Pi_{u_{i,j}}^{\pi}$. If the protocol ends up, and the instance $\Pi_{u_{i,j}}^{\pi}$ sent and received all information, and it also calculated the parameters ($ek_{u_{i,j}}^{\pi} \neq null$), ($dk_{u_{i,j}}^{\pi} \neq null$), $sid_{u_{i,j}}^{\pi}$ and $pid_{u_{i,j}}^{\pi}$, then the current state of instance $\Pi_{u_{i,j}}^{\pi}$ is receiving status, otherwise it enters the ending state.

Definition 1 (Partnering): The definition of partners of the instance $\Pi_{u_{i,j}}^{\pi}$ is that all instances participating in the session key agreement with the instance $\Pi_{u_{i,j}}^{\pi}$. Instance $\Pi_{u_{i,j}}^{\pi}$ and its partners meet the following conditions:

1) Instance $\Pi_{u_{i,j}}^{\pi}$ is a different entity from its partners, that is $u_{i,j} \neq u_{i,t}$; 2) They can accomplish group session key agreement successfully; 3) $sid_{u_{i,j}}^{\pi} = sid_{u_{i,t}}^{\pi'}$; 4) $pid_{u_{i,j}}^{\pi} = pid_{u_{i,t}}^{\pi'}$.

C. SECURITY MODEL

DC-AAGKA protocol proposed in this paper, the group encryption key and the group decryption key are calculated by the different mobile terminals with the key parameters of themselves. In this paper, confidentiality is defined as distinguishing a message encrypted with a group encryption key and the advantage of a random string in the cipher text space can be ignored in the probability polynomial time. The security of the group key agreement protocol is defined by game rules between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In security model, it includes an active attacker and a set of protocol participants, each of whom being modeled as a random oracle, and the game is as the follow steps:

Initialization Phase: In this phase challenger $\mathcal{C}$ runs the system function $Setup(\ell)$, outputting related system parameter and master key. Then system parameter will be sent to adversary, and master key is reserved.

Training Phase: The adversary interacts with the challenger $\mathcal{A}$ by asking following steps:

$Send(\Pi_{u_{i,j}}^{\pi}, m)$: Adversary $\mathcal{A}$ pretending as $pid_{u_{i,j}}^{\pi}$ sends message m to oracle machine $\Pi_{u_{i,j}}^{\pi}$, if message $m = null$ (null is empty), oracle machine $\Pi_{u_{i,j}}^{\pi}$ will initiate a conversation as an active sponsor of the session. Otherwise it will be as a respondent, and response to the corresponding inquiries from adversary according to protocol rules, make a decision of accepting or refusing session and record the message received and sent at each time.

$Ek.Reveal(\Pi_{u_{i,j}}^{\pi})$: After oracle machine $\Pi_{u_{i,j}}^{\pi}$ receives this query, it will return group encryption key $ek_{u_{i,j}}^{\pi}$ calculated during executing group key agreement, if oracle machine is in the rejection state, it will return a terminal symbol $\perp$.

$Dk.Reveal(\Pi_{u_{i,j}}^{\pi})$: After oracle machine $\Pi_{u_{i,j}}^{\pi}$ receives this query, it will return a group decryption key calculated during executing group key agreement.If oracle machine is in rejection state, it will return a terminal symbol $\perp$. It is used to simulate security of known key.

$Corrupt(u_{i,j})$: This query is required the participant inquired return its long-term private key $SK_{u_{i,j}}$, once adversary has long-term private key of participant u_i, it can pretend as participant u_i through $Send$. entity already responded to

Corrupt query is called "corrupted". Use it to simulate the forward security of the key.

Test($\Pi^{\pi}_{u_{i,j}}$)*:* At some time during the game, sends two messages (m_0, m_1) $(|m_0| = |m_1|)$. *Test* query send to a new freshness definition of freshness is explained in definition 2) oracle machine $\Pi^{\pi}_{u_i}$ allows adversary $\mathcal{A}$ to make *Test* query only once.Oracle machine $\Pi^{\pi}_{u_i}$ chooses a bit $b \in \{0, 1\}$ randomly, then returns a new sequence generated by XOR b with the sequence chosen randomly, to $\mathcal{A}$.

Output: After game ends, adversary $\mathcal{A}$ outputs a guess value on bit b (remember as b'). If $b' = b$, adversary $\mathcal{A}$ wins the game. The probability of winning the game successfully of adversary $\mathcal{A}$ is defined as $Adv^{\mathcal{A}}(\ell) = \left|2\Pr[b' = b] - 1\right|$, ($\ell$ is security parameter).

Definition 2 (Freshness): If an oracle machine $\Pi^{\pi}_{u_{i,j}}$ meets the following conditions when the game ends, this oracle machine $\Pi^{\pi}_{u_{i,j}}$ (satisfy $pid^{\pi}_{u_i} = P_j$) is freshness: 1) $\Pi^{\pi}_{u_{i,j}}$ is in the acceptance state; 2) $\mathcal{A}$ did not do query to oracle machine $\Pi^{\pi}_{u_{i,j}}$ and it partner oracle machine $\Pi^{\pi'}_{u_{i,t}}$; 3) not do query *Corrupt*to the set of participating entities P_j.

III. THE PROCESS OF DC-AAGKA

A. INITIALIZATION

Assuming the mobile cloud network contains N domains, and there are R members to participate in group key agreement in the same domain D_i $(1 \le i \le R)$, at most. Let $U_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,n}\}$ be the set of members in the domain D_i $(1 \le i \le R)$, and the $ID_i = \{id_{i,u_1}, id_{i,u_2}, \ldots, id_{i,u_n}\}$ be the set of members' identity in the domain D_i. Assuming G_1 is an additive group, and the G_2 is a multiplicative group and discrete logarithm over G_1 and G_2 are difficult. Assuming $G_1 = \langle g_1 \rangle$, g_1 is generator of G_1. G_1nd G_2 have the same large prime number order q. e is calculable bilinear mapping, $e : G_1 \times G_1 \to G_2$. and $H_3 : \mathbb{Z}^*_p \to \mathbb{Z}^*_p$, $H_4 : G_1 \to \mathbb{Z}^*_p$, are two hash function. system parameter is $params = (q, G_1, G_2, g_1, e, H_3, H_4)$.

Assuming $u_{j,k}(1 \le j \le R, 1 \le k \le n)$ is any member in any domain, where j denotes the member $u_{j,k}$ in the domain D_j, and k denotes the member $u_{j,k}$ is the kth member in the domain D_j.

B. REGISTER KEY GENERATION

Assume members participating the DC-AAGKA protocol come from different domain in the mobile cloud platform. Each domain has a certification authority (CA), each members must registers to its CA when it join this domain. Each member from different domains should be verified before participated in DC-AAGKA.

All the CAs in the mobile cloud network needs to negotiate an alliance public/private key pair before their members to register. Suppose all the CAs negotiate an alliance public/private key pair (PK_{alli}, SK_{alli}). Let take the registration of domain D_j as an example, the registration are as follows:

The certification authority CA_j of domain D_j chooses $SK_{CA_j} \in \mathbb{Z}^*_p$ as its private key randomly and it calculates $PK_{CA_j} = g_1\, SK_{CA_j}$ as its public key. The any member $u_{j,k}$ chooses a random positive integer $\gamma_{j,k} \in \mathbb{Z}^*_p$ and calculates $s_{u_{j,k}} = H_3(id_{j,u_k})$, $Sk_{u_{j,k}} = \gamma_{j,k} s_{u_{j,k}}$, $Pk_{u_{j,k}} = g_1\, Sk_{u_{j,k}}$, where the $(Pk_{u_{j,k}}, Sk_{u_{j,k}})$ can as its public/private key pairs.

(1) $u_{j,k}$ sends its own identity message id_{j,u_k} and corresponding public key $Pk_{u_{j,k}}$ to the CA_j for register.

(2) CA_j chooses a positive integer $\eta in \mathbb{Z}^*_p$ randomly and calculates a blind parameter $f_{CA_j} = (\eta / (SK_{CA_j} + SK_{alli}))g_1$ using its own private key SK_{CA_j} and the alliance private key SK_{alli}. Then, it sends the result f_{CA_j} to $u_{j,k}$.

(3) After receiving the parameter f_{CA_j}, $u_{j,k}$ calculates a parameter $f_{u_{j,k}} = Sk_{u_{j,k}} f_{CA_j}$ using its own private key. Then, $u_{j,k}$ sends $f_{u_{j,k}}$ to CA_j.

(4) After receiving the parameter $f_{u_{j,k}}$, CA_j calculates a parameter $r_{u_{j,k}} = \eta^{-1} f_{u_{j,k}}$, and verifies the equation $e(r_{u_{j,k}}, PK_{CA_j} + PK_{alli}) = e(Pk_{u_{j,k}}, g_1)$. If it's hold, then CA_j sends $r_{u_{j,k}}$ to $u_{j,k}$ as a register key. $u_{j,k}$ register successfully. All the members participated in DC-AAGKA should register to their CA as above.

C. REGISTER KEY GENERATION

In this stage, members participated in group key agreement will calculate a group encryption key and a group decryption key which are used in security communicate among members in a group.

Assuming there are R domains in the network, and there are n members in each domain at most, so the members in the network can be denoted with the set $\Phi = \{u_{j,k} \,|1 \le j \le R, 1 \le k \le n\}$, where j denotes the member $u_{j,k}$ is belonged to the domain D_j, and k denotes that the member $u_{j,k}$ is the kth member in the domain D_j. For description convenience, assuming that members participated in dc-AAGKA come from the kth member in each domain, so all the members of participated in DC-AAGKA can be denoted with the set $U = \{u_{j,k} \,|(1 \le j \le R, 1 \le k \le n)\}$. DC-AAGKA protocol is as follows:

(1) Each member $u_{j,k}$ participated in DC-AAGKA chooses a positive integer $m_{j,k} \in \mathbb{Z}^*_p$ randomly and calculates parameters $T_{j,k} = m_{j,k} r_{u_{j,k}}$, $M_{i,k} = m_{j,k} Pk_{u_{j,k}}$, $(1 \le i \le R, i \ne j)$. Then, $u_{j,k}$ broadcasts the messages $(D_j, T_{j,k}, M_{i,k}, id_{j,u_k})$ to other members in the group.

(2) After receiving the message as shown in TABLE.1, each member $u_{i,k}(1 \le i \le R, i \ne j)$ calculates $\overline{M}_{i,k} = m_{i,k} Pk_{u_{i,k}}$, $\overline{\delta}_{i,k} = Sk^{-1}_{i,k} M_{j,k} = m_{j,k} g_1$, $\overline{\delta}_{i,k} = Sk^{-1}_{i,k} M_{j,k} = m_{j,k} g_1$ (only $u_{i,k}$ can calculates $\overline{M}_{i,k}$, and $\overline{M}_{i,k}$ is not broadcasted to others).

(3) Then, $u_{i,k}$ verifies the identity of $u_{j,k}$ by equation $e(T_{j,k}, (PK_{CA_j} + PK_{alli})) = e(Pk_{u_{j,k}}, \overline{\delta_{i,k}})$. If it holds, $u_{i,k}$ can ensure that $(D_j, T_{j,k}, M_{i,k}, id_{j,u_k})$ are sent by $u_{j,k}$. Otherwise, $u_{i,k}$ reports error.

(4) If all of the group members are verified successfully, each member $u_{i,k} \in U$ in the group can calculate

TABLE 1. The parameters of encryption/decryption.

$Members$	$\Rightarrow$	$u_{1,k}$	$u_{2,k}$	$\cdots$	$u_{R,k}$	$(id_{j,u_k}, T_{j,k}, M_{i,k}, D_j)$
$u_{1,k}$	$\Rightarrow$	$\overline{M}_{1,1}$	$M_{1,2}$	$\cdots$	$M_{1,R}$	$(id_{1,u_k}, T_{1,k}, M_{1,k}, D_1)$
$u_{2,k}$	$\Rightarrow$	$M_{2,1}$	$\overline{M}_{2,2}$	$\cdots$	$M_{2,R}$	$(id_{2,u_k}, T_{2,k}, M_{2,k}, D_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$u_{R,k}$	$\Rightarrow$	$M_{R,1}$	$M_{R,2}$	$\cdots$	$\overline{M}_{R,R}$	$(id_{R,u_k}, T_{R,k}, M_{R,k}, D_R)$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$Parameters$	$\Rightarrow$	M_1	M_2	$\cdots$	M_R	(id_j, T_j, M_j, D_j)

$\delta_{i,k} = m_{i,k}g_1$, $\Gamma = \sum_{j=1}^{R} Pk_{u_{j,k}}$, and it can calculate the group decryption key $dk^{\pi}_{u_{i,k}} = Sk^{-1}_{u_{i,k}}\Omega_{i,k} = \sum_{j=1}^{R} m_{j,k}g_1$ and group encryption key

$$ek^{\pi}_{u_{i,k}} = (\prod_{j=1,j\neq i}^{R} e(T_{j,k}, (PK_{CA_j} + PK_{alli}))).e(Pk_{u_{i,k}}, \delta_{i,k}) = \prod_{1=j}^{R} e(Pk_{u_{j,k}}, \delta_{j,k})$$

(5) Key consistency verification. After calculating the group encryption key and decryption key, each member $u_{j,k}$ verifies the correctness of these group keys by equation $ek^{\pi}_{u_{j,k}} = (dk^{\pi}_{u_{j,k}}, \Gamma)$. If it is correct, the DC-AAGKA is terminated successfully. Otherwise, $u_{j,k}$ should calculate again or report error.

(6) Instance. For any plaintext $m \in \mathcal{M}^*$ ($\mathcal{M}^*$: plaintext space), each $u_{j,k}(1 \leqslant j \leqslant R, 1 \leqslant k \leqslant n)$ with the group encryption key $ek^{\pi}_{u_{j,k}}$ and group decryption key $dk^{\pi}_{u_{j,k}}$ operates as the follows:

Encryption: $u_{j,k}$ chooses a random number $t_{j,k} \in \mathbb{Z}_p^*$, and calculate $\overline{U} = t \sum_{1\leqslant j\leqslant R, 1\leqslant k\leqslant n} Pk_{j,k}$, $\overline{V} = m \oplus H_4((ek^{\pi}_{u_{j,k}})^t)$, Then it broadcasts ciphertext $(\overline{U}, \overline{V})$.

Decryption: After receiving a ciphertext $(\overline{U}, \overline{V})$, anyone can calculate $m = \overline{V} \oplus H_4(e(\overline{U}, dk^{\pi}_{u_{j,k}}))$ with a valid $dk^{\pi}_{u_{j,k}}$.

D. GROUP KEY UPDATE PROTOCOL FOR NODES JOIN GROUP

When a new node want to join the existing group, the join phase occurs and in this case, the existing group should provide the fairness group key formation to the new node. However, it should ensure that none of new member can calculate any of previous group decryption key. Assuming the existing group is $U = \{u_{1,k}, u_{2,k}, \ldots, u_{R,k}\}, (1 \leq k \leq n)$, and low-power node $u_{j,i}, (1 \leq j \leq R, 1 \leq i \leq n, i \neq k)$ wish to join this group. The new set of mobile nodes is $U' = U \cup \{u_{j,i}\} = \{u_{1,k}, u_{2,k}, \ldots, u_{R,k}, u_{j,i}\}$. Let member $u_{R,k}$ in the group be the sponsor of updating group key, the steps are as follows:

(1) $u_{R,k}$ selects a number $m'_{R,k} \in \mathbb{Z}_p^*$ randomly, and calculates $T'_{R,k} = m'_{R,k}r_{u_{R,k}}$, $M'_{j,k} = m'_{R,k}Pk_{u_{j,k}}, (1 \leq j \leq R-1)$ and $M_{R,k} = m_{R,k}Pk_{u_{R,k}}$, then it broadcasts the message $(id_{R,u_k}, T'_{R,k}, M'_{j,k}, D_R)$ to other old members and proclaims that there is a new member to want to join the group.

(2) After receiving the messages, each old member $u_{j,k}, (1 \leq j \leq R-1, 1 \leq k \leq n)$ calculates the parameter $\overline{\delta'_{j,k}} = Sk^{-1}_{u_{j,k}}M'_{j,k} = m'_{R,k}g_1$ and verifies the identity of $u_{R,k}$ by equation $e(T'_{R,k}, (PK_{CA_R} + PK_{alli})) = e(Pk_{u_{R,k}}, \overline{\delta'_{j,k}})$. If it is hold, $u_{j,k}$ can ensure that the messages are sent by $u_{R,k}$ and then it calculates $\theta_{R,k} = m'_{R,k}g_1 + dk^{\pi}_{u_{j,k}}$, otherwise reports error.

(3) The new member $u_{j,i}$ chooses a number $m'_{j,i} \in \mathbb{Z}_p^*$ randomly, it calculates $T'_{j,i} = m'_{j,i}r_{u_{j,i}}$, $M'_{t,k} = m'_{j,i}Pk_{u_{t,k}}, (t = 1, 2, \ldots, R, i \neq k)$ and broadcasts the messages $(T'_{j,i}, M'_{t,k})$ to all the group members.

(4) After receiving the messages broadcasted by $u_{j,i}$, each member $u_{t,k}, (t = 1, 2, \ldots, R, i \neq k)$ calculates $\overline{\delta'_{t,k}} = Sk^{-1}_{u_{t,k}}M'_{t,k} = m'_{j,i}g_1$ and verifies the identity of $u_{j,i}$ by equation $e(T'_{j,i}, (PK_{CA_j} + PK_{alli})) = e(Pk_{u_{j,i}}, \overline{\delta'_{t,k}})$. If it is hold, $u_{t,k}$ can ensure that the message are sent by $u_{j,i}$, and then it calculates $\chi_{t,k} = \overline{\delta'_{t,k}}$, otherwise reports error.

(5) $u_{R,k}$ chooses a number $\lambda'_{R,k} \in \mathbb{Z}_p^*$ randomly and calculates $\beta_{R,k} = \theta_{R,k} + \lambda g_1, \alpha_{R,k} = \lambda Pk_{u_{j,i}}$, then it sends $(\alpha_{R,k}, \beta_{R,k})$ to the new member $u_{j,i}$.

(6) Calculation group decryption key. All the old members can calculate the new group decryption key $dk^{\pi'}_{u_{t,k}} = \theta_{t,k} + \chi_{t,k} = m'_{R,k}g_1 + dk^{\pi}_{u_{t,k}} + m'_{j,i}g_1$; and the new member $u_{j,i}$ can calculate $\varsigma_{u_{j,i}} = Sk^{-1}_{u_{j,i}}\alpha_{R,k}$ and then calculates the new group decryption key $dk^{\pi'}_{u_{j,i}} = \beta_{R,k} - \varsigma_{u_{j,i}} + m'_{j,i}g_1 = dk^{\pi'}_{u_{t,k}}$.

(7) Calculation group encryption key. all the members can calculate the new group encryption key $ek^{\pi'}_{u_{j,k}} = e(T'_{R,k}, (PK_{CA_R} + PK_{alli})) \times e(T'_{j,i}, (PK_{CA_j} + PK_{alli})) \times ek^{\pi}_{u_{j,k}}$.

E. GROUP KEY UPDATE PROTOCOL FOR NODES EXIT GROUP

When a set of nodes wish to leave the group, the remove phase occurs. In this case, for protecting the security communication of the group, either creation of new group keys or the modification of the existing group keys is necessary. In this section, we propose a modification of the existing group key in such a way that none of the leaving user can calculate the subsequent group key generated. Let the set of existing group be $U = \{u_{1,k}, u_{2,k}, \ldots, u_{R,k}\}, (1 \le k \le n)$. For convenience, assume that only one member $u_{l,k}, (1 \le l \le n)$ wish to exit this group. The new set of mobile nodes set is $U'' = U - \{u_{l,k}\} = \{u_{1,k}, u_{2,k}, \ldots, u_{l-1,k}, \ldots, u_{R,k}\}$. Let member $u_{R,k}$ in the group be the sponsor of updating group key, the proposed protocol for implementing the remove phases is as follows.

(1) $u_{l,k}$ broadcasts the messages $(id_{l,u_k}, T_{l,k}, M_{i,k}, D_l)$, $(1 \le i \le R, i \ne l)$ to the group and declares to exit this group.

(2) $u_{R,k}$ calculates $\overline{\delta''_{R,k}} = Sk^{-1}_{u_{R,k}} M'_{R,k} = m'_{l,k} g_1$ and verifies the identify of $u_{l,k}$ by the equation $e(T_{l,k}, (PK_{CA_l} + PK_{alli})) = e(Pk_{u_{l,k}}, \overline{\delta''_{R,k}})$. If it is hold, $u_{R,k}$ can ensure that the messages are sent by $u_{l,k}$. Then $u_{R,k}$ selects a number $m''_{R,k} \in \mathbb{Z}^*_p$ randomly, and calculates $T''_{R,k} = m''_{R,k} r_{u_{R,k}}, M''_{j,k} = m'_{R,k} Pk_{u_{j,k}}, (1 \le j \le R-1, j \ne l)$, $\overline{M}''_{R,k} = m''_{R,k} Pk_{u_{R,k}}$, $\Omega''_{R,k} = \Omega_{R,k} - M_{R,k}$, $\Gamma''_{R,k} = \Gamma_{R,k} - Pk_{u_{l,k}} = \sum_{t=1, t\ne l}^{R} Pk_{u_{t,k}}$, and then broadcasts the messages $(id_{R,u_k}, T''_{R,k}, M''_{j,k}, D_R)$ to the remainder members and declares to update the existing group keys.

(3) After receiving the messages from $u_{R,k}$ and $u_{l,k}$, each remainder member $u_{j,k}, (1 \le j \le R-1, j \ne l)$ calculates $\overline{\delta''_{j,k}} = Sk^{-1}_{j,k} M''_{j,k} = m''_{R,k} g_1$ and verifies the identity of $u_{R,k}$ by the equation $e(T''_{R,k}, (PK_{CA_R} + PK_{alli})) = e(Pk_{u_{R,k}}, \overline{\delta''_{j,k}})$. If it is hold, $u_{j,k}$ calculates $\Omega''_{j,k} = \Omega_{j,k} - M_{j,k}$ and $\Gamma''_{j,k} = \Gamma_{j,k} - Pk_{u_{l,k}} = \sum_{t=1, t\ne l}^{R} Pk_{u_{t,k}}$. Otherwise, reports error.

(4) Calculation group decryption key. All the remainder members $u_{j,k}, (1 \le j \le R-1, j \ne l)$ can calculate $\phi_{j,k} = Sk^{-1}_{u_{j,k}} \Omega''_{j,k}$, and then calculate the group decryption key $dk^{\pi''}_{u_{j,k}} = \phi_{j,k} + \overline{\delta''_{j,k}}$; the sponsor $u_{R,k}$ can calculate the group decryption key $dk^{\pi''}_{u_{R,k}} = Sk^{-1}_{u_{R,k}} (\Omega''_{R,k} + \overline{M''_{R,k}})$.

(5) calculation groups encryption key. All the remainder members $u_{j,k}, (1 \le j \le R-1, j \ne l)$ can calculate the group encryption key $ek^{\pi''}_{u_{j,k}} = e(T''_{j,k}, \phi_{j,k}) \times e(T''_{R,k}, (PK_{CA_j} + PK_{alli}))$; the sponsor $u_{R,k}$ can calculate group encryption key $ek^{\pi''}_{u_{R,k}} = e(dk^{\pi''}_{u_{R,k}}, \Gamma''_{R,k} + Pk_{u_{R,k}})$.

IV. CORRECTNESS AND SECURITY ANALYSIS

A. CORRECTNESS ANALYSIS

If all the participants calculate the group keys correctly by the group key agreement protocol, then they are able to negotiate a pair of group encryption/decryption keys, and they can also decrypt any ciphertext messages encrypted by group encryption key with there group decryption key. The proofs of the correctness of the DC-AAGKA are as the following theorems.

Theorem1: If the equation $e(T_{j,k}, (PK_{CA_j} + PK_{alli})) = e(Pk_{u_{j,k}}, \overline{\delta_{i,k}})$ is hold, then any members of group can ensure that messages $(D_j, T_{j,k}, M_{i,k}, id_{j,u_k})$ are sent by $u_{j,k}$.

Proof: Since $r_{u_{j,k}} = (Sk_{u_{j,k}} / (SK_{CA_j} + SK_{alli}))g_1$, $T_{j,k} = m_{j,k} r_{u_{j,k}}$, $\overline{\delta}_{i,k} = Sk^{-1}_{i,k} M_{j,k} = m_{j,k} g_1$ and the properties of the bilinear pairings, there are:

$$
\begin{aligned}
&e(T_{j,k}, (PK_{CA_j} + PK_{alli})) \\
&= e(m_{j,k} r_{u_{j,k}} g_1, (SK_{CA_j} + SK_{alli}) g_1) \\
&= e(m_{j,k} (Sk_{u_{j,k}} / (SK_{CA_j} + SK_{alli})) g_1, (SK_{CA_j} + SK_{alli}) g_1) \\
&= e(m_{j,k} Sk_{u_{j,k}} g_1, g_1) \\
&= e(m_{j,k} g_1, Sk_{u_{j,k}} g_1) \\
&= e(Pk_{u_{j,k}},)
\end{aligned}
$$

The above proof shows that the message is signed by $u_{j,k}$, so that it can prove that the message is sent by the member $u_{j,k}$.

Theorem 2: By running the DC-AAGKA protocol, all participants can establish consistent group keys, and the group keys are contribution group keys. (1) An identical group decryption key has been established by all the group members.

Proof: Since $M_{i,k} = m_{j,k} Pk_{u_{i,k}}$, each member $u_{i,k}$ receives other member's message $M_{j,k} (1 \le j \le R, j \ne i)$, and then $u_{i,k}$ can calculate $\Omega_{i,k} = \sum_{j=1, j\ne i}^{R} M_{j,k} + \overline{M}_{i,k} = \sum_{j=1}^{R} M_{j,k}$, and then it can calculate the group decryption key as follows:

$$
\begin{aligned}
dk^{\pi}_{u_{i,j}} &= Sk^{-1}_{u_{i,j}} \Omega_{i,k} = Sk^{-1}_{u_{i,j}} \sum_{j=1}^{R} M_{j,k} \\
&= Sk^{-1}_{u_{i,j}} \sum_{j=1}^{R} m_{j,k} Pk_{u_{i,k}} = Sk^{-1}_{u_{i,j}} \sum_{j=1}^{R} m_{j,k} Sk_{u_{i,j}} g_1 \\
&= \sum_{j=1}^{R} m_{j,k} g_1 = \sum_{i=1}^{R} m_{i,k} g_1 = dk^{\pi}_{u_{j,k}}.
\end{aligned}
$$

From above equation, each member can calculate a fixed value for the group decryption key finally. The group decryption key includes all the members' key parameter $m_{j,k} g_1$, so the group decryption key is a contribution group key.

(2) An identical group encryption key has been established by all the group members.

Proof: When each member $u_{i,k}$ receives other members' broadcast message $(D_j, T_{j,k}, M_{i,k}, id_{j,u_k})$, then all the participants can calculate an identical group encryption key. Since $r_{u_{j,k}} = (Sk_{u_{j,k}} / (SK_{CA_j} + SK_{alli}))g_1$, $T_{j,k} = m_{j,k} r_{u_{j,k}}$, $\delta_{j,k} = m_{j,k} g_1$, and the properties of the bilinear pairings,

we have:

$$\begin{aligned} ek^{\pi}_{u_{i,k}} &= (\prod_{1=j,j\neq i}^{R} e(T_{j,k}, (PK_{CA_j} + PK_{alli}))) \\ &\cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j,j\neq i}^{R} e(m_{j,k} r_{u_{j,k}} g_1, (SK_{CA_j} + SK_{alli}) g_1)) \\ &\cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j,j\neq i}^{R} e(m_{j,k}(Sk_{u_{j,k}} / (SK_{CA_j} + SK_{alli})) g_1, \\ &(SK_{CA_j} + SK_{alli}) g_1) \cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j,j\neq i}^{R} e(m_{j,k} Sk_{u_{j,k}} g_1, g_1)) \cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j,j\neq i}^{R} e(m_{j,k} g_1, Sk_{u_{j,k}} g_1)) \cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j,j\neq i}^{R} e(\delta_{j,k}, Pk_{u_{j,k}})) \cdot e(Pk_{u_{i,k}}, \delta_{i,k}) \\ &= (\prod_{1=j}^{R} e(\delta_{j,k}, Pk_{u_{j,k}}) \end{aligned}$$

B. CORRECTNESS ANALYSIS

Theorem 3: for an instance $(G_1, G_2, q, g_1, ag_1, abg_1, e, H_1, H_2)$ of ICDH problem, we construct a Polynomial time simulator $\mathcal{C}$, and solve the problem by utilizing attacker $\mathcal{A}$. Suppose if it is possible to attack DC-AAGKA protocol successfully, it can be proved that $\mathcal{C}$ are able to solve the IDH problem with non-ignorable advantages. Firstly, we suppose that there are members want to participate DC-AAGKAprotocol, adversary $\mathcal{A}$ requires q_{H_1} H_1, requires q_{H_2} H_2, queries to *Ek.Reveal* for q_E times, queries to *Dk.Reveal* for q_D times and to *Corrupt* for q_C times, also the maximum number of the protocol that $\mathcal{A}$ could set up is $q_A$in the attack experimentnt. Adversary $\mathcal{A}$ win this game with advantage $Adv(\mathcal{A}) = \varepsilon$, so then there is an algorithm solving the ICDH problem exists with advantage

$$\varepsilon' = \varepsilon \frac{q^2_{H_1} q_{H_2} \mu^4 (q_{H_1} + q_{H_2})^2}{q_A q_{C_1} q_D q_E n}$$

ICDH problem.

Proof: Suppose $\mathcal{C}$ is a challenge, and $\mathcal{A}$ is an adversary who can decipher the protocol. Given an instance $(G_1, G_2, q, g_1, ag_1, abg_1, e, H_1, H_2)$ of $\mathcal{C}$, unknown number $a, b \in \mathbb{Z}^*_p$, and h is open hash function, H_1 and H_2 are two oracle machines controlled by $\mathcal{C}$. $\mathcal{C}$ solve the ICDH problems by utilizing $\mathcal{A}$. Define a new session, it will have an unique session ID. Challenger $\mathcal{C}$ chooses one session randomly, whose ID is sid_t. ID of corresponding simulated session is $\mathcal{C}_{sid_t}$, and thus the probability that the session is selected is $pr[\mathcal{C}_{sid_t} = 0] = \omega$, and corresponding $pr[\mathcal{C}_{sid_t} = 1] = 1 - \omega$.At the same time, challenger $\mathcal{C}$ records two-tuples $(sid_t, \mathcal{C}_{sid_t})$.

Initialization Phase: When the game starts, $\mathcal{C}$ chooses system parameter $params = (G_1, G_2, q, g_1, ag_1, abg_1, e, H_1, H_2)$ and keep two positive integers $SK_{CA_i}, SK_{alli} \in \mathbb{Z}^*_p$, calculates $PK_{CA_j} = SK_{CA_i} g_1, PK_{alli} = SK_{alli} g_1$, then $\mathcal{C}$ sends the *params* to adversary $\mathcal{A}$.

Training phase. Interactions between challenger and adversary are as following:

$KG(u_{i,j})$: Simulate user register key generation. $\mathcal{C}$ maintains an initially empty list $\mathcal{C}_{L1}$, $\mathcal{C}_{H^i_1} \in \{0, 1\}$ denotes the type of user (where $\mathcal{C}_{H^i_1} = 1$ denotes $\mathcal{C}$ can not calculate the register key of the user, $\mathcal{C}_{H^i_1} = 0$ denotes $\mathcal{C}$ can calculate the register key of the user, the probability that calculate the register key is $pr[\mathcal{C}_{H^i_1} = 0] = \mu$, and corresponding $pr[\mathcal{C}_{H^i_1} = 1] = 1 - \mu$). $\mathcal{A}$ chooses two positive integers $sk_{i,j}, \gamma_{i,j} \in \mathbb{Z}^*_p$ randomly and the identity id_{i,u_j} of $u_{i,j}$, then it calculates the parameters $s_{u_{i,j}} = H_1(id_{i,u_j})$, $Sk_{u_{i,j}} = \gamma_{i,j} s_{u_{i,j}}$, $Pk_{u_{i,j}} = Sk_{u_{i,j}} g_1$, $\mathcal{A}$ queries H_1 on $(id_{i,u_j}, Sk_{u_{i,j}}, Pk_{u_{i,j}})$, $\mathcal{C}$ does the following:

If there is a tuple $(id_{i,u_j}, Sk_{u_{i,j}}, Pk_{u_{i,j}}, r_{u_{i,j}}, \mathcal{C}_{H^i_1})$ on $\mathcal{C}_{L1}$, return $r_{u_{i,j}}$ as the answer. Else if $\mathcal{C}_{H^i_1} = 0$, $\mathcal{C}$ calculates $r_{u_{i,j}} = (Sk_{u_{i,j}} / (SK_{CA_i} + SK_{alli})) g_1$, add $(id_{i,u_j}, Sk_{u_{i,j}}, Pk_{u_{i,j}}, r_{u_{i,j}}, \mathcal{C}_{H^i_1})$ to $\mathcal{C}_{L1}$ and return $r_{u_{i,j}}$ as the answer. Otherwise if $\mathcal{C}_{H^i_1} = 1$, add $(id_{i,u_j}, Sk_{u_{i,j}}, Pk_{u_{i,j}}, null, \mathcal{C}_{H^i_1})$ to $\mathcal{C}_{L1}$ and return *null* as the answer.

Else if q_{H_1} is more then the length of $\mathcal{C}_{L1}$, it outputs symbol $\perp$ (Event 1).

$Send(\Pi^{\pi}_{u_{i,j}}, m)$. $\mathcal{C}$ reserve an empty initialization list $\mathcal{C}_{L2}$. Suppose $pid^{\pi}_{u_{i,j}} = \{id_{u_{i,1}}, id_{u_{i,2}}, \ldots, id_{u_{i,n}}\}$, $\mathcal{C}$ initializes and defines a session identity $sid^{\pi}_{u_{i,j}}$ to a session $\mathcal{C}_{sid^{\pi}_{u_{i,j}}}$. Then id_{i,u_j} submits to oracle machine H_2. $\mathcal{C}$ starts to simulate oracle as following:

(1) If $\mathcal{C}_{sid^{\pi}_{u_i}} = 0$ and $\mathcal{C}_{H^i_1} = 0$, perform the following steps: $\mathcal{C}$ chooses $m_{i,k} (1 \leq k \leq n) \in \mathbb{Z}^*_p$ randomly, obtain $(r_{u_{i,j}}, Pk_{u_{i,j}})$ from list $\mathcal{C}_{L1}$ and calculates $T_{i,j} = m_{i,k} r_{u_{i,j}}$ and $M_{i,j} = m_{i,k} Pk_{u_{i,j}}$. Then, add $(id_{i,u_j}, T_{i,k}, M_{i,j}, Pk_{u_{i,j}}, PK_{CA_j}, PK_{alli}, \mathcal{C}_{sid^{\pi}_{u_i}}, \mathcal{C}_{H^i_1})$ to list $\mathcal{C}_{L2}$ and respond with $(id_{i,u_j}, T_{i,j}, M_{i,j}, Pk_{u_{i,j}}, PK_{CA_j}, PK_{alli})$.

(2) If $\mathcal{C}_{sid^{\pi}_{u_i}} = 0$ and $\mathcal{C}_{H^i_1} = 1$, perform the following steps: $\mathcal{C}$ chooses $m^{*'}_{i,k} (1 \leq k \leq n) \in \mathbb{Z}^*_p$ randomly, obtain $(null, Pk_{u_{i,j}})$ from list $\mathcal{C}_{L1}$ and calculates $M^{*'}_{i,j} = m^{*'}_{i,k} Pk_{u_{i,j}}$. Then, add $(id_{i,u_j}, null, M^{*'}_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli}, \mathcal{C}_{sid^{\pi}_{u_i}}, \mathcal{C}_{H^i_1})$ to list $\mathcal{C}_{L2}$ and respond with $(id_{i,u_j}, null, M^{*'}_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli})$.

(3) If $\mathcal{C}_{sid^{\pi}_{u_i}} = 1$ and $\mathcal{C}_{H^i_1} = 0$, perform the following steps: $\mathcal{C}$ chooses $m^{*}_{i,k} (1 \leq k \leq n) \in \mathbb{Z}^*_p$ randomly, obtain $(r_{u_{i,j}}, Pk_{u_{i,j}})$ from list $\mathcal{C}_{L1}$ and calculates $T^{*}_{i,j} = m^{*}_{i,k} r_{u_{i,j}}$ and $M^{*}_{i,j} = m^{*}_{i,k} Pk_{u_{i,j}}$. Then, add $(id_{i,u_j}, T^{*}_{i,j}, M^{*}_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli}, \mathcal{C}_{sid^{\pi}_{u_i}}, \mathcal{C}_{H^i_1})$ to list $\mathcal{C}_{L2}$ and respond with $(id_{i,u_j}, T^{*}_{i,j}, M^{*}_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli})$.

(4) If $\mathcal{C}_{sid^{\pi}_{u_i}} = 1$ and $\mathcal{C}_{H^i_1} = 1$, perform the following steps: $\mathcal{C}$ chooses $m'_{i,k} (1 \leq k \leq n) \in \mathbb{Z}^*_p$ randomly, obtain $(null, Pk_{u_{i,j}})$ from list $\mathcal{C}_{L1}$ and calculates $M'_{i,j} = m'_{i,k} Pk_{u_{i,j}}$.

Then, add $(id_{i,u_j}, null, M'_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli}, \mathcal{C}_{sid^{\pi}_{u_i}} \mathcal{C}_{H^i_1})$ to list $\mathcal{C}_{L2}$ and respond with $(id_{i,u_j}, null, M'_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli})$.

Ek.Reveal$(\Pi^{\pi}_{u_{i,j}})$. If $state^{\pi}_{u_{i,j}}$ ends successfully, else if $\mathcal{C}_{sid^{\pi}_{u_i}} = 0$ and $\mathcal{C}_{H^i_1} = 0$, according to the list $\mathcal{C}_{L1}$, $\mathcal{C}$ get $Sk_{u_{i,j}}$ and get $(T_{i,k}, M_{i,k}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli})$ from $\mathcal{C}_{L2}$, and then calculates $\delta_{i,j} = Sk^{-1}_{u_{i,j}} M_{i,j}$ and $ek_{i,j} = \prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta_{k,j})$, returns $\prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta_{k,j})$.

Else if $\mathcal{C}_{sid^{\pi}_{u_i}} = 1$ and $\mathcal{C}_{H^i_1} = 0$ according to the list $\mathcal{C}_{L1}$, $\mathcal{C}$ get $Sk_{u_{i,j}}$ and get $(T^*_{i,j}, M^*_{i,j}, Pk_{u_{i,k}}, PK_{CA_j}, PK_{alli})$ from $\mathcal{C}_{L2}$, and then calculates $\delta^*_{i,j} = Sk^{-1}_{u_{i,j}} M^*_{i,j}$ and $ek^*_{i,j} = \prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta^*_{i,j})$, returns $\prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta^*_{i,j})$. Otherwise, it outputs symbol $\perp$ Event 2.

Dk.Reveal$(\Pi^{\pi}_{u_{i,j}})$. If $state^{\pi}_{u_{i,j}}$ do not end successfully, it returns *null*.

Else if $\mathcal{C}_{sid^{\pi}_{u_i}} = 0$ and $\mathcal{C}_{H^i_1} = 0$, according to the list $\mathcal{C}_{L1}$, $\mathcal{C}$ get $Sk_{u_{i,j}}$ and get $M_{i,k}(1 \le k \le n)$ from $\mathcal{C}_{L2}$, and then calculates $\Omega_{i,j} = \sum_{k=1}^{n} M_{i,k}$, $dk^{\pi}_{u_{i,j}} = Sk^{-1}_{u_{i,j}} \Omega_{i,j} = \sum_{k=1}^{n} m_{i,k} g_1$, returns $dk^{\pi}_{u_{i,j}}$.

Else if $\mathcal{C}_{sid^{\pi}_{u_i}} = 1$ and $\mathcal{C}_{H^i_1} = 0$, according to the list $\mathcal{C}_{L1}$, $\mathcal{C}$ get $Sk_{u_{i,j}}$ and get $M^*_{i,k}(1 \le k \le n)$ from $\mathcal{C}_{L2}$, and then calculates $\Omega^*_{i,j} = \sum_{k=1}^{n} M^*_{i,k}$, $dk^{\pi *}_{u_{i,j}} = Sk^{-1}_{u_{i,j}} \Omega^*_{i,j} = \sum_{k=1}^{n} m^*_{i,k} g_1$, returns $dk^{\pi *}_{u_{i,j}}$. Otherwise, it outputs symbol $\perp$ Event 3.

Corrupt$(u_{i,j})$. After receiving query from *Corrupt*, $\mathcal{C}$ find the corresponding identity id_{i,u_j} of $u_{i,j}$ in $\mathcal{C}_{L1}$ firstly. If $\mathcal{C}$ cannot find the identity, it means this is the first time to query. Then as querying oracle machine H_1, $(id_{i,u_j}, Sk_{u_{i,j}}, Pk_{u_{i,j}}, r_{u_{i,j}}, \mathcal{C}_{H^i_1})$ will be added into $\mathcal{C}_{L1}$ through related process. If $\mathcal{C}_{H^i_1} = 1$, outputs symbol $\perp$(Event 4), if $\mathcal{C}_{sid^{\pi}_{u_i}} = 0$, returns $(dk^{\pi}_{u_{i,j}}, \prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta_{k,j}))$. Otherwise, returns $(dk^{\pi *}_{u_{i,j}}, \prod_{k=1}^{R} e(Pk_{u_{k,j}}, \delta^*_{i,j}))$.

Test$(\Pi^{\pi}_{u_i})$. After adversary $\mathcal{A}$ ends related query, it will challenge $\mathcal{C}$, $\mathcal{A}$ chooses a fresh oracle machine $\Pi^{\pi''}_{u_{i,j}}$ and two message m_0, m_1, and $|m_0| = |m_1|$. Suppose $pid^{''\pi}_{u_{i,j}} = \{d''_{u_{i,1}}, d''_{u_{i,2}}, \ldots, d''_{u_{i,n}}\}$, $\mathcal{C}$ responses as following: 1) $\mathcal{C}$ search the related information $(id^{''}_{i,u_j}, Sk^{''}_{u_{i,j}}, Pk^{''}_{u_{i,j}}, r^{''}_{u_{i,j}}, \mathcal{C}^{''}_{H^i_1})$ in list $\mathcal{C}_{L1}$; 2) If $\mathcal{C}^{''}_{H^i_1} = 0$, then extract information $(id^{''}_{i,u_j}, T^{''}_{i,k}, M^{''}_{i,j}, Pk^{''}_{u_{i,j}}, PK_{CA_j}, PK_{alli}, \mathcal{C}^{''}_{sid^{\pi}_{u_i}}, \mathcal{C}^{''}_{H^i_1})$ from list $\mathcal{C}_{L2}$, and calculate $(dk^{\pi''}_{u_{i,j}}, \prod_{k=1}^{R} e(Pk^{''}_{u_{k,j}}, \delta^{''}_{i,j}))$, Otherwise, output symbol $\perp$(Event 5); 3) $\mathcal{C}$ chooses a random number $b \in \{0,1\}$, $t^* \in \mathbb{Z}^*_p$ to calculate $U^* = t^* \sum_{1 \le j \le n} Pk^{''}_{u_{i,j}}$, then encrypt $m_b = V^* \oplus H_4(e(U^*, dk^{\pi''}_{u_{i,j}}))$; Return $c^* =< U^*, V^* >$ to $\mathcal{A}$.

Response. After $\mathcal{A}$ finishes all queries, it returns a $b' \in \{0,1\}$ as an assumption to b. If $\mathcal{A}$ query H_2 to get information $(id^{''}_{i,u_j}, T^{''}_{i,k}, M^{''}_{i,j}, Pk^{''}_{u_{i,j}}, PK_{CA_j}, PK_{alli}, \mathcal{C}^{''}_{sid^{\pi}_{u_i}}, \mathcal{C}^{''}_{H^i_1})$ in the list $\mathcal{C}_{L2}$ and it can guess $b' = b$ correctly with some advantage. That means get $\mathcal{A}$ query H_2 to get information $(id^{''}_{i,u_j}, T^{''}_{i,k}, M^{''}_{i,j}, Pk^{''}_{u_{i,j}}, PK_{CA_j}, PK_{alli}, \mathcal{C}^{''}_{sid^{\pi}_{u_i}}, \mathcal{C}^{''}_{H^i_1})$ in the list $\mathcal{C}_{L2}$, and calculates $\Omega^{''}_{i,j} = \sum_{k=1}^{n} M^{''}_{i,k}$, $dk^{\pi''}_{u_{i,j}} = Sk^{-1}_{u_{i,j}} \Omega^{''}_{i,j} = \sum_{k=1}^{n} m^{''}_{i,k} g_1$, $\mathcal{A}$ get the advantage to obtain $m_b = V^* \oplus H_4(e(U^*, dk^{\pi''}_{u_{i,j}}))$. Then $\mathcal{C}$ can calculate ICDH problem, according to $\mathcal{A}$'s method, because set $abg_1 = \Omega^{''}_{i,j}$, $Pk^{''}_{u_{i,j}} = ag_1$. According to system parameter (g_1, ag_1, bg_1), $dk^{\pi''}_{u_{i,j}} = Sk^{-1}_{u_{i,j}} \Omega^{''}_{i,j} = \sum_{k=1}^{n} m^{''}_{i,k} g_1$ that means $\mathcal{C}$ can calculate value of $bg_1 = (ab/a) g_1$.

lemma 1: If $\mathcal{C}$ does not end the simulation game above, adversary $\mathcal{A}$ cannot distinguish the environment of simulation world from of real world.

Proof: All the outputs in the simulation game described above conform to the rules of the DC-AAGKA protocol, and the messages generated by the entity conform to the uniform distribution of the message space, so adversary cannot perceive the inconsistency between the simulated world and the real world. The probability of success to $\mathcal{C}$ is:

$$
\begin{aligned}
&pr[\mathcal{C}wins] \\
&= pr[\overline{Event1} \wedge \overline{Event2} \wedge \overline{Event3} \\
&\quad \wedge \overline{Event4} \wedge \overline{Event5} \wedge \mathcal{A}wins] \\
&= pr[\mathcal{A}wins \,|\, \overline{Event1} \wedge \overline{Event2} \wedge \overline{Event3} \wedge \overline{Event4} \wedge \overline{Event5}] \\
&\quad pr[\overline{Event5} \,|\, \overline{Event1} \wedge \overline{Event2} \wedge \overline{Event3} \wedge \overline{Event4}] \\
&\quad pr[\overline{Event4} \,|\, \overline{Event1} \wedge \overline{Event2} \wedge \overline{Event3}] \\
&\quad pr[\overline{Event3} \,|\, \overline{Event1} \wedge \overline{Event2}] pr[\overline{Event2} \,|\, \overline{Event1}] \\
&\quad \times pr[\overline{Event1}] \\
&= \varepsilon \cdot \frac{q_{H_2}\mu}{q_A} \cdot \frac{q_{H_1}\mu}{q_C} \cdot \frac{(q_{H_1}+q_{H_2})\omega\mu + (q_{H_1}+q_{H_2})(1-\omega)\mu}{q_D} \\
&\quad \times \frac{(q_{H_1}+q_{H_2})\omega\mu + (q_{H_1}+q_{H_2})(1-\omega)\mu}{q_E} \cdot \frac{q_{H_1}}{n} \\
&= \varepsilon \frac{q_{H_1} q_{H_2} \mu^2 ((q_{H_1}+q_{H_2})\omega\mu + (q_{H_1}+q_{H_2})(1-\omega)\mu)^2 q_{H_1}}{q_A q_{C_1} q_D q_E n} \\
&= \varepsilon \frac{q^2_{H_1} q_{H_2} \mu^2 ((q_{H_1}+q_{H_2})(\omega\mu + (1-\omega)\mu))^2}{q_A q_{C_1} q_D q_E n} \\
&= \varepsilon \frac{q^2_{H_1} q_{H_2} \mu^4 (q_{H_1}+q_{H_2})^2}{q_A q_{C_1} q_D q_E n}
\end{aligned}
$$

V. PERFORMANCE ANALYSIS

Because of limited resources of mobile terminals, during the process of designing protocol, except security, the computational time, the computational complexity, communication energy consumption and computation energy consumption

are important performance measures of the agreement. In this paper, we compared and analyzed the literature that can be quantified in recent years. All these protocols are suitable for mobile terminals. According to data from these protocols, comparative the analyses are from computational time, the complexity of calculation and communication, and protocol consumption of the total energy. Table 2 lists the comparison and analysis between the SC-AGKG protocol and other three group key agreement protocols in the calculation of complexity and traffic.

TABLE 2. Complexity analysis of the four protocols.

protocol	Xu et al.[19]	Zhang et al.[21]	Wei et al.[22]	DC-AAGKA
Exponent iations	$4n+3$	$6n+2$	$n+1$	0
Pairing	$6n$	$2n+3$	$4n+1$	$2n$
Scalar Multiplications	$7n+1$	$5n+1$	$4n$	$3n+2$
Sent Messages	$n+5$	$n+2$	$n+1$	$n+2$
Received Messages	n^2+5n	$3n-1$	$n+1$	$n+2$

From Table1, Xu *et al.* [19] proposed scheme have the lowest computational complexity and their communication complexity. Qikun *et al.* [21] and Wei *et al.* [22] have the high computational complexity. Qikun *et al.* [21], Wei *et al.* [22], and DC-AAGKA have the similar of communication complexity. As the time of processing the protocol shows, analyzing with the data from [22], which ran the related algorithm through program pbc-0.5.12 provides by PBC lab, on environment of Intel(R) Core(TM)2 Duo E8400 CPU(3.00GHz), ubuntu 10.04,the average run time of the multiplication on is 0.016 ms, and the average exponent operation time of and are 3.886 ms and 0.489 ms, respectively, and the average running time of the bilinear pair is 4.354 ms, as shown in Table 3.

TABLE 3. Time cost of some algorithms.

Algorithm	Time cost
Scalar Multiplication Over G_1	0.016ms
Modular Exponentiation Over G_1	3.886ms
Modular Exponentiation OverG_2	0.489ms
Tate Pairing	4.354ms

Based on the above calculation complexity and related algorithm operation time analysis, comparison of the time consumption of results between DC-AAGKA and previous three kinds of research is shown in Fig.1

In this section, we perform the total energy consumption cost analysis of performing GKA using the data provided in [25]. The total energy cost of each GKA protocol is simply the sum of the computation and communication cost, according to Tan *et al.* [21], a 133 MHz "Strong ARM" microprocessor consumes 9.1 mJ for performing a modular exponentiation, 8.8 mJ for performing a scalar multiplication, 47.0 mJ for a Tate pairing, 8.8 mJ for performing a Elliptic Curve Digital Signature Algorithm and 10.9 mJ

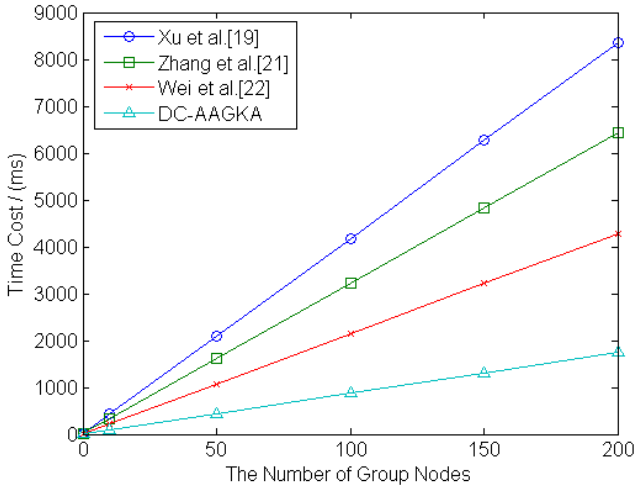

FIGURE 1. Time analysis as a function of applied field. It is good practice to explain the time cost of the protocol in the caption.

for performing a Elliptic Curve Digital Signature Verify Algorithm. As for the communication energy cost, according again to (Makri), an IEEE 802.11 Spectrum24 WLAN card consumes 0.00066 mJ for the transmission of 1 bit and 0.00031 mJ for the reception of 1 bit. The above mentioned energy costs will be used for the performance evaluation of the examined GKA protocols and are summarized in Table 4.

TABLE 4. Energy costs for computation and communication.

Type of Communication	Energy Costs/mJ
Computation cost of Modular Exponentiation	9.1
Computation cost of Scalar Multiplication	8.8
Computation cost of Tate Pairing	47
Communication cost for transmitting a bit	0.00066
Communication cost for receiving a bit	0.00031

The DC-AAGKA protocol is compared with the related works [19], [21], [22] in communication costs, computation costs and other items. These agreements conclude "One-Round Affiliation-Hiding Authenticated Asymmetric Group Key Agreement with Semi-trusted Group Authority" proposed by Xu *et al.* [19], the "identity-based authenticated asymmetric group key agreement protocol" proposed by Qikun *et al.* [21] and "no valid certificate authenticated asymmetric group key agreement protocol" proposed by Wei *et al.* [22]. Based on the data provided by [25], the computing and communication complexity of these agreements are analyzed in Table 1. We compared the DC-AAGKA protocol with other two protocols in computation cost. The result is shown in Fig.2.

In Fig.2, the computation cost of these three group key agreement protocols is presented. In terms of computation, the most efficient protocol is DC-AAGKA. A higher computation cost is proposed by Wei *et al.*'s protocol [22], while the highest computation cost is inferred by Xu *et al.*'s protocol [19]. To compute the communication cost, we need to know the size of the information exchange. We assume

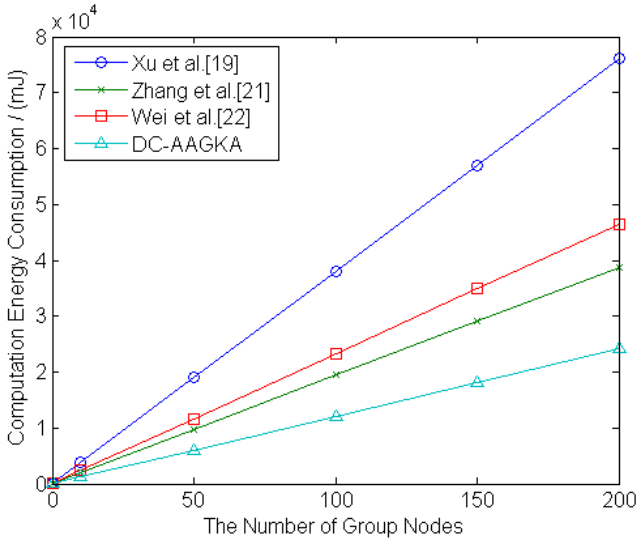

FIGURE 2. Computation energy consumption analysis as a function of applied field. It is good practice to explain the computation cost of the protocol in the caption.

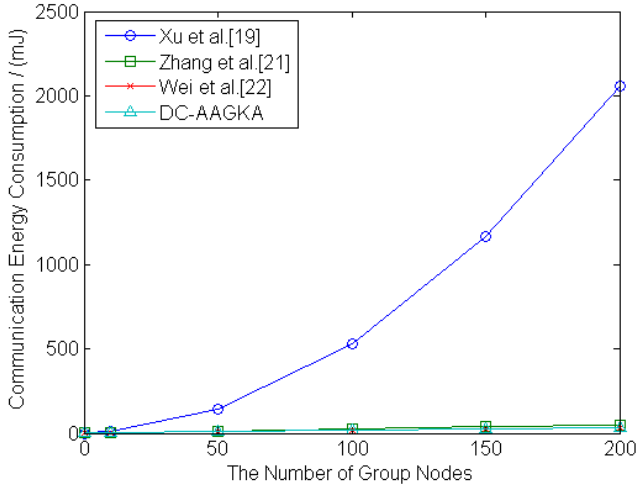

FIGURE 3. Communication cost analysis as a function of applied field. It is good practice to explain the communication consumption of the protocol in the caption.

that the secure key length is 160b of the elliptic curve cryptography. All these three protocols adopt the elliptic curve cryptography, so their key length is 160b. Let the length of the information exchange be 160b, and the communication consumption is shown in Fig.3.

The communication consumption of the examined Protocols are depicted in Fig.3. Xu *et al.*'s protocol [19] has the worst performance, followed by Qikun *et al.*'s protocol [21], Wei *et al.*'s protocol [22] and DC-AAGKA Protocol, are very efficient in terms of communication, bringing a very similar cost. The total energy consumption analysis is shown in fig.4, the total energy consumption increases rapidly with the nodes increased in Xu *et al.*'s protocol [19] and Wei *et al.*'s protocol [22]. DC-AAGKA protocol has a really low total energy cost, which makes its performance exceptional.

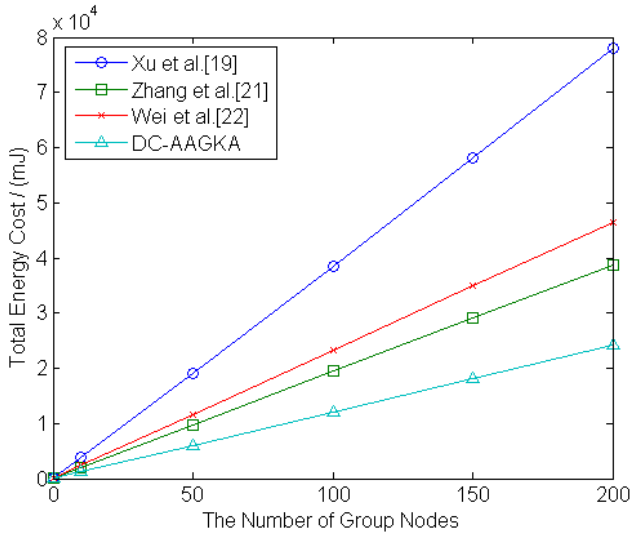

FIGURE 4. Total energy analysis as a function of applied field. It is good practice to explain the total energy consumption of the protocol in the caption.

VI. CONCLUSION

The paper analyses the security need of telemedicine telecommunications systems. For meet the security collaborative computing and security information sharing among the members in medical corps, A DC-AAGKA protocol is proposed. The group keys also have been implemented efficiently in cryptographic systems to provide confidential and privacy. The paper proposed the corresponding dynamic key update scheme. It supports the members leave one group or join another group frequently. DC-AAGKA has both forward secrecy and backward secrecy. It proven that the safety and energy cost performance of DC-AAGKA has good advantages. It is suitable for security group communication in telemedicine

REFERENCES

[1] M. A. Murillo-Escobar, L. Cardoza-Avendaño, and R. M. López-Gutiérrez, "A double chaotic layer encryption algorithm for clinical signals in telemedicine," *J. Med. Syst.*, vol. 41, no. 4, pp. 1–17, 2017, doi: 10.1007/s10916-017-0698-3.

[2] D. Yin, W. Huanzhen, and Z. Zixia, "Research on medical image encryption in telemedicine systems," *Technol. Health Care*, vol. 24, no. s2, pp. S435–S442, Jun. 2016.

[3] O. Mir and M. Nikooghadam, "A secure biometrics based authentication with key agreement scheme in telemedicine networks for E-health services," *Wireless Pers. Commun.*, vol. 83, no. 4, pp. 2439–2461, Aug. 2015.

[4] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement," in *Proc. EUROCRYPT*, Cologne, Germany, 2009, pp. 153–170.

[5] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014.

[6] X. Zhang *et al.*, "Cryptographic key protection against FROST for mobile devices," *Cluster Comput.*, vol. 20, no. 3, pp. 2393–2402, Sep. 2017.

[7] J. H. Davis and J. R. Cogdell, "Calibration program for the 16-foot antenna," Electr. Eng. Res. Lab., Univ. Texas, Austin, TX, USA, Tech. Rep. NGL-006-69-1, Nov. 1987.

[8] R. S. Ranjani, "Lagrange interpolation based asymmetric group key agreement cryptosystem," *Int. J. Eng. Technol.*, vol. 8, no. 2, pp. 635–646, Apr./May 2016.

[9] C. Y. Yeun, K. Han, D. L. Vo, and K. Kim, "Secure authenticated group key agreement protocol in the MANET environment," *Inf. Secur. Tech. Rep.*, vol. 13, no. 3, pp. 158–164, Mar. 2008.
[10] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.
[11] Z. Xingwen, Z. Fangguo, and T. Haibo, "Dynamic asymmetric group key agreement for ad hoc networks," *Ad Hoc Netw.*, vol. 9, no. 4, pp. 928–939, May 2011.
[12] M. Bilal and S.-G. Kang, "A secure key agreement protocol for dynamic group," *Cluster Comput*, vol. 20, no. 3, pp. 2779–2792, Sep. 2017.
[13] M. S. Farash and M. A. Attari, "A secure and efficient identity-based authenticated key exchange protocol for mobile client–server networks," *J. Supercomput.*, vol. 69, no. 1, pp. 395–411, Jul. 2014.
[14] H. Zhu, Y.-A. Tan, X. Zhang, L. Zhu, C. Zhang, and J. Zheng, "A round-optimal lattice-based blind signature scheme for cloud services," *Future Generat. Comput. Syst.*, vol. 73, pp. 106–114, Aug. 2017.
[15] X. Lv, H. Li, and B. Wang, "Authenticated asymmetric group key agreement based on certificateless cryptosystem," *Int. J. Comput. Math.*, vol. 91, no. 3, pp. 447–460, Apr. 2014.
[16] H. Zhu and Y. Zhang, "An efficient chaotic maps-based deniable authentication group key agreement protocol," *Wireless Pers. Commun.*, vol. 96, no. 1, pp. 217–229, Sep. 2017.
[17] R. S. Ranjani, D. L. Bhaskari, and P. S. Avadhani, "An extended identity based authenticated asymmetric group key agreement protocol," *Int. J. Netw. Secur.*, vol. 17, no. 5, pp. 510–516, Sep. 2015.
[18] L. Zhang, Q. Wu, B. Qin, J. Domingo-Ferrer, and Ú. González-Nicolás, "Asymmetric group key agreement protocol for open networks and its application to broadcast encryption," *Comput. Netw.*, vol. 55, no. 15, pp. 3246–3255, Oct. 2011.
[19] C. Xu, L. Zhu, Z. Li, and F. Wang, "One-round affiliation-hiding authenticated asymmetric group key agreement with semi-trusted group authority," *Comput. J.*, vol. 58, no. 10, pp. 2509–2519, Oct. 2015.
[20] C. Xu, H. Guo, Z. Li, and Y. Mu, "Affiliation-hiding authenticated asymmetric group key agreement based on short signature," *Comput. J.*, vol. 57, no. 10, pp. 1580–1590, Oct. 2014.
[21] Q. Zhang, Q. Zhang, Z. Ma, and Y. Tan, "An authenticated asymmetric group key agreement for imbalanced mobile networks," *Chin. J. Electron.*, vol. 23, no. 4, pp. 827–835, Oct. 2014.
[22] G. Wei, X. Yang, and J. Shao, "Efficient certificateless authenticated asymmetric group key agreement protocol," *KSII Trans. Internet Inf. Syst.*, vol. 6, no. 12, pp. 3352–3364, Dec. 2012.
[23] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, and Z. Dong, "Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2352–2364, Nov. 2015.
[24] M. Li, X. Xu, C. Guo, and X. Tan, "AD-ASGKA—Authenticated dynamic protocols for asymmetric group key agreement," *Secur. Commun. Netw.*, vol. 9, no. 11, pp. 1340–1352, Jul. 2016.
[25] E. Makri and E. Konstantinou, "Constant round group key agreement protocols: A comparative study," *Comput. Secur.*, vol. 30, no. 8, pp. 643–678, Oct. 2011.

ZHANG QIKUN was born in Henan, China, in 1980. He received the B.S. degree from Xidian University in 2004, the M.S. degree from the Lanzhou University of Technology in 2008, and the Ph.D. degree from the Beijing Institute of Technology in 2013. He is currently an Associate Professor with the Department of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, China. His research interests include information security and cryptography.

GAN YONG was born in Zhengzhou, China, in 1965. He is currently a Ph.D. Professor with the School of Computer and Communication Engineering, Zhengzhou University of Light Industry. His research interests include multimedia communications, image processing, coding, and network engineering.

ZHANG QUANXIN was born in Shandong, China, in 1974. He received the B.S. degree from the Department of Computer Science, Shandong University of Technology, in 1997, the M.S. degree from the School of Electronic and Information Engineering, Lanzhou Jiaotong University, in 2000, and the Ph.D. degree from the School of Computer Science, Beijing Institute of Technology, in 2003. He is currently a Faculty Member with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include the ad hoc network and mobile computing.

WANG RUIFANG was born in Henan, China, in 1982. She received the B.S. degree from Xidian University in 2004 and the M.S. degree from the Lanzhou University of Technology in 2008. Her research interests include information security and cryptography.

TAN YU-AN was born in Chongqing, China, in 1972. He is currently a Ph.D. Supervisor and a Professor with the Beijing Institute of Technology. His current research interests include information security and network storage. He is a Senior Member of the China Computer Federation.