Efficient Certificateless Authenticated Asymmetric Group Key Agreement Protocol

Guiyi Wei, Xianbo Yang and Jun Shao*
Colleage of Computer and Information Engineering, Zhejiang Gongshang University
Room 420, SCIE Building, Zhejiang Gongshang University
18# Xuezheng Street, Hangzhou, Zhejiang province, 310018, P.R.China
[e-mail: weigy@zjgsu.edu.cn, xianboyang85@gmail.com, chn.junshao@gmail.com]
*Corresponding author: Jun Shao

Received September 15, 2012; revised November 16, 2012; accepted December 10, 2012; published December 27, 2012

Abstract

Group key agreement (GKA) is a cryptographic primitive allowing two or more users to negotiate a shared session key over public networks. Wu et al. recently introduced the concept of asymmetric GKA that allows a group of users to negotiate a common public key, while each user only needs to hold his/her respective private key. However, Wu et al.'s protocol can not resist active attacks, such as fabrication. To solve this problem, Zhang et al. proposed an authenticated asymmetric GKA protocol, where each user is authenticated during the negotiation process, so it can resist active attacks. Whereas, Zhang et al.'s protocol needs a partially trusted certificate authority to issue certificates, which brings a heavy certificate management burden. To eliminate such cost, Zhang et al. constructed another protocol in identity-based setting. Unfortunately, it suffers from the so-called key escrow problem. In this paper, we propose the certificateless authenticated asymmetric group key agreement protocol which does not have certificate management burden and key escrow problem. Besides, our protocol achieves known-key security, unknown key-share security, key-compromise impersonation security, and key control security. Our simulation based on the pairing-based cryptography (PBC) library shows that this protocol is efficient and practical.

Keywords: Certificateless, authenticated, asymmetric group key agreement, bilinear map

This work was partially supported by 973 project No. 2012CB315804, NSFC No.61003308, QJD1102009, ZJNSF LR12F02002, the Program for Zhejiang Leading Team of Science and Technology Innovation, the Science and Technology Planning Projects of Zhejiang Grants 2010C33045 and 2011C13006-1, and SRF for ROCS, SEM. This work was also industrially supported by the National Development and Reform Commission, China under Special Grants "The Operation System of Multimedia Cloud Based on the Integration of Telecommunications Networks, Cable TV Networks and the Internet (Wasu Media Network Co.) ".

http://dx.doi.org/10.3837/tiis.2012.12.018

1. Introduction

1.1 Related Work

With the rapid development of global information technology, computer network provides great convenience for people's communication and information transmission. However, it also puts forward a huge challenge on the information security protection at the same time. On many occasions, it's necessary for users to communicate with others through a confidential communication channel. Group key agreement (GKA) is such a cryptographic primitive to realize this communication channel. It refers to two (or more) users in the insecure public communication channel negotiate a shared session key to achieve the purpose of secure communication. GKA is widely used in many collaborative and distributed applications, such as multi-party computations, file distribution/sharing, audio/video conference and chat systems [1].

The first GKA protocol was proposed by Diffie and Hellman [2] in 1976. Their protocol is an one-round two users GKA protocol. While more than two users may be involved in the actual communication environments. There have been many attempts to extend the Diffie-Hellman protocol to n users case. Joux protocol [3] and Burmester-Desmedt protocol [4] are the well-known ones among them. In the conventional GKA protocol, all the group users establish a shared common secret key, and only the group users are allowed to send messages to other group users. Nevertheless, anyone is probably to be a potential sender in practice. Based on this reality, the concept of asymmetric group key agreement (AGKA) was introduced by Wu et al. [5] in 2009. By their definition, the group users negotiate a public key that is accessible to any user, but they keep respective private keys. In this way, anyone who knows the negotiated public key could be a sender.

However, Wu et al.'s protocol can only withstand passive attacks, where the attackers are only allowed to eavesdrop the communication channel. In the real world, attackers are usually the other so-called active attackers, who can control the communication channel and launch more powerful active attacks, such as masquerade, fabrication, message replay, message modification, and denial of service. For instance, an active attacker is able to launch well-known man-in-the-middle attacks. In fact, in order to realize secure key agreement in the open communication channel, the most important thing is to withstand these active attackers, while authentication is one of the best methods to solve this problem. Based on this idea, Zhang et al. proposed the notion of authenticated asymmetric group key agreement (AAGKA) [6] in 2011. Authentication, the act of confirming the truth of a user's identity, can ensure that the user is who he/she claims to be. Group key agreement protocols with authentication allow a group of users to negotiate a group key in the open networks, and ensure that only the participating user can obtain the negotiated key.

To the best of our knowledge, the authentication property of the existing AAGKA protocols is achieved in the following two cryptographic mechanisms: Public-key infrastructure (PKI) cryptography [6][7][8][9] and identity-based cryptography (IBC) [1]. In the PKI-based cryptography, each user has a public key, which is bound with the user by a certificate generated by a partially trusted certificate authority (CA). During the execution process of the AAGKA protocol, users can authenticate each other in an explicit or implicit way by using the certificates. This brings a heavy burden for the network due to the generation, distribution, preservation and verification of the certificate.

On the other hand, to eliminate the heavy burden of the certificate management, Shamir [10] proposed the concept of IBC. In this scenario, the public key of each user can be easily derived from its identity (e.g., its telephone number or email address), which is the distinguishing feature of IBC. The private key of each user is generated by a fully trusted authority called private key generator (PKG) who owns the master secret of the system and takes the user's identity as input. As a result, the certificate management burden is removed but it suffers from the key escrow problem. In particular, users' private keys are completely generated by the PKG, and all users must fully trust the PKG. Once the PKG is compromised, the whole system will collapse. This situation is called the key escrow problem. In other word, the trust level of PKG in IBC is higher than that of CA in PKI-based cryptography. See the details in Section 2.2.

In order to resolve these disadvantages, this paper focuses on AAGKA protocols which do not suffer from the above limitations. In particular, we introduce the idea of certificateless public key cryptography (CL-PKC) [11][12][13] into the research of AAGKA. In CL-PKC, the private key of the user is computed from the partial private key computed by a partially trusted authority called key generation center (KGC) and a secret value chosen by the user, and the corresponding public key is computed from the KGC's public parameters and the secret value. Hence, the key escrow problem and certificate management problem are solved in CL-PKC.

Based on the above idea, this paper constructs a specific certificateless authenticated asymmetric group key agreement (CL-AAGKA) protocol. It satisfies various typical security properties which are generally needed to achieve by group key agreement protocols. To the best of our knowledge, our proposed protocol is the first that employs the idea of CL-PKC to produce the group key in AAGKA. More importantly, it does not suffer from certificate management burden and key escrow problem. In many resource-constrained open network environments, this is a very valuable attribute. In addition, based on the Pairing-Based Cryptography library, we realize the proposed protocol with C language in the Linux operating system. The simulation results show that the time cost of the protocol is low, and the protocol is effective and practical.

1.2 Contributions

In this paper, we present the concept of CL-AAGKA by introducing certificateless public key cryptography into AAGKA. The CL-AAGKA protocol we proposed captures the typical security properties (i.e., known-key security and key control security) for AAGKA protocols. Due to the fact that the KGC does not know the full private key of each user (though it knows the partial private key), our protocol is free from the key escrow problem. This is unlike the previous AAGKA protocols realized by IBC mechanism [1]. In addition, the proposed protocol achieves a trust level similar to the traditional PKI-based AAGKA protocols but it does not suffer from the certificate management burden which seems to be a very useful property to build advanced systems for some applications, such as resource-constrained audio/video conference.

1.3 Paper Outline

The rest of the paper is organized as follows. Section 2 presents some preliminaries about this paper, such as bilinear maps, some complexity assumptions, the definition of CL-AAGKA, and some related security definitions. We propose our protocol and analyze its security properties in Section 3. In Section 4, we realize our protocol under the pairing-based

cryptography (PBC) library [14] and analyze its simulation results. Section 5 concludes the paper.

2. Preliminaries

In this section, we briefly introduce some involved basic knowledge of this paper.

2.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be multiplicative groups of prime order q, and g be a generator of $\mathbb{G}$. A map $\hat{e}: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is called a bilinear map if it satisfies the following properties:

1.Bilinearity: $\hat{e}(g^{\alpha}, g^{\beta}) = \hat{e}(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q^*$.

2.Non-degeneracy: There exist $\alpha \in \mathbb{G}, \beta \in \mathbb{G}$ such that $\hat{e}(\alpha, \beta) \neq 1$.

3.Computability: There exists an efficient algorithm to compute $\hat{e}(\alpha, \beta)$ for any$\alpha \in \mathbb{G}, \beta \in \mathbb{G}$.

2.2 Trust Levels

Based on the trust level on a third party, Girault [15] divided the third party into three different levels:

Level 1: The third party can not only know (or compute) users' private keys. but also impersonate any user at any time without being detected.

Level 2: Though the third party does not know (or can not compute) users' private keys. It can still impersonate any user by generating his/her false public key without being detected.

Level 3: The third party does not know (or can not compute) users' private keys. If it is to launch attacks by generating false public keys of users, it will expose its malicious behavior.

As we can see from above, the trust placed in the third party decreases as the level increases. PKI cryptography, IBC, and CL-PKC belong to level 3, level 1 and level 2, respectively. However, as stated in [11], CL-PKC can be easily promoted to level 3.

2.3 Complexity Assumptions

The security of our protocol is based on the hardness of Computational Diffie-Hellman (CDH) problem and Divisible Computational Diffie-Hellman (DCDH) problem. Let $\mathbb{G}$ be multiplicative group of prime order q, and g be a generator of $\mathbb{G}$. We briefly review them as follows [1].

CDH problem: Given g, g^{α}, g^{β} in $\mathbb{G}$ for unknown $\alpha, \beta \in \mathbb{Z}_q^*$, compute $g^{\alpha\beta}$.

CDH assumption: Let $\mathcal{B}$ be an algorithm that has advantage

$$\mathbf{Adv}(\mathcal{B}) = \Pr[\mathcal{B}(g, g^{\alpha}, g^{\beta}) = g^{\alpha\beta}]$$

in solving the CDH problem. The CDH assumption states that $\mathbf{Adv}(\mathcal{B})$ is negligible for any polynomial-time algorithm $\mathcal{B}$.

DCDH problem: Given g, g^{α}, g^{β} in $\mathbb{G}$ for unknown $\alpha, \beta \in \mathbb{Z}_q^*$, compute $g^{\alpha/\beta}$.

DCDH assumption: Let $\mathcal{B}$ be an algorithm which has advantage

$$\mathbf{Adv}(\mathcal{B}) = \Pr[\mathcal{B}(g, g^{\alpha}, g^{\beta}) = g^{\alpha/\beta}]$$

in solving the DCDH problem. The DCDH assumption states that $\mathbf{Adv}(\mathcal{B})$ is negligible for any polynomial-time algorithm $\mathcal{B}$.

2.4 Algorithms of Certificateless Authenticated Asymmetric Group Key Agreement (CL-AAGKA)

CL-AAGKA protocol consists of twelve polynomial-time algorithms: system setup (SS), partial private key extract (PPKE), set secret value (SSV), set secret key (SSK), set public key (SPK), billboard initialization (BI), joining (Join), leaving (Leave), group public key derivation (GPKD), group secret key derivation (GSKD), encryption (Enc), decryption (Dec). These algorithms are defined as follows.

SS: In this algorithm, the KGC takes a security parameter λ as input, and returns a pair of master-keys for itself and the system parameters $params$.

PPKE: This algorithm that is also performed by the KGC takes the system parameters $params$, master-keys and a user's identity ID_i as input, and outputs the user's partial private key D_i.

SSV: This algorithm that is run by user U_i takes the system parameters $params$ and his/her identity ID_i as input, and outputs his/her secret value v_i.

SSK: This algorithm that is run by user U_i takes the system parameters $params$, a partial private key D_i, a secret value v_i as input, and outputs his/her private key sk_i.

SPK: This algorithm that is run by user U_i takes the system parameters $params$, a secret value v_i with his/her identity ID_i as input, and outputs his/her public key pk_i.

BI: This algorithm permits the billboard (It consists of two tables called billboard part I and billboard part II, and contains uses' information for the key agreement) maintainer to output the initialized information of billboard part I and billboard part II while the input is the system parameters $params$.

Join: This algorithm permits user U_i to join the group when it is not full. It is run by the billboard maintainer who accepts the information submitted by U_i as input, and puts it onto billboard part II as output if all the validation equalities hold or rejects the joining request when some validation equalities do not hold.

Leave: This algorithm permits user U_i to leave the group. It is run by the billboard maintainer who accepts U_i's identity ID_i as input and deletes his/her information in billboard part II.

GPKD: With the messages published by all users in the billboard, any user can compute the group public key (gpk). This algorithm which is run by U_i takes its identity ID_i as input and outputs gpk.

GSKD: With the messages published by all users in the billboard, each user U_i can run this algorithm to compute his/her group private key gsk_i respectively. It inputs U_i's identity ID_i and outputs his/her group private key gsk_i.

Enc: Anyone who knows the group encryption key gpk can run this algorithm. This algorithm takes the message m from the message space, the group public key gpk and the system parameters $params$ as input and outputs the ciphertext c.

Dec: Only entitled user U_i with his/her decryption key gsk_i in the group can run this algorithm. This algorithm takes the ciphertext c and U_i's gsk_i as input and outputs the plaintext m.

2.5 Security Definitions

Several generally accepted desirable security properties for conventional AAGKA protocols have been defined [4][16][17]. We re-define these security properties as follows for analyzing the security of CL-AAGKA protocols:

- Known-key security: If the protocol does not fail, each user can compute his/her unique group private key and a unique common group public key. Known-key security guarantees that even if some group private keys corresponding to some sessions are leaked, this should not compromise the secrecy of the group private key corresponding to any other session.
- Unknown key-share security: An adversary should not make user A think that he/she is sharing a key with user B with whom he/she is not really sharing a key.
- Key-compromise impersonation security: The compromise of user A's long-term private key will allow an adversary to impersonate A, but it should not make the adversary be able to impersonate other users to A.
- Key control security: None of users is able to force the session key to be a preselected value.

3. The Proposed CL-AAGKA Protocol

In this section, we propose our concrete CL-AAGKA protocol from bilinear maps for dynamic broadcast system. A CL-AAGKA protocol involves a KGC, a group of users and a billboard maintainer which maintains the billboard in an authenticated way. The sender of messages in the system may be a user of the group or not. But only the group users can decrypt the encrypted messages in the system. We let λ be a security parameter given to the SS algorithm. Some parts of the following algorithms are similar with that of Zhang et al.'s scheme [6]. However, their scheme is PKI-based.

SS: This algorithm runs as follows:

1. Given security parameter λ, the KGC gets λ to generate $(\mathbb{G}, \hat{e}, g, q)$, where $\mathbb{G}$ is a multiplicative group of prime order q, $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map and g is an arbitrary generator of $\mathbb{G}$.
2. The KGC selects a random value $s \in \mathbb{Z}_q^*$ as the master private key msk and computes $g^s \in \mathbb{G}$ as the master public key mpk.
3. The KGC randomly selects $g_1, \ldots, g_\omega \in \mathbb{G}$, where ω is the largest group size that the system can support.
4. The KGC also chooses a cryptographic hash function: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}^*$. And the KGC publishes the system parameters $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, g_1, \ldots, g_\omega, q, mpk, H_1)$.

PPKE: This algorithm takes $params$ and U_i's identity $ID_i \in \{0,1\}^*$ as input, the KGC computes $Q_i = H_1(ID_i) \in \mathbb{G}^*$, and outputs the partial private key $D_i = Q_i^s$ for U_i. And U_i can check the correctness of the partial private key by verifying $\hat{e}(D_i, g) = \hat{e}(Q_i, mpk)$.

SSV: This algorithm takes $params$ and U_i's identity ID_i as input. It randomly selects $v_i \in \mathbb{Z}_q^*$ as U_i's secret value.

SSK: This algorithm takes $params$, U_i's partial private key D_i and secret value v_i as input, U_i constructs his/her private key as $sk_i = (D_i, v_i) = (Q_i^s, v_i)$.

`SPK`: This algorithm takes $params$, U_i's identity ID_i and secret value v_i as input. It constructs U_i's public key as $pk_i = (x_i, A_i) = (g^{-v_i}, \hat{e}(D_i, g))$.

`BI`: Assume that the broadcast system is of size N. Let the billboard maintainer (an extraordinary user of the system) be U_0 holding private key sk_0. U_0 initializes the billboard part I (**Table 1**) as follows:

1. For $1 \le i \le N$, randomly choose $D_i^0 \in \mathbb{G}, v_i^0 \in \mathbb{Z}_q^*$ and compute $x_i^0 = g^{-v_i^0}, A_i^0 = \hat{e}(D_i^0, g)$.
2. For $1 \le i, j \le N$, compute $\sigma_{i,j}^0 = D_i^0 g_j^{v_i^0}$.
3. Publish $\mathbb{L}_i = \{\sigma_{i,1}^0, \cdots, \sigma_{i,i-1}^0, null, \sigma_{i,i+1}^0, \cdots, \sigma_{i,N}^0, (x_i^0, A_i^0)\}$.

For the billboard part II, as shown in **Table 2**, U_0 sets each row to be $\mathbb{L}_i' = \{\overbrace{null, null, \cdots, null}^{N+1}\}, 0 \le i \le N$.

Table 1. Billboard Part I

Column	1	2	3	$\cdots$	n	n+1
$\mathbb{L}_1$	null	$\sigma_{1,2}^0$	$\sigma_{1,3}^0$	$\cdots$	$\sigma_{1,n}^0$	(x_1^0, A_1^0)
$\mathbb{L}_2$	$\sigma_{2,1}^0$	null	$\sigma_{2,3}^0$	$\cdots$	$\sigma_{2,n}^0$	(x_2^0, A_2^0)
$\mathbb{L}_3$	$\sigma_{3,1}^0$	$\sigma_{3,2}^0$	null	$\cdots$	$\sigma_{3,n}^0$	(x_3^0, A_3^0)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbb{L}_n$	$\sigma_{n,1}^0$	$\sigma_{n,2}^0$	$\sigma_{n,3}^0$	$\cdots$	null	(x_n^0, A_n^0)

Table 2. Billboard Part II

Column	1	2	3	$\cdots$	n	n+1
$\mathbb{L}_0'$	null	null	null	$\cdots$	null	null
$\mathbb{L}_1'$	null	null	null	$\cdots$	null	null
$\mathbb{L}_2'$	null	null	null	$\cdots$	null	null
$\mathbb{L}_3'$	null	null	null	$\cdots$	null	null
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbb{L}_n'$	null	null	null	$\cdots$	null	null

`Join`: When a user U_i with a secret-public key pair (sk_i, pk_i)=$((D_i, v_i),(g^{-v_i}, e(D_i, g)))$ wants to join the system, he/she first checks whether there is a row with all null elements on billboard part II. If there is no such row, it means the broadcast system is full. Otherwise, assuming that the index of that row is i for simplicity, he/she does the following:

1. For $1 \le j \le N$, compute $\sigma_{i,j} = D_i g_j^{v_i}$.
2. Submit $\{ID_i, \sigma_{i,1}, \cdots, \sigma_{i,i-1}, null, \sigma_{i,i+1}, \cdots, \sigma_{i,N}, (x_i, A_i)\}$ to the billboard maintainer.

When the billboard maintainer receives $\{ID_i, \sigma_{i,1}, \cdots, \sigma_{i,i-1}, null, \sigma_{i,i+1}, \cdots, \sigma_{i,N}, (x_i, A_i)\}$, he/she first verifies whether all the equalities $\hat{e}(\sigma_{i,j}, g)\hat{e}(g_j, g^{-v_i}) = \hat{e}(D_i, g)$ hold; If they are valid, the billboard maintainer sets $\mathbb{L}_i' = \{\sigma_{i,1}, \cdots, \sigma_{i,i-1}, null, \sigma_{i,i+1}, \cdots, \sigma_{i,N}, (x_i, A_i)\}$ and $\mathbb{L}_{0,i}' = \{ID_i\}$. Otherwise, he/she refuses this message.

`Leave`: Suppose that a user U_i leaves the system and his/her published message is in the ith row for simplicity. The billboard maintainer sets $\mathbb{L}_i' = \{\overbrace{null, null, \cdots, null}^{N+1}\}$ and $\mathbb{L}_{0,i}' = \{null\}$.

GPKD: Let $\mathbb{S}$ be the set of indices such that $\mathbb{L}_i' = \{\overbrace{null, null, \cdots, null}^{N+1}\}$ and $\bar{\mathbb{S}}$ be the set of indices such that $\mathbb{L}_i' \neq \{\overbrace{null, null, \cdots, null}^{N+1}\}$ on the billboard part II, respectively. Any user can compute the group public key $gpk = (x, A)$, where:$x = \prod_{j\in\mathbb{S}} x_j^0 \prod_{j\in\bar{\mathbb{S}}} x_j, A = \prod_{j\in\mathbb{S}} A_j^0 \prod_{j\in\bar{\mathbb{S}}} A_j$. If all the equalities $\hat{e}(\sigma_{i,j}, g)\hat{e}(g_j, g^{-v_i}) = \hat{e}(D_i, g)$ hold, he/she accepts the gpk.

GSKD: To derive the group private key gsk_j, user U_j computes $gsk_j = \prod_{i\in\mathbb{S}} \sigma_{i,j}^0 \prod_{i\in\bar{\mathbb{S}}} \sigma_{i,j}$. If all the equalities $\hat{e}(\sigma_{i,j}, g)\hat{e}(g_j, g^{-v_i}) = \hat{e}(D_i, g)$ hold, user U_j accepts the gsk_j.

Enc: Anyone who knows the system parameters $params$ and the group public key $gpk = (x, A)$ can encrypt any plaintext m from the message space as follows:

1. Select a random $t \in \mathbb{Z}_q^*$.
2. Compute $c_1 = g^t, c_2 = x^t, c_3 = m \cdot A^t$.
3. Output the ciphertext $c = (c_1, c_2, c_3)$.

Dec: To decrypt the ciphertext $c = (c_1, c_2, c_3)$, user U_j in the system does the following:

$$m = \frac{c_3}{\hat{e}(gsk_j,\ c_1) \cdot \hat{e}(g_j,\ c_2)}$$

The correctness of the decryption can be obtained by the following.

$$\begin{aligned}
& \frac{c_3}{\hat{e}(gsk_j,\ c_1) \cdot \hat{e}(g_j,\ c_2)} \\
= & \frac{m \cdot A^t}{\hat{e}(\prod_{i\in\mathbb{S}} \sigma_{i,j}^0 \prod_{i\in\bar{\mathbb{S}}} \sigma_{i,j},\ g^t) \cdot \hat{e}(g_j,\ x^t)} \\
= & \frac{m \cdot (\prod_{i\in\mathbb{S}} A_i^0 \prod_{i\in\bar{\mathbb{S}}} A_i)^t}{\hat{e}(\prod_{i\in\mathbb{S}} \sigma_{i,j}^0 \prod_{i\in\bar{\mathbb{S}}} \sigma_{i,j},\ g^t) \cdot \hat{e}(g_j,\ (\prod_{i\in\mathbb{S}} x_i^0 \prod_{i\in\bar{\mathbb{S}}} x_i)^t)} \\
= & \frac{m \cdot (\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g))^t}{\hat{e}(\prod_{i\in\mathbb{S}} (D_i^0 g_j^{v_i^0}) \prod_{i\in\bar{\mathbb{S}}} (D_i g_j^{v_i}),\ g^t) \cdot \hat{e}(g_j,\ (\prod_{i\in\mathbb{S}} g^{-v_i^0} \prod_{i\in\bar{\mathbb{S}}} g^{-v_i})^t)} \\
= & \frac{m \cdot (\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g))^t}{\hat{e}(\prod_{i\in\mathbb{S}} D_i^0 \prod_{i\in\mathbb{S}} g_j^{v_i^0} \prod_{i\in\bar{\mathbb{S}}} D_i \prod_{i\in\bar{\mathbb{S}}} g_j^{v_i},\ g^t) \cdot \hat{e}(g_j,\ (\prod_{i\in\mathbb{S}} g^{-v_i^0} \prod_{i\in\bar{\mathbb{S}}} g^{-v_i})^t)} \\
= & \frac{m \cdot (\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g))^t}{\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g^t) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g^t)\hat{e}(\prod_{i\in\mathbb{S}} g_j^{v_i^0} \prod_{i\in\bar{\mathbb{S}}} g_j^{v_i},\ g^t)\hat{e}(g_j,\ (\prod_{i\in\mathbb{S}} g^{-v_i^0} \prod_{i\in\bar{\mathbb{S}}} g^{-v_i})^t)} \\
= & \frac{m \cdot (\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g))^t}{(\prod_{i\in\mathbb{S}} \hat{e}(D_i^0, g) \prod_{i\in\bar{\mathbb{S}}} \hat{e}(D_i, g))^t \hat{e}(g_j,\ \prod_{i\in\mathbb{S}} g^{v_i^0 t} \prod_{i\in\bar{\mathbb{S}}} g^{v_i t})\hat{e}(g_j,\ (\prod_{i\in\mathbb{S}} g^{-v_i^0} \prod_{i\in\bar{\mathbb{S}}} g^{-v_i})^t)} \\
= & \ m
\end{aligned}$$

3.1 Security Analysis

In this section, we analyze that our protocol achieves all the security properties which are defined in Section 2.4.

1. Known-key security: Since v_i is randomly selected by each user U_i within our protocol, the group public key and the group private keys in one session are assigned uniformly and independently to the corresponding keys in other sessions. As a result, if the group private key(s) for one session leaked, it would not compromise the group private key(s) of any other session. The protocol satisfies the known-key security property explicitly.

2. Unknown key-share security: In our protocol, the billboard maintainer would check $\hat{e}(\sigma_{i,j}, g)\hat{e}(g_j, g^{-v_i}) = \hat{e}(D_i, g)$. If v_i is not generated by U_i, these equalities will not hold. It implies that all users must be the real users as claimed in a successful run of our protocol. That is to say, the shared session keys must be held by true users as they claimed to be. Therefore, our protocol follows the unknown key-share security property, and achieves security against man-in-the-middle attacks.

3. Key-compromise impersonation security: Due to verifiable equalities $\hat{e}(\sigma_{i,j}, g) \cdot \hat{e}(g_j, g^{-v_i}) = \hat{e}(D_i, g)$, if U_i's private key sk_i is corrupted by an adversary, it still can not simulate another user U_j with the private key of U_i. It indicates that the key-compromise impersonation security property follows.

4. Key control security: Since each user has an input for computing the keys in the protocol, it is explicit that only the user who submits his/her message lastly could have a chance to control the keys. Generally speaking, we assume the last user is U_n and the preselected gpk is (x', A'). With knowing (x', A') and $\prod_{i=1}^{n-1} x_i, \prod_{i=1}^{n-1} A_i$, to compute (x_n, A_n),$U_n$should solve the DCDH problem. As the same to this, it has to solve the DCDH problem to derive a group private key. That is, U_n can not make a group public key or a group private key preselected, and the key control security property holds.

5. A secure group key agreement protocol should guarantee that only the group users can decrypt the ciphertexts, which is an important and essential requirement in group key agreement protocols. In our CL-AAGKA protocol, only user U_i who knows his/her $^{\sigma_{i,i}}$ which is not been published in the Billboard Part II can compute his/her corresponding arbitrary group private key gsk_i. Anyone else who did not take part in the process of negotiation can not compute an effective group private key. In other words, only users who have the group private keys can decrypt the ciphertexts in the proposed protocol. Our CL-AAGKA protocol enjoys this useful and essential requirement.

4. Simulation Results Analysis and Performance Evaluation

4.1 Simulation Results Analysis

In this section, we present the results of the simulation to show the proposed protocol's practical performance. We have realized the protocol by using the PBC library [14] (the library version was pbc-0.5.12) on a personal compute equipped with the Intel(R) Core(TM)2 Duo E8400 CPU that runs at a frequency of 3.00 GHz. The operating system of our simulation platform is ubuntu-10.04. We have chosen the type A pairing parameters from the $param$ subdirectory of the decompressed PBC archive to initialize our system. Refer to the PBC manual, the pairing of type A is a symmetric pairing. The pre-computations such as the

definition and initiation of variables are omitted in the performance analysis. Each result is an average value of running the program for 1,000 times.

The results show that the average time cost of a multiplication operation in $\mathbb{G}$ is 0.016 ms. while, the average time cost of a multiplication operation in $\mathbb{G}_T$ is 0.003 ms. The average time cost of an exponentiation operation in $\mathbb{G}$ is 3.886 ms. The average time cost of an exponentiation operation in $\mathbb{G}_T$ is 0.489 ms. And the average time cost of a pairing operation in $\mathbb{G}_T$ is 4.354 ms. We summarize the above results in **Table 3**. For type A pairing, because the author of the PBC library does only present the test results of average pairing time with the library version of pbc-0.4.7 on a 1GHz Pentium III hardware platform, there is no direct comparability with our simulation results. Nevertheless, based on the reality that the equipment configuration and the version of the PBC library in our simulation are higher than that in the author's test, the result that the average pairing time in our simulation is faster than that of the author provided is entirely reasonable.The average time costs of computing gpk (gpk-T in **Fig. 1**) and gsk_i (gsk-T in **Fig. 1**) are shown in **Fig. 1**, from which we can see that the average time costs for one user to compute the group public key and the group private key grow linearly as the system capacity N increases. When the system capacity N is 100, the average time costs of computing gpk and gsk_i are still only 2.23 ms and 5.89 ms respectively, which shows the system is still reasonably efficient. In many scenarios of practical applications, the vast majority of group member number is less than 100. Based on this reality, the maximum system capacity in our simulation which is set as 100 is reasonable and practical.

Table 3. The time costs of basic operations in our simulation

Operation	Time cost
multiplication in $\mathbb{G}$	0.016 ms
multiplication in $\mathbb{G}_T$	0.003 ms
exponentiation in $\mathbb{G}$	3.886 ms
exponentiation in $\mathbb{G}_T$	0.489 ms
pairing operation	4.354 ms

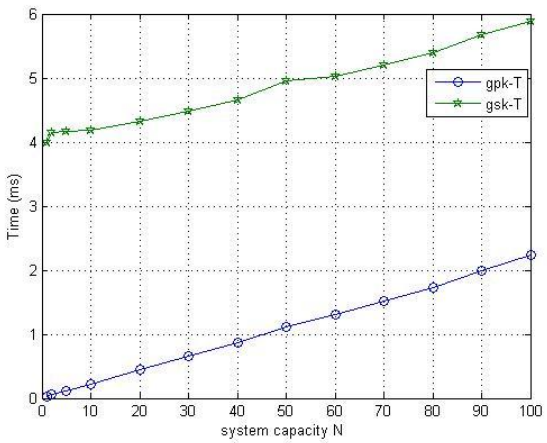


Fig. 1. The relationships between gpk and system capacity N and between gsk and system capacity N

The average time cost for a user to do an encryption operation (enc-T in **Fig. 2**) is 8.332 ms. Correspondingly, the time cost for a user to do a decryption operation (dec-T in **Fig. 2**) is 8.848 ms. They are constant regardless of the system capacity N. The relationships between the system capacity N and the encryption/decryption operation are shown in **Fig. 2**. Besides, the average time cost of an encryption operation or a decryption operation is reasonable, and it can be rarely perceived by human beings. Therefore, the simulation results indicate that our protocol is practical and efficient.

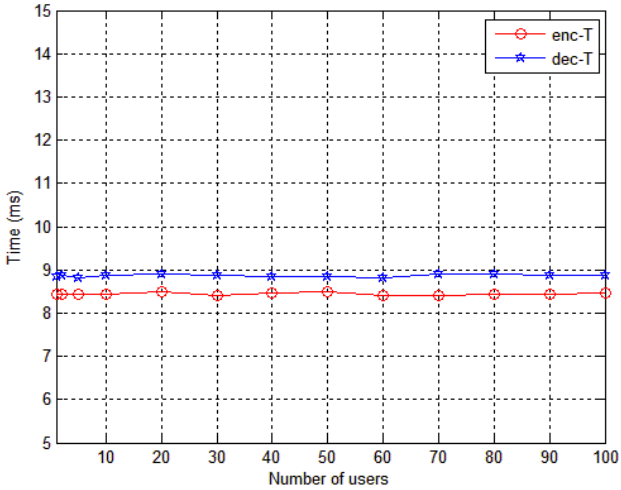


Fig. 2. The relationships between the system capacity N and the encryption/decryption operation.

4.2 Performance Evaluation

In this section, we evaluate the performance of our proposed CL-AAGKA protocol. In our evaluation, we always presume that the group size in our protocol is N.

Our protocol has low storage overhead. Because any one who wants to send or receive a message in the group has no need to store all the information published on the billboard in our protocol. In particular, a user who wants to send a message to the group only needs to store the $N+1th$ column on the billboard part I and the billboard part II. On the other hand, user U_i in our protocol who wants to receive messages in the group, he/she needs to store the information of the ith and $N+1th$ columns on the billboard part I and the billboard part II, respectively.

In most of the existing secure GKA protocols (e.g., [8]), it needs two or more rounds to negotiate a shared session key. In other words, all users in those protocols need to be connected concurrently. In contrast, our protocol only requires one round like the protocol in [6], which means that each user only needs to transport the message once in our protocol. The one-round feature in GKA protocols implies several advantages [3] compared to two or more rounds. For instance, a group of users who wish to share their confidential files via the open networks should be online at the same time in a two-round GKA protocol. Nevertheless, it is hard for distributed users to be online concurrently in practice (especially when the users of the protocol live in different time zones). Hence, our protocol with one-round feature is very desirable in practice.

In our protocol, a user of the protocol who wants to compute the group public key gpk, he/she only needs to verify $N-1$ equalities and compute $2N$ group multiplication operations, without having to verify any signature. On the other hand, user U_i in the group only needs to verify $N-1$ equalities, and compute N group multiplication operations in computing the group private key. We compare the above results with other two existing secure GKA protocols (D.B in [8] and Z.W.Q.D in [6]) in **Table 4**. From **Table 4**, though one may notice that our CL-AAGKA (W.Y.S in **Table 4**) protocol has a comparable overhead with the protocol in [6], our protocol does not need to verify any signature. Thanks to this property, our protocol does not suffer from heavy certificate management burden. Besides, it does not have the inherent private key escrow problem as that in the ID-based protocol.

Table 4. Protocol comparison

		D.B	Z.W.Q.D	W.Y.S
Gpk	Exp1	—	0	0
	Exp2	—	0	0
	Mul	—	2N	2N
	Ver-sign	—	N	0
	Equ-ch	—	0	N-1
Gsk	Exp1	3	0	0
	Exp2	0	0	0
	Mul	2N-2	N	N
	Ver-sign	N+1	N	0
	Equ-ch	N	0	N-1
Enc	Exp1	—	2	2
	Exp2	—	1	1
	Mul	—	1	1
Dec	Div	—	1	1
	Mul	—	1	1
	Pairing	—	2	2
Rounds		2	1	1
Cert-man-bur		Yes	Yes	No
Pri-key-esc		No	No	No

The following notations are used in our summarization:

Gpk：The process of computing group public key.

Gsk：The process of computing group private key.

Enc：The process of encryption.

Dec：The process of decryption.

Rounds：Total number of rounds.

Cert-man-bur：Whether suffering from certificate management burden or not.

Pri-key-esc：Whether suffering from private key escrow or not.

Exp1：Maximum number of exponentiations in $\mathbb{G}$ computed by a user.

Exp2：Maximum number of exponentiations in $\mathbb{G}_T$ computed by a user.

Mul：Maximum number of multiplications in $\mathbb{G}$ and $\mathbb{G}_T$ computed by a user.

Div：Maximum number of division in $\mathbb{G}$ and $\mathbb{G}_T$ computed by a user.

Pairing：Maximum number of pairing in $\mathbb{G}$ and $\mathbb{G}_T$ computed by a user.

—：There is no such process in the protocol.

In addition to some advantages similar to those of the GKA protocol in [6], we show that our protocol has some other special performance. Because of the certificateless public key mechanism, our CL-AAGKA protocol enjoys some other outstanding performance. The infrastructure needed to support our CL-AAGKA is lightweight when compared to a GKA protocol which is based on traditional PKI. This is because the requirement of managing certificates is completely eliminated in our CL-AAGKA protocol. This useful property makes our protocol attractive for bandwidth-constrained, low-power scenarios, such as mobile security applications.

Although our CL-AAGKA protocol is no longer identity-based, it does enjoy the property that the generation of private key can be executed after the generation and use of its corresponding public key. This is a useful feature that allows our CL-AAGKA protocol to enjoy some properties of attribute-based cryptography [18]. In particular, the identity ID in our CL-AAGKA could contain the attributes that the intended user should hold.

5. Conclusion

Group key agreement protocols are important tools for secure communications over open networks. In this paper, we have proposed a CL-AAGKA scheme to circumvent the defects of the existing AAGKA protocols. It captures most desirable security properties of key agreement protocols, but does not suffer from complicated certificate management burden in PKI cryptographic mechanism or private key escrow problem in ID-based cryptographic mechanism. We have also realized the protocol by using the PBC library. And the simulation results throw out that the proposed protocol is efficient and practical.

References

[1] L. Zhang, Q. Wu, B. Qin, and J. Domingo-Ferrer, “Provably secure one-round identity-based authenticated asymmetric group key agreement protocol,” *Information Sciences*, vol.181, no.19, pp.4318-4329, Oct.2011. Article (CrossRef Link).

[2] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644-654, November, 1976. Article (CrossRef Link).

[3] A. Joux, “A one round protocol for tripartite diffie-hellman,” *Journal of Cryptology*, vol.17, no.4, pp.263-276, Sep. 2004. Article (CrossRef Link).

[4] M. Burmester and Y. Desmedt, “A secure and efficient conference key distribution system,” in *Proc. of EUROCRYPT’94 on the Theory and Application of Cryptographic Techniques*, pp.275-286, May.1994. Article (CrossRef Link).

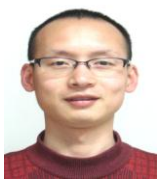
[5] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, “Asymmetric group key agreement,” in: *Proc. of EUROCRYPT’09 on the Theory and Applications of Cryptographic Techniques*, pp.153-170, Apr.2009. Article (CrossRef Link).

[6] L. Zhang, Q. Wu, B. Qin, J. Domingo-Ferrer, and úrsula González-Nicolás, “Asymmetric group key agreement protocol for open networks and its application to broadcast encryption,” *Computer Networks*, vol.55, no.15, pp.3246-3255, Oct.2011.Article (CrossRef Link).

[7] L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer, "A scalable robust authentication protocol for secure vehicular communications," *IEEE Transactions on Vehicular Technology*, vol.59, no.4, pp.1606-1617, May.2010. Article (CrossRef Link).
[8] R. Dutta and R. Barua, "Constant round dynamic group key agreement," in: *Proc. of ISC 2005 on Information Security*, pp.74-88, Sep.2005. Article (CrossRef Link).
[9] H. J. Kim, S. M. Lee, and D. H. Lee, "Constant-round authenticated group key exchange for dynamic groups," in: *Proc. of Asiacrypt'04 on the Theory and Application of Cryptology and Information Security*, pp.245-259, Dec.2004. Article (CrossRef Link).
[10] A. Shamir, "Identity based cryptosystems and signature schemes," in: *Proc. of CRYPTO'84 on Advances in Cryptology*, vol.196, pp.47-53, 1984. Article (CrossRef Link).
[11] S.S. Al-Riyami and K.G. Paterson, "Certificateless public key cryptography," in: *Proc. of the ASIACRYPT'03 on the Theory and Application of Cryptology and Information Security*, pp.452-473, Nov.2003. Article (CrossRef Link).
[12] J. Liu, M. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," in: *ACM ASIACCS'07*, pp.273-283, 2007. Article (CrossRef Link).
[13] W. Yap, S. Heng, and B. Goi, "An efficient certificateless signature scheme," in: *Proc. of EUC Workshops on Emerging Directions in Embedded and Ubiquitous Computing*, pp.322-331, August 1-4, 2006. Article (CrossRef Link).
[14] Pairing-Based Cryptography Library. Article (CrossRef Link).
[15] M. Girault, "Self-certified public keys," in: *Proc. Of EUROCRYPT'91 on the Theory and Application of Cryptographic Techniques*, pp.490-497, Apr.1991. Article (CrossRef Link).
[16] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in: *Proc. of 6th IMA Int. Conference on Cryptography and Coding*, pp.30-45, December 17-19, 1997. Article (CrossRef Link).
[17] C. Mitchell, M. Ward, and P. Wilson, "Key control in key agreement protocols," *Electronic Letters*, vol.34, no.10, pp.980-981, May.1998. Article (CrossRef Link).
[18] John Bethencourt, Amit Sahai and Brent Waters, "Ciphertext-Policy Attribute-Based Encryption," in *Proc. of IEEE Symposium on Security and Privacy*, pp.321~334, May. 2007. Article (CrossRef Link).

Guiyi Wei obtained his Ph.D. in Dec 2006 from Zhejiang University. He has research interests in wireless networks, mobile computing, cloud computing, social networks and network security. Guiyi Wei is a full professor and vice dean of the School of Computer Science and Information Engineering at Zhejiang Gongshang University. He is also the director of the Networking and Distributed Computing Laboratory.

Xianbo Yang obtained his B.S. degree in electronic information engineering from China Jiliang University in 2008. He is currently working toward the M.S. degree in the Colleage of Computer and Information Engineering at Zhejiang Gongshang University. His major research interests are cryptography, computer and network security, mobile computing and cloud computing.



Jun Shao obtained his Ph.D. in June 2008 from Shanghai Jiao Tong University. From May 2008 to April 2010, Dr. Shao was a post-doctoral researcher in S2 Lab in College of Information Sciences and Technology at Pennsylvania State University with Prof. Peng Liu. Dr. Shao has research interests in cryptography, computer and network security. Dr. Shao is now an associate professor of Colleage of Computer and Information Engineering at Zhejiang Gongshang University.