Secure Mobile Content Delivery Using Dynamic Group Key Agreement with Batch Verification*

Seokhyang Cho[1], Kiwon Song[1], Dongsub Cho[1], and Dongho Won[2]

[1] Department of Computer Science and Engineering, Ewha Womans University, 11-1 Daehyun-dong, Seodaemun-gu, Seoul 120-750, Republic of Korea
shcho@security.re.kr, keeweesong@hanmail.net, dscho@ewha.ac.kr
[2] Information Security Group, Sungkyunkwan University, 300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, Republic of Korea
dhwon@security.re.kr

Abstract. Recently, the bilinear pairings such as the Weil and the Tate pairings defined on algebraic curves over a finite field have found applications in the design of cryptographic protocols. One useful application in mobile environments is for secure group communication over a public network. The members in the group need to establish a common group key that will be used to encrypt messages to be broadcast to the group. Furthermore, it is important to update the group key with low computational costs when the members join and leave the group. In this paper, we propose a pairing-based key exchange protocol for dynamic groups. The proposed protocol achieves low communication complexity and provides some computational savings by the batch verification of signatures. We show that the security of our scheme is guaranteed against an active adversary in the random oracle model under the bilinear Diffie-Hellman (BDH) assumption.

Keywords: Group key agreement, bilinear map, batch verification, BDH assumption.

1 Introduction

The basic requirement for secure group communications through insecure public channels is that all group members must agree on a common secret key. This shared secret key, called the *session key*, can later be used to encrypt messages to be broadcast to the group. Group key agreement protocols are designed to meet this requirement, with the fundamental security goal being to establish the session key in such a way that no one except the group members can know the value of the session key.

In key agreement protocols, more than one party contribute information to generate the common session key. In this paper we focus on *contributory* key agreement protocols in which the session key is derived as a function of contributions provided by all parties [1]. Therefore in our contributory key agreement

* This work was supported by the 2nd phase of Brain Korea (BK) 21 Project funded by the Korea Research Foundation.

O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4706, Part II, pp. 996–1007, 2007.

protocols, a correctly behaving party is assured that as long as his contribution is chosen at random, even a coalition of all other parties will not be able to have any means of controlling the final value of the session key.

The mobile computing architecture we visualize is asymmetric in the sense of computational capabilities of participants. That is, the protocol participants consist of a stationary server (also called *application server* or *service provider*) with sufficient computational power and a cluster of mobile devices (also called *clients*) with limited computational resources. An unbalanced mobile environment is common in a number of applications such as Internet stock quotes, audio and music delivery, and so on [17].

Unfortunately, signature verifications based on pairings are ten times or one hundred times slower than that of RSA or DSA [9]. This problem may be critical in some applications such as electronic commerce or banking services in which one server has to verify many signatures simultaneously [25]. So, in order to enhance the efficiency of verification process, we adopt a variant of the signature scheme by F. Hess [16] consisting of multiple signatures generated by a single signer.

In this paper, we propose a pairing-based key exchange protocol. The proposed protocol is suited for dynamic groups in which group members may join and leave the current group at any given time. Our protocol achieves low communication complexity and provides large computational savings by the batch verification of signatures. Moreover, our protocol also achieves forward secrecy and is provably secure against an active adversary in the random oracle model under the bilinear Diffie-Hellman assumption.

Related Work. Ever since 2-party Diffie-Hellman key exchange was first proposed in 1976, a number of works [1,4,5,7,10,18,22,23,24] have attempted to solve the fundamental problem of securely distributing a session key among a group of n parties. But unfortunately, all of them suffer from one or more of the drawbacks as $O(n)$ or $O(\log n)$ rounds of communication, $O(n)$ broadcasts per round, and lack of forward secrecy. In fact, most published protocols require $O(n)$ communication rounds to establish a session key, and hence become prohibitively expensive as the group size grows. Other protocols [10,24], while they require only a constant number of rounds to complete key agreement, do not achieve forward secrecy.

In [7], Burmester and Desmedt (BD) presented a two-round protocol which provides forward secrecy with no proof of security in the original paper. Recently K. Y. Choi *et al.* [13] transformed the BD protocol into a pairing-based version(B-GKA) and then proposed an ID-based authenticated group key scheme(AGKA) with security proof. But their two-round B-GKA protocol turns out to be vulnerable to an impersonation attack by F. Zhang [26]. In 2005, they also proposed an efficient ID-based AGKA protocol which achieves only *half forward secrecy* in the sense that disclosure of client's long-term secret keys does not compromise the security of previously established session, while disclosure of server's long-term secret keys does compromise the security[14].

Katz and Yung [22] proposed a three-round protocol which provides a rigorous security proof against an active adversary in the standard model. However, an

obvious drawback of this protocol is that communication overhead is significant with three rounds of n broadcasts. This means that each user in this protocol, in each of three rounds, must receive $n-1$ messages from the rest of the group before he/she can proceed to the next step. It is obvious that this kind of extreme connectivity inevitably delays the whole process of the protocol.

The initial work [6] proposed by Bresson *et al.* deals with the static case, and shows a protocol which is secure under the DDH assumption. Later works [4,5,3] focus on the dynamic group key agreement to support membership changes that users join or leave and the session key must be updated whenever it occurs. More recently, Bresson and Catalano proposed a constant round key exchange protocol, based on secret sharing techniques that combines with ElGamal cryptosystem as underlying encryption primitive [2]. However, with increasing number of users, the complexity of the protocol goes beyond the capabilities of limited-function devices such as PDAs and handheld computers.

2 The Proposed Scheme

Let $(\mathbb{G}, +)$ and $(\mathbb{V}, \cdot)$ denote cyclic groups of prime order q, $P \in \mathbb{G}$ a generator of $\mathbb{G}$ and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{V}$ be a bilinear mapping which satisfies the following properties.

1. **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q^*$. It is known that this can be restated in the following way. For any $P, Q, R \in \mathbb{G}$, $e(P + Q, R) = e(P, R)e(Q, R)$ and $e(P, Q + R) = e(P, Q)e(P, R)$.
2. **Non-degeneracy:** If P is a generator of $\mathbb{G}$, then $e(P, P)$ is a generator of $\mathbb{V}$. In other words, $e(P, P) \neq 1$.

We also assume that $e(P, Q)$ can easily be computed while, for any given random $Q \in \mathbb{G}$ and $v \in \mathbb{V}$, it should be infeasible to compute $P \in \mathbb{G}$ such that $e(P, Q) = v$. We define three hash functions

$$h : \{0,1\}^* \times \mathbb{V} \to \mathbb{Z}_q^*, \quad H : \{0,1\}^* \to \mathbb{G}^*, \quad \mathcal{H} : \{0,1\}^* \to \{0,1\}^\ell$$

where $\mathbb{G}^* := \mathbb{G} \setminus \{0\}$ and ℓ is the length of the session key to be distributed in the protocols. We also abbreviate $\mathbb{V}^* := \mathbb{V} \setminus \{1\}$.

By the notation U_n, we denote a special user called *server* whose role will become apparent in the description of the protocol. In the setup phase, any trusted authority (or TA) chooses $\mathbb{G}, \mathbb{V}$ and P as defined above. The public parameters $e, \mathbb{G}, \mathbb{V}, P, h, H$ and $\mathcal{H}$ are assumed to be known a priori to all parties. We also assume that each user knows the authentic public keys of all other users.

We now present a dynamic key agreement scheme consisting of three protocols P_{gka}, P_{leave}, and P_{join} for initial group key establishment, user leave, and user join, respectively. First, the protocol P_{gka} proceeds as follows:

2.1 Group Key Agreement: Protocol P_{gka}

Let $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$ be a set of n users who wish to generate a session key by participating in our group key agreement protocol P_{gka}.

Setup. The TA picks a random integer $t \in \mathbb{Z}_q^*$, computes $Q_{TA} = tP$ and publishes Q_{TA} while t is a secret.

Extract. This algorithm is performed by the TA when a user U_i requests the secret key corresponding to his identity which is given as the string ID_i. The TA then computes the secret key of ID_i as $S_{ID_i} = tH(ID_i)$ and returns it to the user U_i.

Key agreement. The protocol P_{gka} runs in two rounds, once with $n-1$ unicasts and once with a single broadcast, as follows:

– **Round 1.** Each U_i $(i \in [1, n-1])$ selects random integers $k_i, r_i \in \mathbb{Z}_q^*$, *precomputes* $s_i = e(S_{ID_i}, P)^{k_i}$, $P_i = r_i P$ and $Q_i = (h(s_i) + k_i)S_{ID_i}$. Then each client $U_i \neq U_n$ sends a message $m_i = (s_i, P_i, Q_i)$ to the server U_n. The server U_n also selects a random $r_n \in \mathbb{Z}_q^*$ and then *precomputes* $P_n = r_n P$.

– **Round 2.** After having received all the $n-1$ messages from user U_i $(i \in [1, n-1])$, U_n can do batch verification by checking the correctness of the following equation:

$$\prod_{i=1}^{n-1} s_i = \prod_{i=1}^{n-1} e(Q_i, P)e(H(ID_i), -Q_{TA})^{h(s_i)}$$

The server U_n chooses a random $r \in \mathbb{Z}_q^*$ and computes $P_r = rP$. Then the server generates a nonce $\delta \in \{0,1\}^\ell$ and computes X that

$$X = \bigoplus_{i \in [1,n]} \mathcal{H}(\delta \parallel x_i)$$

where ℓ is a security parameter and $x_i = e(P_i, rQ_{TA})$. U_n also computes $Y = \{X_i \mid X_i = X \oplus \mathcal{H}(\delta \parallel x_i), i \in [1, n-1]\}$ and then generates the signature σ_n of message $\delta \parallel P_r \parallel Y \parallel \mathcal{U}$. Now U_n broadcasts the message $m_n = (\delta, P_r, Y, \mathcal{U}, \sigma_n)$ to the entire client group members.

Key computation. Having received the broadcast message from U_n, each $U_i \neq U_n$ first verifies the correctness of the server's signature, and then computes

$$X = X_i \oplus \mathcal{H}(\delta \parallel x_i)$$

where $x_i = e(P_r, Q_{TA})^{r_i}$. Lastly, $U_i (\in \mathcal{U})$ computes its session key K as $K = \mathcal{H}(X, Y)$.

2.2 User Leave: Protocol P_{leave}

Assume a scenario where a set of users $\mathcal{L}$ leaves the group $\mathcal{U}$ except for the server U_n. Then protocol P_{leave} is executed to provide each user of the new group $\mathcal{U}' = \mathcal{U} \setminus \mathcal{L}$ with a new session key. Protocol P_{leave} requires only one communication round with a single broadcast and it proceeds as follows:

Round 1. The server U_n generates a new nonce $\delta_1 \in \{0,1\}^\ell$ and computes

$$X' = \bigoplus_{U_i \in \mathcal{U}'} \mathcal{H}(\delta_1 \parallel x_i).$$

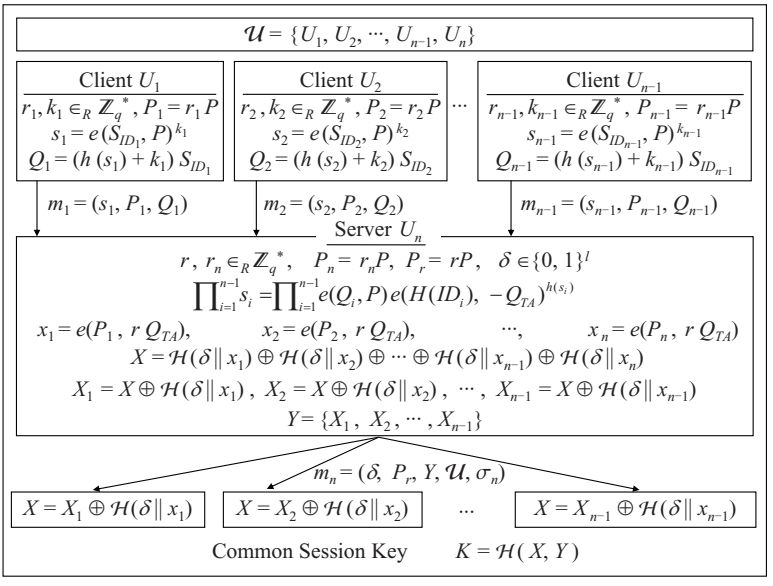

Fig. 1. Initial Key Agreement Protocol (P_{gka})

And U_n also computes $Y' = \{X_i' \mid X_i' = X' \oplus \mathcal{H}(\delta_1 \parallel x_i), U_i \in \mathcal{U}' \backslash \{U_n\}\}$. Then U_n generates the signature σ_n' of message $\delta_1 \parallel P_r \parallel Y' \parallel \mathcal{U}'$. Now U_n broadcasts the message $m_n' = (\delta_1, P_r, Y', \mathcal{U}', \sigma_n')$ to the entire client group members.

Key computation. Upon receiving the broadcast message m_n', each U_i computes

$$X' = X_i' \oplus \mathcal{H}(\delta_1 \parallel x_i)$$

where $x_i = e(P_r, Q_{TA})^{r_i}$. Lastly, each user $U_i (\in \mathcal{U}')$ computes its session key K as $K = \mathcal{H}(X', Y')$.

2.3 User Join: Protocol P_{join}

Assume a scenario in which a set of j new users, $\mathcal{J}$, joins the current group $\mathcal{U}$ to form a new group $\mathcal{U}'' = \mathcal{U} \cup \mathcal{J}$. Then the join protocol P_{join} is run to provide the users of $\mathcal{U}''$ with a session key. P_{join} takes two communication rounds, once with j unicasts and once with a single broadcast, and it proceeds as follows:

Round 1. Each $U_i \in \mathcal{J}$ selects random $k_i, r_i \in \mathbb{Z}_q^*$ and *precomputes* $s_i = e(S_{ID_i}, P)^{k_i}$, $P_i = r_i P$, and $Q_i = (h(s_i) + k_i) S_{ID_i}$. $U_i \in \mathcal{J}$ then sends a message $m_i = (U_i, s_i, P_i, Q_i)$ to the server, and stores its random k_i and r_i.

Round 2. Having received all the messages from the new user $U_i \in \mathcal{J}$, U_n verifies the following equation:

$$\prod_{U_i \in \mathcal{J}} s_i = \prod_{U_i \in \mathcal{J}} e(Q_i, P) e(H(ID_i), -Q_{TA})^{h(s_i)}$$

U_n proceeds in the usual way, generating a new random nonce $\delta_2 \in \{0,1\}^\ell$, computing X'', Y'', and $K = \mathcal{H}(X'', Y'')$, updating the new x_i's. Then having received all the j messages from the new users, U_n computes X'' as

$$X'' = \bigoplus_{U_i \in \mathcal{U}''} \mathcal{H}(\delta_2 \parallel x_i)$$

where $x_i = e(P_i, rQ_{TA})$. U_n also computes $Y'' = \{X_i'' \mid X_i'' = X'' \oplus H(\delta_2 \parallel x_i), U_i \in \mathcal{U}'' \backslash \{U_n\}\}$, and then generates the signature σ_n'' of message $\delta_2 \parallel P_r \parallel Y'' \parallel \mathcal{U}''$. Now U_n broadcasts the message $m_n'' = (\delta_2, P_r, Y'', \mathcal{U}'', \sigma_n'')$ to the entire client group members.

Key computation. Having received the broadcast message from U_n, each $U_i (\neq U_n)$ first verifies the correctness of the server's signature, and then computes

$$X'' = X_i'' \oplus \mathcal{H}(\delta_2 \parallel x_i)$$

where $x_i = e(P_r, Q_{TA})^{r_i}$ and its session key K as $K = \mathcal{H}(X'', Y'')$.

3 Efficiency

To analyze the communication complexity and computation cost, we now discuss the efficiency of the protocol introduced in the preceding section.

Communication Complexity. It is easy to see that our P_{gka} protocol runs only in two rounds of communication, requiring $n-1$ *unicasts* in the first round and a single broadcast in the second one. Hence the total number of messages required by our P_{gka} protocol is n, which is optimal as shown in [11].

In contrast, the two-round protocol presented by Burmester and Desmedt (BD) [7] requires n broadcasts in each of two rounds, and therefore requires, in total, $2n$ *broadcast messages* to complete key agreement (as already mentioned, the protocol presented by Katz and Yung [22], in its basic form, is essentially the same as the BD protocol). More seriously, without the ability of broadcasting communication, this protocol requires $O(n^2)$ messages to be sent or received, which makes it inefficient for many applications.

And in the same manner, the protocol B-GKA, which is a bilinear variant of the BD protocol and the ID-GKA (ID-based authenticated group key agreement protocol) based on the B-GKA, which is by K. Y. Choi *et al.* [13], have the same message complexity as the original protocol [7].

Computational Complexity. Each U_i in our P_{gka} protocol computes 2 pairing operations, 2 scalar multiplications on $\mathbb{G}$, and 2 exponentiations in $\mathbb{V}$, except U_n who generates one signature, verifies the validity of $n-1$ transcripts from U_i simultaneously, and performs $O(n)$ exponentiations, $O(n)$ pairing operations, and 3 scalar multiplications. If pre-computations are possible, most of the computations in the first round can be performed off-line and thus, only 1 exponentiation and 1 pairing operation per client is required to be done on-line. On the other hand, each user U_i in the K. Y. Choi *et al*'s ID-GKA protocol

Table 1. Complexity Comparison

	Complexity	B-GKA [13]	ID-GKA [13]	Our protocol
Communication	Rounds	2	2	2
	Unicast	0	0	$O(n)$
	Broadcast	$O(n)$	$O(n)$	$O(1)$
Computation	Multiplication	$O(n \log n)$	$O(n \log n)$	0
	Exponentiation	1	1	2 ($O(n)$ for U_n)
	Scalar multiplication	3	4	2 (3 for U_n)
	Pairing operation	2	4	2 ($O(n)$ for U_n)

computes $O(n \log n)$ multiplications, 1 exponentiation, 4 pairing operations and 4 scalar multiplications. And in the B-GKA protocol, each user U_i needs fewer computations as many as 1 scalar multiplication and 2 pairing operations than the B-GKA protocol. Furthermore all users in our P_{leave} protocol do not need such a large amount of computations since they use the stored values and only the newly members in our P_{join} protocol need the same amount of computations as that required in P_{gka}.

In the table 1, we have compared the complexity of our protocol with those of the K. Y. Choi *et al*'s two protocols: the B-GKA protocol and the ID-GKA protocol. As for computational costs, the table lists the amount of computation performed per user. As seen from the table, our protocol is better than the K. Y. Choi *et al*'s ID-GKA protocol in terms of computational complexity. However, in situations where users with equal computational capabilities communicate over a broadcast network, the fully-symmetric protocol of Burmester and Desmedt (or ID-GKA) might be more favorable than our protocol which, in contrast, is well suited for more realistic settings where users with asymmetric computational resources are spread across a wide area network.

4 Security Proof

We now claim that the group key agreement protocol proposed in this paper is secure against active adversaries provided that the bilinear Diffie-Hellman (BDH) problem is computationally hard. The scheme in [8] is shown to be secure if an elliptic curve variant of the computational Diffie-Hellman problem is infeasible.

Definition 1 (Bilinear Diffie-Hellman (BDH) Problem). Let $\mathbb{G}$ be a cyclic group $\langle P \rangle$ of prime order q and a, b, c are chosen at random in $\mathbb{Z}_q^*$. A (T, ϵ)-BDH-attacker in $(\mathbb{G}, \mathbb{V}, e)$ is a probabilistic algorithm running in time T that given (P, aP, bP, cP), outputs $e(P, P)^{abc}$ with probability of at least ϵ. The advantage of any probabilistic, polynomial time algorithm $\mathcal{A}$ for solving the BDH problem in $(\mathbb{G}, \mathbb{V}, e)$ is defined to be :

$$\mathsf{Adv}^{\mathrm{BDH}}_{\mathcal{A},(\mathbb{G},\mathbb{V},e)} = |\Pr[e(P,P)^{abc} \leftarrow \mathcal{A}(\mathbb{G},\mathbb{V},e,P,aP,bP,cP) \mid P \in \mathbb{G}^*; a,b,c \in_R \mathbb{Z}_q^*]|$$

The BDH problem is (T,ϵ)-**intractable** if there is no (T,ϵ)-attacker in $(\mathbb{G},\mathbb{V},e)$.

Definition 2 (Authenticated Group Key Agreement). The security of an authenticated group key agreement scheme $\mathcal{P}$ is defined in the following context. The adversary executes a protocol P_{gka}, P_{leave}, or P_{join} as many times as he/she wishes in an arbitrary order with P_{gka} being the first one executed. During executions of the protocols, the adversary $\mathcal{A}$, at any time, asks Test query to a fresh user, gets back an ℓ-bit string as the response to this query, and at some later point in time, outputs a bit b' as a guess for the secret bit b. Let CG (Correct Guess) be the event that the adversary $\mathcal{A}$ correctly guesses the bit b, i.e., the event that $b' = b$. Then we define the advantage of $\mathcal{A}$ in attacking $\mathcal{P}$ as

$$\mathsf{Adv}_{\mathcal{A},\mathcal{P}}(k) = 2 \cdot \Pr[\mathsf{CG}] - 1$$

We say that a group key agreement scheme $\mathcal{P}$ is secure if $\mathsf{Adv}_{\mathcal{A},\mathcal{P}}(k)$ is negligible for any probabilistic polynomial time adversary $\mathcal{A}$.

Theorem 1. *Let* $\mathsf{Adv}_{\mathcal{P}}(t,q_{ex},q_{se})$ *be the maximum advantage in attacking* $\mathcal{P}$*, where the maximum is taken over all adversaries that run in time* t*, and make* q_h *random oracle queries,* q_{ex} $\mathsf{Execute}$ *queries, and* q_{se} Send *queries. Then we have*

$$\mathsf{Adv}_{\mathcal{P}}(t,q_{ex},q_{se}) \leq 2nq_hq_{ex} \cdot \mathsf{Adv}^{\mathrm{BDH}}_{\mathbb{G},\mathbb{V},e}(t) + n\mathsf{Adv}^{\mathsf{Forge}}_{\Gamma}(t),$$

where $\mathsf{Adv}^{\mathsf{Forge}}_{\Gamma}(t)$ *is the maximum advantage of any forger* $\mathcal{F}$ *running in time* t*.*

Proof. First the adversary $\mathcal{A}$ is assumed to gain its advantage by forging authentication transcripts. We can use $\mathcal{A}$ to construct a forger $\mathcal{F}$ that generates a valid message pair $\langle ID, rP, s, Q\rangle$ with respect to an authentication scheme Γ as follows: a forger $\mathcal{F}$ honestly generates all other public/private keys by running the $\mathsf{Extract}$ algorithm. $\mathcal{F}$ then simulates the oracle queries of $\mathcal{A}$ in the natural way. This results in a perfect simulation unless $\mathcal{A}$ makes the query $\mathsf{Corrupt}(ID)$. If this occurs, $\mathcal{F}$ halts and outputs "`fail`". Otherwise, if $\mathcal{A}$ outputs $\langle ID, rP, s, Q\rangle$ as a valid forgery, then $\mathcal{F}$ generates the message pair $\langle ID, rP, s, Q\rangle$. The success probability of $\mathcal{F}$ satisfies $\Pr_{\mathcal{A}}[\mathsf{Forge}] \leq n\mathsf{Adv}^{\mathsf{Forge}}_{\Gamma}(t)$.

Assume that an adversary $\mathcal{A}$ can guess the hidden bit b correctly with probability $1/2 + \epsilon$. Then we construct from $\mathcal{A}$ an algorithm that solves the BDH problem in $(\mathbb{G},\mathbb{V},e)$ with probability ϵ/q_hq_{ex}. Let us first define the following two distributions:

$$\mathsf{Real} = \left\{ (T,K) \;\middle|\; \begin{array}{l} k_1,k_2,\cdots,k_{n-1},r_1,r_2,\cdots,r_n,r \in_R \mathbb{Z}_q^*; \qquad \delta \in \{0,1\}^l; \\ P_1 = r_1P, P_2 = r_2P, \cdots, P_n = r_nP, P_r = rP; \\ s_1 = e(S_{ID_1},P)^{k_1}, \cdots, s_{n-1} = e(S_{ID_{n-1}},P)^{k_{n-1}}; \\ Q_1 = (h(s_1) + k_1)S_{ID_1}, \cdots, \\ Q_{n-1} = (h(s_{n-1}) + k_{n-1})S_{ID_{n-1}}; \\ x_1 = e(P,P)^{rtr_1}, \cdots, x_n = e(P,P)^{rtr_n}; \\ h_1 = \mathcal{H}(\delta \parallel x_1), h_2 = \mathcal{H}(\delta \parallel x_2), \cdots, h_n = \mathcal{H}(\delta \parallel x_n); \\ X = \oplus_{i=1}^n h_i; \qquad y_i = X \oplus h_i, i \in [1,n-1] \end{array} \right\},$$

$$\mathsf{Fake} = \left\{ (T,K) \;\middle|\; \begin{array}{l} k_1, k_2, \cdots, k_{n-1}, r_1, r_2, \cdots, r_n, r \in_R \mathbb{Z}_q^*; \\ \delta, w_1, w_2, \cdots, w_n \in \{0,1\}^l; \\ P_1 = r_1 P, P_2 = r_2 P, \cdots, P_n = r_n P, P_r = rP; \\ s_1 = e(S_{ID_1}, P)^{k_1}, \cdots, s_{n-1} = e(S_{ID_{n-1}}, P)^{k_{n-1}}; \\ Q_1 = (h(s_1) + k_1) S_{ID_1}, \cdots, \\ Q_{n-1} = (h(s_{n-1}) + k_{n-1}) S_{ID_{n-1}}; \\ h_1 = w_1, h_2 = w_2, \cdots, h_n = w_n; \\ X = \oplus_{i=1}^n h_i; \qquad y_i = X \oplus h_i, i \in [1, n-1] \end{array} \right\},$$

where $T = (s_1, \cdots, s_{n-1}, P_r, P_1, \cdots, P_{n-1}, Q_1, \cdots, Q_{n-1}, \delta, y_1, \cdots, y_{n-1})$ and $K = \mathcal{H}(y_1, \cdots, y_n, X)$.

Lemma 1. *Let $\mathcal{A}'$ be an algorithm that, given (T, K) coming from one of the two distributions* Real *and* Fake*, runs in time t and outputs 0 or 1. Then we have:*

$$\begin{aligned} |\Pr[\mathcal{A}'(T,K) = 1 | (T,K) \longleftarrow \mathsf{Real}\,] - \Pr[\mathcal{A}'(T,K) = 1 | (T,K) \longleftarrow \mathsf{Fake}\,]| \\ \leqq q_h \mathsf{Adv}^{\mathrm{BDH}}_{\mathbb{G},\mathbb{V},e}(t + (2n-4)t_{smul} + t_{pair} + (3n-7)t_{exp}) \end{aligned}$$

where t_{smul}, t_{exp}, and t_{pair} are the time required to compute a scalar multiplication on $\mathbb{G}$, an exponentiation in $\mathbb{V}$ and a pairing operation respectively.

Proof. Assume that an algorithm $\mathcal{A}$ can distinguish between the two distributions with a non-negligible probability. Then, since $\mathcal{H}$ is a random oracle and a difference between the Real and the Fake is in the method of computing $h_i (i \in [1, n])$, we must find out at least one value of x_i to distinguish between them. Now, given an input tuple $(P, rP, tP, r_2 P)$ in $\mathbb{G}^4$ we construct an algorithm that outputs a value $e(P,P)^{r' t r_2}$ as follows.

We first choose random integers $r_1, r_3, \alpha_i, \beta_i, \gamma_i (i \in [4, n])$ in $\mathbb{Z}_q^*$ and define a random r_i as $r_1\alpha_i + r_2\beta_i + r_3\gamma_i (i \in [4, n]) \bmod q$. We then can compute x_i and $X = \oplus_{i=1}^n h_i$ with a random $h_i \in \{0,1\}^l$, and finally construct $y_i = X \oplus h_i$. Consider the following distribution

$$\mathsf{Simul} = \left\{ (T,K) \;\middle|\; \begin{array}{l} k_1, \cdots, k_{n-1}, r_1, r_3, \alpha_4, \beta_4, \gamma_4, \cdots, \alpha_n, \beta_n, \gamma_n, x_i' \in_R \mathbb{Z}_q^*; \\ \delta, h_1, h_2, \cdots, h_n \in \{0,1\}^l; \\ r_4 = r_1\alpha_4 + r_2\beta_4 + r_3\gamma_4, \cdots, r_n = r_1\alpha_n + r_2\beta_n + r_3\gamma_n; \\ P_1 = r_1 P, P_2 = r_2 P, \cdots, P_n = r_n P, P_r = rP; \\ s_1 = e(S_{ID_1}, P)^{k_1}, \cdots, s_{n-1} = e(S_{ID_{n-1}}, P)^{k_{n-1}}; \\ Q_1 = (h(s_1) + k_1) S_{ID_1}, \cdots, Q_{n-1} = (h(s_{n-1}) + k_{n-1}) S_{ID_{n-1}}; \\ x_1 = e(P,P)^{rtr_1}, x_2 = e(P,P)^{r' t r_2}, x_3 = e(P,P)^{rtr_3}, \\ \qquad x_4 = e(P,P)^{t(rr_1\alpha_4 + r' r_2 \beta_4 + rr_3\gamma_4)}, \cdots, \\ \qquad x_n = e(P,P)^{t(rr_1\alpha_n + r' r_2 \beta_n + rr_3\gamma_n)}; \\ X = \oplus_{i=1}^n h_i; \qquad y_i = X \oplus h_i, i \in [1, n-1] \end{array} \right\},$$

where T and K are as defined above. If $(P, rP, tP, r_2 P, e(P,P)^{r' t r_2})$ is a BDH tuple (i.e., $r = r'$), we have $\mathsf{Simul} \equiv \mathsf{Real}$ since $x_i = e(P,P)^{rtr_i}$ for all $i \in [1, n]$.

Otherwise, i.e., if $(P, rP, tP, r_2P, e(P,P)^{r'tr_2})$ is a random tuple, it is clear that $\mathsf{Simul} \equiv \mathsf{Fake}$.

Given the transcript (T, K) from the distribution Simul as an input of $\mathcal{A}'$, we simulate a random oracle $\mathcal{H}$ at the same time. When $\mathcal{A}'$ finishes finally the execution, we select a random $\delta \parallel x_i'$ that inputs in the random oracle simulation table. If $x_i' = x_i$, we can solve BDH problem $x_i = e(P,P)^{t(rr_1\alpha_i + rr_2\beta_i + rr_3\gamma_i)}$. Therefore the algorithm $\mathcal{A}'$ having the transcript (T, K) provided by the simulation can not distinguish between two distributions. □

Lemma 2. *For any (computationally unbounded) adversary $\mathcal{A}$, we have :*

$$\Pr[\mathcal{A}'(T, K_b) = b | (T, K_1) \longleftarrow \mathsf{Fake}\ ; K_0 \longleftarrow \{0,1\}^l\ ; b \longleftarrow \{0,1\}] = 1/2.$$

Proof. In the experiment Fake, we represent from the transcript T the value y_i by the following $n-1$ equations and can write the solution $(h_1, h_2, \cdots, h_n)$ as follows

$$\begin{aligned}
y_1 &= h_2 \oplus h_3 \oplus \cdots \oplus h_n = h_1 \oplus h_n \oplus y_n, & h_1 &= y_1 \oplus y_n \oplus h_n \\
y_2 &= h_1 \oplus h_3 \oplus \cdots \oplus h_n = h_2 \oplus h_n \oplus y_n, & h_2 &= y_2 \oplus y_n \oplus h_n \\
&\vdots \\
y_{n-1} &= h_1 \oplus h_2 \oplus \cdots \oplus h_{n-2} \oplus h_n = h_{n-1} \oplus h_n \oplus y_n, & h_{n-1} &= y_{n-1} \oplus y_n \oplus h_n \\
& & h_n &.
\end{aligned}$$

Therefore the adversary does not obtain any information about the value X from any one of transcripts since there are a lot of solutions, amounting to 2^l solutions for an independent variable h_n. This implies that

$$\Pr[\mathcal{A}'(T, X_b) = b | (T, X_1) \longleftarrow \mathsf{Fake}\ ; X_0 \longleftarrow \{0,1\}^l\ ; b \longleftarrow \{0,1\}] = 1/2.$$

Since $\mathcal{H}$ is a random oracle, the statement of Lemma 2 immediately follows. □

Armed with the two lemmas above, we now give the details of the algorithm $\mathcal{B}$ from construction of the distribution Simul. Assume that an adversary $\mathcal{A}$ makes its Test query to an oracle activated by the $\delta^{th}\mathsf{Execute}$ query. The algorithm $\mathcal{B}$ begins by choosing a random $d \in \{1, 2, \cdots, q_{ex}\}$ as a guess for the value of δ. $\mathcal{B}$ then invokes $\mathcal{A}$ and simulates the queries of $\mathcal{A}$. $\mathcal{B}$ answers all the queries from $\mathcal{A}$ in the obvious way, following the protocol exactly as specified, except for the case where a query is the $d^{th}\mathsf{Execute}$ query. In this latter case, the algorithm $\mathcal{B}$ generates (T, K) depending on the distribution Simul and answers the $d^{th}\mathsf{Execute}$ query of $\mathcal{A}$ with T.

The algorithm $\mathcal{B}$ outputs a random element in $\mathbb{V}$ if $d \neq \delta$. Otherwise, the algorithm answers the Test query of $\mathcal{A}$ with K. At some later point, when $\mathcal{A}$ terminates and outputs its guess b'. Applying the Lemma 1 and 2 together with the fact that $\Pr[b = b'] = 1/2$ and $\Pr[d = \delta] = 1/q_{ex}$, we obtain

$$\begin{aligned}
&\Pr[\mathcal{A}(T, K_b) = b \,|(T, K_1) \longleftarrow \mathsf{Real}\,;\ K_0 \longleftarrow \{0,1\}^l;\ b \longleftarrow \{0,1\}] = 1/2 + \epsilon, \\
&\mathsf{Adv}^{\mathrm{BDH}}_{\mathbb{G},\mathbb{V},e}(\mathcal{B}) = \epsilon/(q_h q_{ex}),
\end{aligned}$$

which immediately yields the statement of Theorem 1. □

5 Conclusion

In this paper we have proposed a pairing-based group key agreement scheme with optimal message complexity; the protocol runs only in two rounds, once with $n-1$ unicasts and once with a single broadcast. Therefore, due to its low communication cost and reduced computational complexity resulting from the batch verification, the protocol is well suited for dynamic groups. Furthermore, the protocol provides forward secrecy and has been proven to be secure against an active adversary under the bilinear Diffie-Hellman assumption. However, for practical purposes, more realistic and suitable applications need to be found. Also we leave some experimental results to demonstrate the efficiency of our scheme for further research.

References

1. Ateniese, G., Steiner, M., Tsudik, G.: New multiparty authentication services and key agreement protocols. IEEE Journal on Selected Areas in Communications 18(4), 628–639 (2000)
2. Bresson, E., Catalano, D.: Constant round authenticated group key agreement via distributed computation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 115–129. Springer, Heidelberg (2004)
3. Bresson, E., Chevassut, O., Essiari, A., Pointcheval, D.: Mutual authentication and group key agreement for low-power mobile devices. In: Proc. of MWCN'03, pp. 59–62 (2003)
4. Bresson, E., Chevassut, O., Pointcheval, D.: Provably authenticated group Diffie-Hellman key exchange — the dynamic case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)
5. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic group Diffie-Hellman key exchange under standard assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
6. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group Diffie-Hellman key exchange. In: Proc. of CCS'01, pp. 255–264 (2001)
7. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
8. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Barreto, P., Kim, H., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)
10. Boyd, C., Nieto, J.M.G.: Round-optimal contributory conference key agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2002)
11. Becker, K., Wille, U.: Communication complexity of group key distribution. In: Proc. of CCS'98, pp. 1–6 (1998)
12. Choo, K.R., Boyd, C., Hitchcock, Y.: Errors in computational complexity proofs for protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)

13. Choi, K.Y., Hwang, J.Y., Lee, D.H.: Efficient ID-based group key agreement with bilinear maps. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 130–144. Springer, Heidelberg (2004)
14. Choi, K.Y., Hwang, J.Y., Lee, D.H., Seo, I.S.: ID-based authenticated key agreement for low-power mobile devices. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 494–505. Springer, Heidelberg (2005)
15. Dutta, R., Barua, R.: Constant round dynamic group key agreement. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 74–88. Springer, Heidelberg (2005)
16. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
17. Huang, Y., Garcia-Molina, H.: Publish/subscribe in a mobile environment. In: Proc. of MobiDE'01, pp. 27–34 (2001)
18. Ingemarsson, I., Tang, D., Wong, C.: A conference key distribution system. IEEE Trans. on Information Theory 28(5), 714–720 (1982)
19. Just, M., Vaudenay, S.: Authenticated multi-party key agreement. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 36–49. Springer, Heidelberg (1996)
20. Kim, H.J., Lee, S.M., Lee, D.H.: Constant-round authenticated group key exchange for dynamic groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245–259. Springer, Heidelberg (2004)
21. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proc. of CCS'05, pp. 180–189 (2005)
22. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
23. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Trans. on Parallel and Distributed Systems 11(8), 769–780 (2000)
24. Tzeng, W.-G., Tzeng, Z.-J.: Round-efficient conference key agreement protocols with provable security. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 614–627. Springer, Heidelberg (2000)
25. Yoon, H.J., Cheon, J.H., Kim, Y.: Batch verifications with ID-based signatures. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 233–248. Springer, Heidelberg (2006)
26. Zhang, F., Chen, X.: Attack on an ID-based authenticated group key agreement scheme from PKC 2004. In: Information Processing Letters archive, vol. 91(4), pp. 191–193. Elsevier Science Inc., Amsterdam (2004)