Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 865367, 18 pages
http://dx.doi.org/10.1155/2014/865367

Research Article

Interdomain Identity-Based Key Agreement Schemes

Chun-I Fan,[1] Yi-Hui Lin,[2] Tuan-Hung Hsu,[1] and Ruei-Hau Hsu[3]

[1] *Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan*
[2] *Institute of Information Science, Academia Sinica, Taipei 115, Taiwan*
[3] *College of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan*

Correspondence should be addressed to Chun-I Fan; cifan@faculty.nsysu.edu.tw

Received 15 April 2014; Accepted 11 August 2014; Published 30 November 2014

Academic Editor: Tadeusz Kaczorek

In order to simplify key management, two-party and three-party key agreement schemes based on user identities have been proposed recently. Multiparty (including more than three parties) key agreement protocols, which also are called conference key schemes, can be applied to distributed systems and wireless environments, such as ad hoc networks, for the purpose of multiparty secure communication. However, it is hard to extend two- or three-party schemes to multiparty ones with the guarantee of efficiency and security. In addition to the above two properties, interdomain environments should also be considered in key agreement systems due to diversified network domains. However, only few identity-based multiparty conference key agreement schemes for single domain environments and none for interdomain environments were proposed in the literature and they did not satisfy all of the security attributes such as forward secrecy and withstanding impersonation. In this paper, we will propose a novel efficient single domain identity-based multiparty conference key scheme and extend it to an interdomain one. Finally, we prove that the proposed schemes satisfy the required security attributes via formal methods.

1. Introduction

The technique of key agreement allows two or more parties to exchange information and negotiate a common session key. The first key exchange scheme was proposed by Diffie and Hellman in 1976 [1] where two parties can exchange public information and then compute a common key by their private keys and received information. However, the basic Diffie-Hellman protocol lacks mutual authentication between two parties such that the man-in-the-middle attack is valid in this scheme. Many researchers modified Diffie-Hellman protocol to ensure mutual authentication between two parties, which are called authenticated key agreement (AKA) protocols. Lots of varieties of Diffie-Hellman protocol have been proposed and several different kinds of key agreement mechanisms have been shown in [2]. Up to now, Diffie-Hellman key exchange protocol is still an important basis for most key agreement protocols.

In 1984, Shamir proposed an identity-based cryptosystem [3], where the public key of each user is her/his public identity information, and there exists a private key generator (PKG), a key generation center (KGC), or a Trusted Authority (TA) which is trusted by all users. PKG, KGC, or TA, which will be called TA below, can produce each user's private key according to her/his public key. In almost all of the identity-based key agreement schemes, TA provides the private/public key generation services for users. When a user registers with TA, the user's public information like ID or email address will be her/his public key and TA gives the user the private key corresponding to her/his public key.

Pairing is a tool which is initially applied to cryptography to convert the Discrete Logarithm problem in elliptic curves to that in finite fields, and it can be derived from bilinear pairing, namely, Weil pairing [4] or Tate pairing [5]. First, Joux [6] used pairing to construct the first 3-party key agreement protocol based on a certificate system in 2000 and his scheme. Later, researchers found that pairing is suitable for the implementation of identity-based cryptosystems. Smart [7] proposed a two-party identity-based authenticated key agreement scheme in 2002. Boneh and Franklin [4] proposed an identity-based encryption scheme based on Weil pairing in 2003. Afterwards, pairing has become an important mathematic foundation of cryptography. There are many identity-based key agreement schemes, which have been proposed in the literature [7–11], based on pairings.

A conference key agreement scheme is a variety of a multiparty key agreement or group key agreement scheme, but it is different from conference key distribution scheme. In a conference key distribution scheme, a session conference chair decides the conference key and then broadcasts it to every member in this session conference. In particular, in a conference key agreement scheme, we must guarantee that the protocol satisfies the following three properties.

(1) Each conference key is negotiated by all session members.
(2) Every session member can compute the conference key via the same algorithm.
(3) No session member can predict or preselect the conference key.

The first formal security analysis in an identity-based two-party key agreement scheme was introduced by Chen and Kudla [9] and they improved the first identity-based key agreement scheme based on pairings [7]. Chen and Kudla proved that their protocol is secure on the security model of Bellare and Rogaway [12]. Later, Al-Riyami and Paterson also proposed four kinds of tripartite authenticated key agreement protocols by improving Joux's scheme [13], and they showed that their scheme is secure. Unfortunately, Shim and Woo [14] pointed out that their scheme has some weaknesses. Furthermore, there are several conference key agreement schemes based on bilinear pairing which have been proposed in the literature [15–19], but they are all insecure, where their security weaknesses will be shown in Section 3 of the paper.

Section 4 will present two new hard problems, the n-Linear Diffie-Hellman (n-LDH) problem and the Decisional n-Linear Diffie-Hellman (n-DLDH) problem, on which our key agreement schemes are based.

In Section 5, we will propose a novel efficient identity-based conference key agreement scheme by combining the concepts of [16, 19]. In addition to a single TA, we also discuss how the users, who have registered with distinct TAs, negotiate a common conference key. Moreover, in order to formally demonstrate the security of our proposed schemes, we adopt the random oracle method, which was proposed by Bellare and Rogaway [12], to prove the security of our schemes under some well-known assumptions. We will define several security attributes in the third part of Section 2 and formally prove the security of our schemes in Section 6. Finally, we also provide performance comparison to demonstrate that our proposed schemes are more efficient than others.

Our contributions are summarized as follows.

(1) We find some security flaws in the schemes of [15–19].
(2) We introduce two new hard problems.
(3) We propose interdomain identity-based conference key agreement schemes.
(4) We formally prove that our schemes completely satisfy all of the security attributes.

2. Preliminaries

In this section, we review the concept of pairing which includes definitions, computationally hard problems, and security attributes of key agreement based on pairings.

2.1. Pairing

2.1. Pairing. Pairing [20] in an elliptic curve cryptosystem is a function which maps a pair of elliptic curve points to an element of a multiplicative group in a finite field. It has been applied to key agreement, signatures, broadcast encryption, and identity-based encryption widely. In the following, we will review the definitions and properties of pairings.

2.1.1. Bilinear Pairing. We briefly describe the concept of bilinear pairing [20]. Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, +)$ be abelian groups written in additive notation with prime order q and identity elements O_1 and O_2, respectively, such that $qP = O_1$ and $qQ = O_2$, where $\forall P \in \mathbb{G}_1$ and $\forall Q \in \mathbb{G}_2$. Suppose that $(\mathbb{G}_T, *)$ is a cyclic group of order q written in multiplicative notation with identity element 1_T. Now we have the groups $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, and $(\mathbb{G}_T, *)$. The mapping function is

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T. \quad (1)$$

Typically, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of the points on an elliptic curve over a finite field and $\mathbb{G}_T$ is a subgroup of a multiplicative group over a finite field.

In addition, the following additional properties must be satisfied:

(i) bilinearity

$\forall P, P' \in \mathbb{G}_1, Q, Q' \in \mathbb{G}_2$,
$e(P+P', Q) = e(P,Q) \cdot e(P', Q)$ and $e(P, Q+Q') = e(P,Q) \cdot e(P,Q')$,
$e(aP, Q) = e(P,Q)^a = e(P, aQ)$ for all $a \in \mathbb{Z}_q^*$;

(ii) nondegeneracy

$\forall P \in \mathbb{G}_1$, with $P \neq O_1$, $\exists Q \in \mathbb{G}_2$ such that $e(P,Q) \neq 1_T$,
$\forall Q \in \mathbb{G}_2$, with $Q \neq O_2$, $\exists P \in \mathbb{G}_1$ such that $e(P,Q) \neq 1_T$,
$\forall P \in \mathbb{G}_1$ and $\forall Q \in \mathbb{G}_2$, $e(P, O_2) = e(O_1, Q) = 1_T$;

(iii) computability

if $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, there exists an efficient algorithm which can compute $e(P,Q)$ in polynomial time.

The schemes in Section 3 use symmetric bilinear pairing, so they set $\mathbb{G}_1 = \mathbb{G}_2$. In order to make the following decisional problems remain hard, we set $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no polynomial-time computable isomorphism $\phi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$, such that $\phi(P) = Q$, where P is a generator of $\mathbb{G}_1$ and Q is a generator of $\mathbb{G}_2$.

2.2. Hard Problems

(1) *The Discrete Logarithm* (DL) problem:
given $P, P' \in \mathbb{G}_1$, find an integer $x \in \mathbb{Z}_q^*$ such that $P' = xP$.

(2) *The Computational Diffie-Hellman* (CDH) problem: for $x, y \in_R \mathbb{Z}_q^*$, given $(P, xP, yP) \in \mathbb{G}_1^3$, compute xyP.

(3) *The Decisional Diffie-Hellman* (DDH) problem:
for $x, y \in_R \mathbb{Z}_q^*$, given $(P, xP, yP, zP) \in \mathbb{G}_1^4$ where $z = xy \pmod q$ or $z \in_R \mathbb{Z}_q^*$ is decided by flipping a coin. Output "Yes" if $z = xy \pmod q$; otherwise output "No".

(4) *The Divisible Computational Diffie-Hellman* (DCDH) problem [21]: for $x, y \in_R \mathbb{Z}_q^*$, given $(P, xP, yP) \in \mathbb{G}_1^3$, compute $xy^{-1}P$.

(5) *The Decisional Linear Diffie-Hellman* (DLDH) problem in $\mathbb{G}_1$ [22, 23]: for $x_1, x_2, x_3, x_4 \in_R \mathbb{Z}_q^*$, given $(P, x_1P, x_2P, x_1x_3P, x_2x_4P, Z) \in \mathbb{G}_1^6$, where $Z = (x_3 + x_4)P$ or $Z \in_R \mathbb{G}_1$ is decided by flipping a coin. Output "Yes" if $Z = (x_3 + x_4)P$; otherwise output "No".
This hard problem was first proposed by Boneh et al. [22] in 2004 and then Boyen and Waters [23] extended it to asymmetric bilinear groups in 2006.

(6) *The co-Bilinear Diffe-Hellman* (co-BDH) problem [4]: given $(P_1, aP_1, bP_1) \in \mathbb{G}_1^3$ and $(P_2, aP_2, cP_2) \in \mathbb{G}_2^3$ in asymmetric bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, compute $e(P_1, P_2)^{abc} \in \mathbb{G}_T$.
We propose the variant-CDH problem and extend the DLDH problem to the n-LDH and n-DLDH problems. We will prove that they are also hard in Section 4.

(7) *The Variant Computational Diffie-Hellman* (variant-CDH) problem: given $(P_1, aP_1, bP_1) \in \mathbb{G}_1^3$ and $(P_2, aP_2) \in \mathbb{G}_2^2$, compute abP_1.

(8) *The n-Linear Diffie-Hellman* (n-LDH) problem: given P, x_iP's, and $x_ix_{n+j}P$'s $\in \mathbb{G}_1$ for all i, j with $1 \le i, j \le n$, $i \ne j$, and $n \ge 2$, compute $(x_{n+1} + x_{n+2} + \cdots + x_{2n})P$.

(9) *The Decisional n-Linear Diffie-Hellman* (n-DLDH) problem: given Z, P, x_iP's, and $x_ix_{n+j}P$'s $\in \mathbb{G}_1$ for all i, j with $1 \le i, j \le n, i \ne j$, and $n \ge 2$, where $Z = (x_{n+1} + x_{n+2} + \cdots + x_{2n})P$ or $Z \in_R \mathbb{G}_1$ is decided by flipping a coin. Output "Yes" if $Z = (x_{n+1} + x_{n+2} + \cdots + x_{2n})P$; otherwise output "No".

2.3. Security Attributes. There are some security definitions in the identity-based key agreement schemes based on pairing [13, 14]. We describe them as follows.

Known Session Key Security. A key agreement protocol should produce a unique common secret key, which is called a session key, for every session. The protocol should still achieve this goal when an adversary has learned all of the other session keys.

(Perfect) Forward Secrecy. Forward secrecy is that any adversary cannot derive previous session keys from compromised long-term private keys of one or more parties. *Partial forward secrecy* is that one or more (not all) parties' long-term private keys are corrupted but any adversary cannot get any previous session keys which were established by these parties. *Perfect forward secrecy* means that any adversaries cannot derive previous session keys even though they have obtained the long-term private keys of all parties. In ID-based systems, perfect forward secrecy implies that TA's and all users' long-term private keys are corrupted but any previous session key established by the registered users cannot be derived by adversaries. We also call it *TA forward secrecy*.

Key-Compromise Impersonation. A protocol can resist key-compromise impersonation if an adversary cannot impersonate some users even though the other users' long-term private keys were disclosed.

Man-in-the-middle attack is a special case of key-compromise impersonation in ID-based systems. If an adversary intercepts messages, retransmits them, and then communicates with users without being detected in the key agreement protocol, we say that he succeeds in impersonation.

Withstanding key-compromise impersonation also covers unknown key-share resilience. It is the basic security attribute for key agreement scheme. Some users cannot have a key agreement with the other users without the knowledge of them. If some users cannot impersonate the others, they cannot run the key agreement scheme for them.

Key Control. It should be impossible for any participant (or an adversary) to preselect a value as a session key or predict the value of the session key.

3. Security Problems in the Previous Schemes

In the section, we briefly introduce security weaknesses on the schemes [15–19, 24–26]. The details of the security problems in these schemes are in the Appendices.

Shi et al. [19] proposed an ID-based authenticated group key agreement protocol in 2005. The design of the protocol is efficient because it only takes one round to finish a group key agreement and it needs no exponentiation computation besides a pairing computation. We find that the protocol does not resist key-compromise impersonation since the users do not verify the messages with one another in the protocol. Moreover, it only achieves partial forward secrecy.

Du et al. [15] proposed an ID-based authenticated group key agreement protocol in 2003 and improved it in the same year. Both of them does not achieve perfect forward secrecy. Although they embed a signature scheme to verify the messages, both of the protocols still suffer from key-compromise impersonation found by Zhang and Chen [29]. In the attack of [29], the adversaries collect the messages of the user in the previous session and replay them after modifying the messages. Zhang and Chen [30] also attacked Choi et al. [24] with the same method in 2004. The protocol of Zhang et al. [18] in 2005 has the same security problem as Du's since they embed the same signature scheme in the protocol.

Kim et al. [17] aims to design a one-round key agreement protocol. But we find that the protocol cannot even achieve known session key security. Anyone can compute the session key through collecting the broadcasting messages.

Zhou et al. [26] proposed two schemes, one is one-round and the other is two-round. We find that both of them cannot withstand key-compromise impersonation. For the first scheme, the other users can collide to impersonate the user U_i. For the second one, the user U_1 can impersonate any other user he wants. We also find that the protocol of Yao et al. [25] is not immune to key-compromise impersonation, either. A user can impersonate another by rebroadcasting the messages. The work of [31] improved the flaw but did not provide any formal proofs. Yuan et al. [27] improved it with formal proofs.

4. Three New Hard Problems

We formally prove our proposed problems, the Variant Computational Diffie-Hellman problem, the n-Linear Diffie-Hellman problem, and the Decisional n-Linear Diffie-Hellman problem, being hard by using problem reduction and generic model, respectively.

4.1. The Variant Computational Diffie-Hellman (Variant-CDH) Problem

Theorem 1. *The variant-CDH problem is hard if the co-BDH problem is hard.*

Proof. Suppose that there exists an oracle which can solve the variant-CDH problem with nonnegligible probability. We will prove that the oracle can help us to solve the co-BDH problem with nonnegligible probability. Given any parameters of the co-BDH problem, $(P_1, aP_1, bP_1) \in \mathbb{G}_1^3$ and $(P_2, aP_2, cP_2) \in \mathbb{G}_2^3$, we input (P_1, aP_1, bP_1) and (P_2, aP_2) into the variant-CDH oracle. The oracle will output abP_1. Then, we solve the co-BDH problem by computing $e(abP_1, cP_2) = e(P_1, P_2)^{abc}$. □

4.2. The n-Linear Diffie-Hellman (n-LDH) Problem

Theorem 2. *The n-LDH problem is hard if and only if the DCDH problem is hard.*

Proof. (1) n-LDH ⇒ DCDH. Suppose that there exists an oracle which can solve the n-LDH problem with nonnegligible probability. We will prove that the oracle can help us to solve the DCDH problem with nonnegligible probability.

For any DCDH triple (P, xP, yP), we convert them into the n-LDH oracle's input parameters which are shown in (2):

$$\begin{bmatrix} x_1P & x_2P & \dots & x_{n-1}P & x_nP \\ \perp & x_2x_{n+1}P & \dots & x_{n-1}x_{n+1}P & x_nx_{n+1}P \\ x_1x_{n+2}P & \perp & \dots & x_{n-1}x_{n+2}P & x_nx_{n+2}P \\ \vdots & \vdots & & \vdots & \vdots \\ x_1x_{2n-1}P & x_2x_{2n-1}P & \dots & \perp & x_nx_{2n-1}P \\ x_1x_{2n}P & x_2x_{2n}P & \dots & x_{n-1}x_{2n}P & \perp \end{bmatrix}. \quad (2)$$

We randomly pick $t_2, t_3, \dots, t_n \in \mathbb{Z}_q^*$ and $Q_1, Q_2, \dots, Q_{n-1} \in \mathbb{G}_1^*$, compute $Q_n = xP - Q_1 - Q_2 - \cdots - Q_{n-1}$, and set other parameters in (3):

$$\begin{bmatrix} yP & t_2yP & t_3yP & \dots & t_{n-1}yP & t_nyP \\ \perp & t_2Q_1 & t_3Q_1 & \dots & t_{n-1}Q_1 & t_nQ_1 \\ Q_2 & \perp & t_3Q_2 & \dots & t_{n-1}Q_2 & t_nQ_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ Q_{n-1} & t_2Q_{n-1} & t_3Q_{n-1} & \dots & \perp & t_nQ_{n-1} \\ Q_n & t_2Q_n & t_3Q_n & \dots & t_{n-1}Q_n & \perp \end{bmatrix}. \quad (3)$$

Equation (2) is equal to (3); that is, $x_1P = yP$, $x_2P = t_2x_1P = t_2yP$, $x_3P = t_3x_1P = t_3yP, \dots, x_nP = t_nx_1P = t_nyP$ in row 1, $x_2x_{n+1}P = t_2x_1x_{n+1}P = t_2Q_1$, $x_3x_{n+1}P = t_3x_1x_{n+1}P = t_3Q_1, \dots, x_nx_{n+1}P = t_nx_1x_{n+1}P = t_nQ_1$ in row 2 (suppose that $x_1x_{n+1}P = Q_1$), $x_1x_{n+2}P = Q_2$, $x_3x_{n+2}P = t_3x_1x_{n+2}P = t_3Q_2, \dots, x_nx_{n+2}P = t_nx_1x_{n+2}P = t_nQ_2$ in row 3,…, and $x_1x_{2n}P = Q_n$, $x_2x_{2n}P = t_2x_1x_{2n}P = t_2Q_n$, $x_3x_{2n}P = t_3x_1x_{2n}P = t_3Q_n, \dots, x_{n-1}x_{2n}P = t_{n-1}x_1x_{2n}P = t_{n-1}Q_n$ in row $n+1$. The oracle will output $(x_{n+1} + x_{n+2} + x_{n+3} + \cdots + x_{2n-1} + x_{2n})P$. Thus, we have that

$$\begin{aligned} xP &= Q_1 + Q_2 + \cdots + Q_{n-1} + Q_n \\ &= x_1x_{n+1}P + x_1x_{n+2}P + \cdots + x_1x_{2n-1}P + x_1x_{2n}P \\ &= (x_{n+1} + x_{n+2} + \cdots + x_{2n-1} + x_{2n})\, x_1P \\ &= (x_{n+1} + x_{n+2} + \cdots + x_{2n-1} + x_{2n})\, yP. \end{aligned} \quad (4)$$

From (4), we can get $xy^{-1}P = (x_{n+1} + x_{n+2} + \cdots + x_{2n-1} + x_{2n})P$.

(2) n-LDH ⇐ DCDH. Suppose that there exists an oracle which can solve the the DCDH problem with nonnegligible probability. We will prove that the oracle can help us to solve the n-LDH problem with nonnegligible probability, too.

For any n-LDH tuple in (2), we input $(x_1x_{n+2}P, x_1P)$, $(x_1x_{n+3}P, x_1P), \dots, (x_1x_{2n}P, x_1P)$, and $(x_2x_{n+1}P, x_2P)$ into the oracle. Then the oracle outputs $x_{n+2}P, x_{n+3}P, \dots, x_{2n}P$, and $x_{n+1}P$, respectively.

Finally, we can compute $x_{n+1}P + x_{n+2}P + \cdots + x_{2n}P = (x_{n+1} + x_{n+2} + \cdots + x_{2n})P$ to solve the n-LDH problem. □

We use a way similar to [22] to prove the n-DLDH problem being hard. In the generic model, elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are encoded as unique random strings, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups and $\mathbb{G}_T$ is a multiplicative group. There is a bilinear pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let S_1, S_2, and S_T be the sets of strings. The opaque encoding of the elements of $\mathbb{G}_1$ is modeled as an injective function

$\zeta_1 : \mathbb{Z}_q \rightarrow S_1$, where $S_1 \subset \{0,1\}^*$, which maps all $a \in \mathbb{Z}_q$ to the string representation $\zeta_1(a)$ of $aP \in \mathbb{G}_1$. Analogous mapping $\zeta_2 : \mathbb{Z}_q \rightarrow S_2$ and $\zeta_T : \mathbb{Z}_q \rightarrow S_T$ map all $a \in \mathbb{Z}_q$ to the string representation $\zeta_2(a)$ of $aP' \in \mathbb{G}_2$ and $\zeta_T(a)$ of $g^a \in \mathbb{G}_T$, where $g = e(P, P')$.

4.3. The Decisional n-Linear Diffie-Hellman (n-DLDH) Problem

Theorem 3. *Let $\mathscr{A}$ be an algorithm that solves the n-DLDH problem in the generic bilinear group model with at most q_k oracle queries. Let x_i's, x_{n+j}'s, and $z \in \mathbb{Z}_q$ be chosen at random, where $1 \leq i, j \leq n$, and $n \geq 2$, ζ_1, ζ_2, and ζ_T are random encoding functions for G_1, G_2, and G_T, and $b \in \{0,1\}$ is a random bit. Let $w_b = z$ and $w_{1-b} = \sum_{j=1}^{n} x_{n+j} \bmod q$. The probability is*

$$\Pr\left[\begin{array}{c}\mathscr{A}\left(\zeta_1,\zeta_2,\zeta_T;1,x_i\text{'s},x_ix_{n+j}\text{'s},w_0,w_1\right)=b:\\ \forall i,j,x_i,x_{n+j}\in\mathbb{Z}_q^*,\\ 1\le i,j\le n, i\ne j, n\ge 2, b\in\{0,1\}\end{array}\right] = \frac{1}{2} + \frac{2\left(n^2+3+q_k\right)^2}{q}. \tag{5}$$

Proof. $\mathscr{B}$ plays the following game with $\mathscr{A}$. $\mathscr{B}$ maintains the lists $L_1 = \{(F_{1,s}, \zeta_{1,s}) : s = 0,1,2,\dots,\tau_1 - 1\}$, $L_2 = \{(F_{2,t}, \zeta_{2,t}) : t = 0,1,2,\dots,\tau_2 - 1\}$, and $L_T = \{(F_{T,u}, \zeta_{T,u}) : t = 0,1,2,\dots,\tau_T - 1\}$. Let X_i's, X_iX_{n+j}'s, W_0, and W_1 ($\forall i,j, 1 \leq i, j \leq n, i \neq j$) be indeterminate. All $F_{1,s}$'s, $F_{2,t}$'s, and $F_{T,u}$'s $\in \mathbb{Z}_q[X_i\text{'s}, X_iX_{n+j}\text{'s}, W_0, W_1]$ are polynomials and $\zeta_{1,s}$'s, $\zeta_{2,t}$'s, $\zeta_{T,u}$'s $\in \{0,1\}^*$ are distinct strings. At the beginning of the game, $\mathscr{B}$ sets $F_{1,0} = 1$, $F_{1,1} = X_1$, $F_{1,2} = X_2, \dots, F_{1,n} = X_n$, $F_{1,n^2+1} = W_0$, $F_{1,n^2+2} = W_1$, $F_{2,0} = 1$, $F_{T,0} = 1$, and the following polynomials: $[(\perp, F_{1,n+1} = X_2X_{n+1}, \dots, F_{1,2n-2} = X_{n-1}X_{n+1}, F_{1,2n-1} = X_nX_{n+1})$, $(F_{1,2n} = X_1X_{n+2}, \perp, \dots, F_{1,3n-3} = X_{n-1}X_{n+2}, F_{1,3n-2} = X_nX_{n+2})$, $\dots$, $(F_{1,n^2-2n+3} = X_1X_{2n-1}, F_{1,n^2-2n+4} = X_2X_{2n-1}, \dots, \perp, F_{1,n^2-n+1} = X_nX_{2n-1})$, $(F_{1,n^2-n+2} = X_1X_{2n}, F_{1,n^2-n+3} = X_2X_{2n}, \dots, F_{1,n^2} = X_{n-1}X_{2n}, \perp)]$, where the symbol "$\perp$" means emptiness and $\mathscr{B}$ gives $\mathscr{A}$ the distinct strings $\zeta_{1,0}, \zeta_{1,1}, \dots, \zeta_{1,n^2+2}$, $\zeta_{2,0}$, and $\zeta_{T,0}$. In the initial list index, the numbers of the records in L_1, L_2, and L_T are $\tau_1 = n^2 + 3$, $\tau_2 = 1$, and $\tau_T = 1$, respectively, where $\tau = \tau_1 + \tau_2 + \tau_T = n^2 + 5$. At any step in the game, $\mathscr{A}$ can make the group and pairing queries. $\mathscr{B}$ performs and responds to $\mathscr{A}$ as follows.

Group Action. $\mathscr{A}$ gives $\mathscr{B}$ two operands $\zeta_{1,s}$, $\zeta_{1,t}$ and a sign bit, where $0 \leq s, t < \tau_1$. $\mathscr{B}$ sets $F_{1,\tau_1} \leftarrow F_{1,s} \pm F_{1,t}$. If $F_{1,\tau_1} = F_{1,l}$ for some $l < \tau_1$, $\mathscr{B}$ sets $\zeta_{1,\tau_1} \leftarrow \zeta_{1,l}$. Otherwise, $\mathscr{B}$ sets ζ_{1,τ_1} to be a string in $\{0,1\}^*$ distinct from $\zeta_{1,0}, \zeta_{1,1}, \dots, \zeta_{1,\tau_1-1}$. Finally, $\mathscr{B}$ adds $(F_{1,\tau_1}, \zeta_{1,\tau})$ to the list L_1, gives ζ_{1,τ_1} to $\mathscr{A}$, and sets $\tau_1 \leftarrow \tau_1 + 1$. The group action queries in G_2 and G_T are simulated similarly.

Pairing. $\mathscr{A}$ gives $\mathscr{B}$ two operands $\zeta_{1,s}$ and $\zeta_{1,t}$ with $0 \leq s < \tau_1$ and $0 \leq t < \tau_2$. $\mathscr{B}$ sets the product $F_{T,\tau_T} \leftarrow F_{1,s}F_{2,t}$. If $F_{T,\tau_T} = F_{T,l}$ for some $l < \tau_T$, $\mathscr{B}$ sets $\zeta_{T,\tau_T} \leftarrow \zeta_{T,l}$. Otherwise, $\mathscr{B}$ sets ζ_{T,τ_T} to be a string in $\{0,1\}^*$ distinct from $\zeta_{T,0}, \zeta_{T,1}, \dots, \zeta_{T,\tau_T-1}$. Finally, $\mathscr{B}$ adds $(F_{T,\tau_T}, \zeta_{T,\tau_T})$ to the list L_T, gives ζ_{T,τ_T} to $\mathscr{A}$, and sets $\tau_T \leftarrow \tau_T + 1$. □

Consider the operation that $\mathscr{B}$ performs: (1) $\mathscr{B}$ adds/subtracts all polynomials in the list L_1, L_2, and L_T by any $\mathscr{A}$'s query. (2) $\mathscr{B}$ produces any of two polynomials in L_1 and L_2 to generate a new polynomial in L_T. For any variant X_{n+j}, it occurs within the monomials $X_1X_{n+j}, X_2X_{n+j}, \dots, X_{n+j-1}X_{n+j}, X_{n+j+1}X_{n+j}, \dots, X_{2n}X_{n+j}$ in L_1 and L_T lists and it occurs no monomial in L_2 list. Therefore, $\mathscr{B}$ cannot produce any polynomial that contains the monomial cX_{n+j}'s in G_1 or G_2 and the monomial $F_{2,t}X_{n+j}$'s in G_T for any coefficient $c \neq 0$ and any nonzero monomial $F_{2,t}$ in L_2 in the available operations.

After at most q queries, $\mathscr{A}$ terminates and returns a guess $\hat{b} \in \{0,1\}$. The distinct values of operands provide no information to $\mathscr{A}$ because they are random bit strings. Therefore, the probability that $\mathscr{A}$ wins the game in the generic model is 1/2.

However, when $\mathscr{B}$ randomly chooses x_i's, x_{n+j}'s, and $z \in \mathbb{Z}$, sets $w_b = \sum_{j=1}^{n} x_{n+j}$ and $w_{1-b} = z$, and assigns $X_1 \leftarrow x_1, X_2 \leftarrow x_2, \dots, X_{2n} \leftarrow x_{2n}, W_b \leftarrow w_b$, and $W_{1-b} \leftarrow w_{1-b}$, a nontrivial equality relation may occur and give $\mathscr{A}$ some information that is not revealed in the generic model; that is, for some $F_{1,s} \neq F_{1,t}$ (and $F_{T,s} \neq F_{T,t}$, resp.) and $s, t < \tau_1$ (and $s, t < \tau_T$, resp.), $F_{1,s}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = F_{1,t}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1)$ (and $F_{T,s}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = F_{T,t}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1)$, resp.).

The probability of the occurrence is computed according to the following lemma.

Lemma 4 (see [32]). *Let q be prime and let $t \geq 1$. Let $F(X_1, X_2, \dots, X_k) \in Z/q^t[X_1, X_2, \dots, X_k]$ be a nonzero polynomial of total degree d. Then for random $x_1, x_2, \dots, x_k \in Z/q^t$, the probability that $F(x_1, x_2, \dots, x_k) = 0$ is at most d/q.*

By Lemma 4, all polynomials in the L_1 have degree at most 2, so that, for some s, t and $F_{1,s} \neq F_{1,t}$, the probability of $F(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = F_{1,s}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) - F_{1,t}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = 0$ is at most $2/q$. The degree of polynomials in the L_2 is 0. All polynomials in the L_T have degree at most 2, so that, for some s, t and $F_{T,s} \neq F_{T,t}$, the probability of $F'(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = F_{T,s}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) - F_{T,t}(x_i\text{'s}, x_ix_{n+j}\text{'s}, w_0, w_1) = 0$ is at most $2/q$. Therefore, $\mathscr{A}$ wins the game with the probability $\varepsilon \leq 1/2 + \binom{\tau_1}{2}(2/q) + \binom{\tau_T}{2}(2/q)$. Since $\tau_1 + \tau_T \leq n^2 + 4 + q_k$, we have $\varepsilon \leq 1/2 + (n^2 + 4 + q_k)^2/q$, where the advantage is not greater than $\mathscr{O}(q_k^2/q)$.

5. Our Key Agreement Schemes

In this section, we propose two conference key agreement schemes. The first scheme is designed for the situation where the users who register with a single TA (single domain) want to negotiate a session conference key. Furthermore, the second scheme makes it possible for the users in distinct groups who register with different TAs (interdomain) to negotiate a session conference key. In addition, we will prove the security of the two proposed schemes in Section 6 and compare them with others in Section 7.

5.1. The Proposed Scheme in Single TA

Setup. TA inputs a security parameter κ into a setup algorithm which returns groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ ($\mathbb{G}_1 \neq \mathbb{G}_2$) of prime order q with $q \in \{0,1\}^\kappa$, a suitable bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, generators $P \in \mathbb{G}_1$, $P' \in \mathbb{G}_2$, and three hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_1, H_1 : \mathbb{G}_1 \rightarrow \{0,1\}^l$, $H_2 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \{0,1\}^l$, and $H_3 : \mathbb{G}_1 \rightarrow \{0,1\}^l$ where l is the output length of the hash functions. TA randomly generates a long-term private key $s \in \mathbb{Z}_q^*$ and the public key (sP, sP') and then publishes $\langle q, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, sP, P', sP', H, H_1, H_2, H_3 \rangle$.

Extract. When a user U_i registers a public identity (ID), such as an email address, with TA, TA will check whether the ID belongs to the user. If true, TA issues a long-term private key sQ_i to U_i where $Q_i = H(\text{TA's ID}\|U_i\text{'s ID})$ is U_i's public key.

Conference Key Agreement. Suppose that there are n legal users $U_1, U_2, \ldots,$ and U_n who want to negotiate a conference key. Our conference key agreement scheme contains three rounds described as follows.

Round 1: every user U_i $(1 \leq i \leq n)$ randomly picks an integer $r_i \in \mathbb{Z}_q^*$ as a blinding factor, and then U_i computes $b_i = r_i^{-1}P$ and broadcasts b_i to all users who join this session. The flow is shown in Algorithm 1.

Round 2: after U_i receives all b_j's $(1 \leq j \leq n, j \neq i)$, she/he randomly picks an integer $k_i \in \mathbb{Z}_q^*$ as an ephemeral key and computes $B_i = (k_ib_1, k_ib_2, \ldots, k_ib_{i-1}, \perp, k_ib_{i+1}, \ldots, k_ib_n)$, and then U_i broadcasts B_i. The flow is shown in Algorithm 2.

Round 3: for all B_i's in Round 2, we can rearrange them as shown in (7). When receiving B_j's, U_i only stores k_jb_i $(1 \leq j \leq n, j \neq i)$ and drops other useless information k_jb_t's $(1 \leq j \leq n, 1 \leq t \leq n, t \neq i)$. For example, U_1 stores column 1 and U_2 stores column 2 in (7). Then U_i computes K as follows:

$$\begin{aligned} K &= r_i \sum_{j=1}^{n} k_jb_i \\ &= r_i \cdot \left(k_1r_i^{-1} + k_2r_i^{-1} + \cdots + k_ir_i^{-1} + \cdots + k_nr_i^{-1}\right)P \\ &= (k_1 + k_2 + \cdots + k_i + \cdots + k_n)P. \end{aligned} \tag{6}$$

All B_i's in Round 2

$$\begin{array}{c|cccccc} & U_1 & U_2 & \ldots & U_i & \ldots & U_n \\ \hline B_1 & \perp & k_1r_2^{-1}P & \ldots & k_1r_i^{-1}P & \ldots & k_1r_n^{-1}P \\ B_2 & k_2r_1^{-1}P & \perp & \ldots & k_2r_i^{-1}P & \ldots & k_2r_n^{-1}P \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ B_i & k_ir_1^{-1}P & k_ir_2^{-1}P & \ldots & \perp & \ldots & k_ir_n^{-1}P \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ B_n & k_nr_1^{-1}P & k_nr_2^{-1}P & \ldots & k_nr_i^{-1}P & \ldots & \perp \end{array} \tag{7}$$

U_i computes $\alpha = H_1(K)$, $T_i = \alpha sQ_i + r_i^{-1}sP$, and $\beta_i = H_2(T_i, \alpha)$, and then she/he broadcasts (T_i, β_i). When U_i receives all (T_j, β_j)'s $(1 \leq j \leq n, j \neq i)$, she/he first verifies all β_j's by checking if $H_2(T_j, \alpha) = \beta_j$ for each j. If they are true, U_i randomly chooses $a_1, a_2, \ldots, a_n \in \mathbb{Z}_q^*$, computes $V_i = e(\sum_{i=1}^n a_ib_i + \alpha\sum_{i=1}^n a_iQ_i, sP')$, and verifies whether $V_i = e(\sum_{i=1}^n a_iT_i, P')$ or not. If true, U_i accepts K and computes the session conference key $\text{SCK} = H_3(K)$. Algorithm 3 illustrates the flow in Round 3.

5.2. The Proposed Scheme in Distinct TAs.

Our single domain conference key agreement scheme can be extended to an interdomain conference key agreement scheme. Interdomain means that there are distinct domains with different TAs', respectively. In this subsection, we present our interdomain conference key scheme. Assume that there are m Trusted Authorities $\text{TA}_1, \text{TA}_2, \ldots,$ and TA_m and m user groups $G_1, G_2, \ldots,$ and G_m who register with the m distinct TAs, respectively. In the proposed scheme, the users in m different groups can negotiate a session conference key SCK via the following process.

Setup. TA_i, $i \in \{1, 2, \ldots, m\}$, inputs a security parameter κ into a setup algorithm which returns two groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ ($\mathbb{G}_1 \neq \mathbb{G}_2$) of prime order $q \in \{0,1\}^\kappa$, a suitable bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, generators $P \in \mathbb{G}_1$, $P' \in \mathbb{G}_2$, and three hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_1 : \mathbb{G}_1 \rightarrow \{0,1\}^l$, $H_2 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \{0,1\}^l$, and $H_3 : \mathbb{G}_1 \rightarrow \{0,1\}^l$ where l is the output length of the hash functions. TA_i randomly generates a long-term key $s_i \in \mathbb{Z}_q^*$ and public key (s_iP, s_iP') and then publishes $\langle q, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, s_iP, P', s_iP', H, H_1, H_2, H_3 \rangle$.

Extract. When a user U_j in group i registers a public ID with TA_i, TA_i will check whether the ID belongs to the user. If true, TA_i issues private key $s_iQ_{i,j}$ to the user, where $Q_{i,j} = H(\text{TA}_i\text{'s ID}\|U_{i,j}\text{'s ID})$ is the public key and $U_{i,j}$ denotes user U_j who has registered with TA_i.

Interdomain Conference Key Agreement. Suppose that users in m distinct domains or groups want to negotiate a conference key. Let n_h $(1 \leq h \leq m)$ be the number of users in the hth domain and N be the number of the total users $(N = n_1 + n_2 + \cdots + n_m)$. Our interdomain conference key agreement protocol contains three rounds where Round 1 and Round 2 are similar to those of the proposed single domain conference key protocol.

Round 1: every user $U_{i,j}$ randomly picks an integer $r_{i,j} \in \mathbb{Z}_q^*$ and then broadcasts $b_{i,j} = (r_{i,j})^{-1}P$ to the users who join this session.

Round 2: after receiving $b_{h,l}$'s $(1 \leq h \leq m, 1 \leq l \leq n_h, h \neq i, l \neq j)$, $U_{i,j}$ computes $k_{i,j}b_{h,l}$, where $k_{i,j} \in_R \mathbb{Z}_q^*$ is $U_{i,j}$'s ephemeral key, and broadcasts $B_{i,j} = (k_{i,j}b_{1,1}, k_{i,j}b_{1,2}, \ldots, k_{i,j}b_{1,n_1}, \ldots, k_{i,j}b_{i,1}, \ldots, k_{i,j}b_{i,j-1}, \perp, k_{i,j}b_{i,j+1}, \ldots, k_{i,j}b_{i,n_i}, \ldots, k_{i,j}b_{m,1}, \ldots, k_{i,j}b_{m,n_m})$.

Round 3: when receiving $B_{h,l}$'s, $U_{i,j}$ only stores $k_{h,l}b_{i,j}$'s $(1 \leq h \leq m, 1 \leq l \leq n_h, h \neq i,$ and $l \neq j)$ and drops other useless information $k_{h,l}b_{u,v}$'s $(1 \leq h \leq m, 1 \leq l \leq n_h, 1 \leq u \leq$

$\boxed{\text{Round 1}}$

$\underline{U_i}$ $\qquad\qquad$ $\underline{U_1, U_2, \dots, U_{i-1}, U_{i+1}, \dots, U_n}$

Pick $r_i \in_R \mathbb{Z}_q^*$
Compute $b_i = r_i^{-1} P$

$$\xrightarrow{\text{Broadcast } b_i}$$

$$\xleftarrow{b_1, b_2, \dots, b_{i-1}, b_{i+1}, \dots, b_n}$$

Algorithm 1: Round 1 of the first conference key protocol.

$\boxed{\text{Round 2}}$

$\underline{U_i}$ $\qquad\qquad$ $\underline{U_1, U_2, \dots, U_{i-1}, U_{i+1}, \dots, U_n}$

Pick $k_i \in_R \mathbb{Z}_q^*$
Compute
$k_i b_j = k_i r_j^{-1} P \ (1 \le j \le n, j \ne i)$
$B_i = (k_i b_1, k_i b_2, \dots, k_i b_{i-1}, \perp, k_i b_{i+1}, \dots, k_i b_n)$,

$$\xrightarrow{\text{Broadcast } B_i}$$

$$\xleftarrow{B_1, B_2, \dots, B_{i-1}, B_{i+1}, \dots, B_n}$$

Algorithm 2: Round 2 of the first conference key protocol.

m, $1 \le v \le n_u, u \ne i$, and $v \ne j$). Then $U_{i,j}$ computes every domain's key K_h as follows:

$$\begin{aligned} K_h &= r_{i,j} \sum_{t=1}^{n_h} k_{h,t} b_{i,j} \\ &= r_{i,j} \left(k_{h,1} r_{i,j}^{-1} + k_{h,2} r_{i,j}^{-1} + \cdots + k_{h,n_h} r_{i,j}^{-1} \right) P \\ &= \left(k_{h,1} + k_{h,2} + \cdots + k_{h,n_h} \right) P. \end{aligned} \tag{8}$$

$U_{i,j}$ computes $K = \sum_{h=1}^{m} K_h$, $\alpha = H_1(K)$, $T_{i,j} = \alpha s_i Q_{i,j} + r_{i,j}^{-1} s_i P$, and $\beta_{i,j} = H_2(T_{i,j}, \alpha)$, and then she/he broadcasts $(T_{i,j}, \beta_{i,j})$. When $U_{i,j}$ receives all messages, $U_{i,j}$ first verifies all $\beta_{h,l}$'s by examining if $H_2(T_{h,l}, \alpha) = \beta_{h,l}$ for each h and l ($1 \le h \le m$, $1 \le l \le n_h$, $h \ne i$, and $l \ne j$). If all $\beta_{h,l}$'s are correct, $U_{i,j}$ randomly chooses N integers $a_{h,l}$'s, where each $a_{h,l} \in \mathbb{Z}_q^*$, computes $V_{h,l} = \prod_{h=1}^{m} e(\sum_{l=1}^{n_h} a_{h,l} b_{h,l} + \alpha \sum_{l=1}^{n_h} a_{h,l} Q_{h,l}, s_h P')$, and checks if $V_{h,l} = e(\sum_{h=1}^{m} \sum_{l=1}^{n_h} a_{h,l} T_{h,l}, P')$. If it is true, $U_{i,j}$ accepts K and computes the session conference key SCK $= H_3(K)$.

6. Security Proof

Bellare-Rogaway random oracle model [12, 33], which was extended by Blake-Wilson et al. [34], is suitably modified and adapted in analyzing the security of key agreement protocols like those in the literatures [9, 13]. In this section, we modify Bellare-Rogaway random oracle model and adopt the similar concepts and definitions in [8] to set our security game.

Definition 5 (game environment). Let adversary $\mathscr{A}$ be a probabilistic polynomial time Turing machine and $\mathscr{B}$ a simulator to simulate this game for $\mathscr{A}$. Let $\mathscr{I} = \{U_1, U_2, \dots, U_q\}$ be all users and $\mathscr{U}$ the group users who follow our first identity-based conference key ($\mathscr{IDCK}$) scheme, where q is the order of $\mathbb{G}_1$ and $\mathscr{U} \subseteq \{U_1, U_2, \dots, U_q\}$. In the game, we allow $\mathscr{A}$ to make the following types of queries.

(1) *Execute*$(\Pi_{\mathscr{U}}^s)$: when $\mathscr{A}$ makes the *Execute* query, $\mathscr{B}$ simulates $\mathscr{U}$ to run the first $\mathscr{IDCK}$ protocol Π (Section 5.1) and responds with all public messages (i.e., (b_i, B_i, T_i, β_i)'s for all U_i's $\in \mathscr{U}$) in the sth session.

(2) *Send*$(\Pi_{\widetilde{\mathscr{U}}}^s, m)$: when $\mathscr{A}$ makes the *Send* query with a set of users $\widetilde{\mathscr{U}} \subset \mathscr{U}$ and a message m which is the set of (b_j, B_j, T_j, β_j)'s broadcast by the users in $\widetilde{\mathscr{U}}$, $\mathscr{B}$ simulates all U_i's $\in \mathscr{U} - \widetilde{\mathscr{U}}$ to interact with $\mathscr{A}$ by broadcasting the messages (b_i, B_i, T_i, β_i)'s of U_i's in the sth session.

(3) *Reveal*$(\Pi_{\mathscr{U}}^s)$: $\mathscr{B}$ reveals the session conference key SCK which was held by $\mathscr{U}$ in the sth session.

(4) *Corrupt*(U_i): $\mathscr{B}$ responds with the long-term private key of U_i.

(5) *Test*$(\Pi_{\mathscr{U}}^s)$: when $\mathscr{A}$ makes the *Test* query, $\mathscr{B}$ returns the broadcast messages of the sth session and gives the adversary either the session key of the sth session or a random string. $\mathscr{A}$ then outputs a bit to decide whether the string is the session key or not.

$\boxed{\text{Round 3}}$

$\underline{U_i}$ $\qquad\qquad$ $\underline{U_1, U_2, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n}$

Compute

$K = r_i \sum_{j=1}^{n} k_j b_i = (k_1 + k_2 + \cdots + k_i + \cdots + k_n) P$

$\alpha = H_1(K)$

$T_i = \alpha s Q_i + r_i^{-1} s P$

$\beta_i = H_2(T_i, \alpha)$

$$\xrightarrow{\text{Broadcast } (T_i, \beta_i)}$$

$$\xleftarrow{(T_1, \beta_1),\ (T_2, \beta_2), \ldots,\ (T_{i-1}, \beta_{i-1}),\ (T_{i+1}, \beta_{i+1}), \ldots,\ (T_n, \beta_n)}$$

Verify:

$\beta_1, \beta_2, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_n$

$V_i = e\left(\sum_{i=1}^{n} a_j b_i + \alpha \sum_{i=1}^{n} a_i Q_i, sP'\right) ? = e\left(\sum_{i=1}^{n} a_i T_i, P'\right)$

Compute session conference key SCK $= H_3(K)$

Algorithm 3: Round 3 of the first conference key protocol.

(6) $H(\cdot)$: when a participant inputs a string to H, it responds with the hashed value of the string and the hashed value will be recorded.

(7) $H_1(\cdot)$: when a participant inputs a message $m \in \mathbb{G}_1$ to H_1, it responds with the hashed value of m and the hashed value will be recorded, too.

(8) $H_2(\cdot)$: when a participant inputs (m_1, m_2), where $m_1 \in \mathbb{G}_1$ and $m_2 \in \{0,1\}^*$, to H_2, it responds with the hashed value of (m_1, m_2) and the hashed value will be recorded, too.

6.1. Correctness

Theorem 6 (correctness). *In the presence of a benign adversary $\mathscr{A}$, all the parties always accept holding the same session conference key, which is distributed randomly and uniformly in $\{0,1\}^{\kappa}$, where κ is the security parameter.*

Proof. Every user U_i can generate a valid message (b_i, B_i, T_i, β_i) by following our proposed single domain scheme (Section 5.1), verify the correctness of the message $V_i = e(\sum_{i=1}^{n} a_i b_i + \alpha \sum_{i=1}^{n} a_i Q_i, sP') = e((a_1 r_1^{-1} + a_2 r_2^{-1} + \cdots + a_n r_n^{-1})P + (\alpha a_1 Q_1 + \alpha a_2 Q_2 + \cdots + \alpha a_n Q_n), sP') = e(a_1(\alpha s Q_1 + r_1^{-1} sP) + a_2(\alpha s Q_2 + r_2^{-1} sP) + \cdots + a_n(\alpha s Q_n + r_n^{-1} sP), P') = e(\sum_{i=1}^{n} a_i(\alpha s Q_i + r_i^{-1} sP), P') = e(\sum_{i=1}^{n} a_i T_i, P')$, and negotiate a common session conference key SCK $= H_3(K)$.

In our proposed interdomain scheme (Section 5.2), U_{ij} can generate a valid message $(b_{ij}, B_{ij}, T_{ij}, \beta_{ij})$ and verify the correctness of the message because

$$\begin{aligned}
V_{h,l} &= \prod_{h=1}^{m} e\left(\sum_{l=1}^{n_h} a_{h,l} b_{h,l} + \alpha \sum_{l=1}^{n_h} a_{h,l} Q_{h,l}, s_h P'\right) \\
&= e\Big(\left(a_{1,1} r_{1,1}^{-1} + a_{1,2} r_{1,2}^{-1} + \cdots + a_{1,n_1} r_{1,n_1}^{-1}\right) P \\
&\qquad + \left(\alpha a_{1,1} Q_{1,1} + \alpha a_{1,2} Q_{1,2} + \cdots + \alpha a_{1,n_1} Q_{1,n_1}\right), s_1 P'\Big) \\
&\quad \cdot e\Big(\left(a_{2,1} r_{2,1}^{-1} + a_{2,2} r_{2,2}^{-1} + \cdots + a_{2,n_2} r_{2,n_2}^{-1}\right) P \\
&\qquad + \left(\alpha a_{2,1} Q_{2,1} + \alpha a_{2,2} Q_{2,2} + \cdots + \alpha a_{2,n_2} Q_{2,n_2}\right), \\
&\qquad s_2 P'\Big) \\
&\quad \cdot e\Big(\left(a_{m,1} r_{m,1}^{-1} + a_{m,2} r_{m,2}^{-1} + \cdots + a_{m,n_m} r_{m,n_m}^{-1}\right) P \\
&\qquad + \left(\alpha a_{m,1} Q_{m,1} + \alpha a_{m,2} Q_{m,2} + \cdots + \alpha a_{m,n_m} Q_{m,n_m}\right), \\
&\qquad s_m P'\Big) \\
&= e\Big(a_{1,1}\left(\alpha s_1 Q_{1,1} + r_{1,1}^{-1} s_1 P\right) \\
&\qquad + a_{1,2}\left(\alpha s_1 Q_{1,2} + r_{1,2}^{-1} s_1 P\right) + \cdots \\
&\qquad + a_{1,n_1}\left(\alpha s_1 Q_{1,n_1} + r_{1,n_1}^{-1} s_1 P\right), P'\Big) \\
&\quad \cdot e\Big(a_{2,1}\left(\alpha s_2 Q_{2,1} + r_{2,1}^{-1} s_2 P\right) + a_{2,2}\left(\alpha s_2 Q_{2,2} + r_{2,2}^{-1} s_2 P\right) \\
&\qquad + a_{2,n_2}\left(\alpha s_2 Q_{2,n_2} + r_{2,n_2}^{-1} s_2 P\right), P'\Big) \\
&\qquad \vdots \\
&\quad \cdot e\Big(a_{m,1}\left(\alpha s_m Q_{m,1} + r_{m,1}^{-1} s_m P\right) \\
&\qquad + a_{m,2}\left(\alpha s_m Q_{m,2} + r_{m,2}^{-1} s_m P\right)
\end{aligned}$$

$$\begin{aligned} &+\cdots+a_{m,n_m}\left(\alpha s_m Q_{m,n_m}+r_{m,n_m}^{-1}s_m P\right),P'\Big) \\ &=e\left(\sum_{h=1}^{m}\sum_{l=1}^{n_h}a_{h,l}\left(\alpha s_h Q_{h,l}+r_{h,l}^{-1}s_h P\right),P'\right) \\ &=e\left(\sum_{h=1}^{m}\sum_{l=1}^{n_h}a_{h,l}T_{h,l},P'\right). \end{aligned} \tag{9}$$

□

6.2. Known Session Key Security. After given broadcast messages and previous session keys according to the $\mathcal{IDCK}$ scheme, an adversary makes a Test query and then receives a random string or a current session key. The adversary can continue asking for broadcast messages and other session keys. If no polynomial-time adversary can decide whether the received string is the current session key or not with nonnegligible advantage, we say that the $\mathcal{IDCK}$ scheme satisfies known session key security.

Definition 7 (known session key security). An $\mathcal{IDCK}$ scheme is with known session key security if no polynomial-time adversary can decide if a challenge string is a current session key or a random string under the knowledge of previous session keys with the probability at least $(\varepsilon + 1/2)$ where ε, called the advantage, is nonnegligible.

Theorem 8. *If an adversary $\mathcal{A}$ can $(t, q_E, q_S, q_C, q_R, q_{H_1}\varepsilon)$-decide whether the string received from a Test query is the session key SCK held by $\Pi_{\mathcal{U}}^{\ell}$ or not with advantage at least ε, where t is the running time and q_E, q_S, q_C, q_R, and q_{H_1} are the numbers of making Execute queries, Send queries, Corrupt queries, Reveal queries, and H_1 queries, respectively, there exists an algorithm which can solve the n-DLDH problem with advantage at least ε' in time t', where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{q_0}, \\ t' &\approx t+q_E\mathcal{O}(t_E)++q_S\mathcal{O}(t_S)+q_C\mathcal{O}(t_C) \\ &\quad +q_R\mathcal{O}(t_R)+q_{H_1}\mathcal{O}(t_{H_1})+\mathcal{O}(n^2), \end{aligned} \tag{10}$$

$q_0 = q_E + q_S + q_C + q_R + q_H$, t_E, t_S, t_C, t_R, and t_{H_1} are the computing time of the Execute oracle, the Send oracle, the Corrupt oracle, the Reveal oracle, and H_1 oracle, respectively.

Proof. Initially, we construct a simulator $\mathcal{B}$ which prepares the pairing parameters and simulates the system as follows. $\mathcal{B}$ randomly picks $s \in \mathbb{Z}_q^*$ as the system master private key and computes (sP, sP') as the system master public key. $\mathcal{B}$ computes each user's long-term public/private key pair (Q_i, sQ_i). $\mathcal{B}$ allows $\mathcal{A}$ to make the following queries

(i) *Execute*$(\Pi_{\mathcal{U}}^{t})$: $\mathcal{A}$ can request $\mathcal{U}$ that is a set of users who are chosen by itself to run the key agreement protocol in session t. $\mathcal{B}$ follows the protocol (Section 5.1) to produce every $(r_i, b_i, B_i, T_i, \beta_i, k_i)$, and $K = \sum_{U_i\in\mathcal{U}} k_i P$, and then records them. Finally, $\mathcal{B}$ responds every (b_i, B_i, T_i, β_i) to $\mathcal{A}$, where $U_i \in \mathcal{U}$.

(ii) *Send*$(\Pi_{\widetilde{\mathcal{U}}}^{t}, m)$: if $\mathcal{A}$ actively broadcasts the messages of users $\widetilde{\mathcal{U}} \subset \mathcal{U}$ to run the key agreement protocol in session t, $\mathcal{B}$ follows the protocol (Section 5.1) to produce every $(r_i, b_i, B_i, T_i, \beta_i, k_i)$ and generate the session conference key in the end of the protocol and then records them. Finally, $\mathcal{B}$ responds (b_i, B_i, T_i, β_i) to $\mathcal{A}$ for each $U_i \in \mathcal{U} - \widetilde{\mathcal{U}}$.

(iii) *Reveal*$(\Pi_{\mathcal{U}}^{t})$: if $\Pi_{\mathcal{U}}^{t}$ does not exist, $\mathcal{B}$ creates $\Pi_{\mathcal{U}}^{t}$. $\mathcal{B}$ returns the session conference key SCK $= H_3(K)$.

(iv) *Corrupt*(U_i): $\mathcal{B}$ returns (Q_i, sQ_i) to $\mathcal{A}$.

(v) $H_1(m)$: after given $m \in \mathbb{G}_1$, $\mathcal{B}$ randomly chooses $\alpha \in \{0,1\}^{\lambda}$, returns $H_1(m) = \alpha$, and stores (m, α) in a list, called H_1-list.

(vi) *Test*$(\Pi_{\mathcal{U}^*}^{\ell})$: $\mathcal{B}$ guesses that $\mathcal{A}$ will send a Test query at the hth session in advance. If $\mathcal{A}$ makes a *Test* query at the ℓth session, where $\ell \neq h$, $\mathcal{B}$ randomly answers "YES" or "NO" to the n-DLDH problem. When $\mathcal{A}$ makes a *Test* query at the ℓth session, where $\ell = h$, $\mathcal{B}$ checks whether there exists U_i which has been corrupted or not, where $U_i \in \mathcal{U}^*$ and $|\mathcal{U}^*| = n$. If one of them has been corrupted, $\mathcal{B}$ returns $\perp$ and aborts the game. Otherwise, $\mathcal{B}$ is given the parameters of an instance of the n-DLDH problem, Z, P, and $(x_iP, x_ix_{n+j}P)$'s for all i, j with $1 \leq i, j \leq n$, and $i \neq j$, and takes the advantage of $\mathcal{A}$ to decide whether $Z = (x_{n+1} + x_{n+2} + \cdots + x_{2n})P$ or not. $\mathcal{B}$ sets the public messages $(b_i^*, B_i^*, T_i^*, \beta_i^*)$ of every user in $\mathcal{U}^*$ in the ℓth session as follows.

First, $\mathcal{B}$ forms $b_1^* = x_1P, b_2^* = x_2P, \ldots, b_n^* = x_nP$, $K^* = Z$, $\alpha^* = H_1(K^*)$, and prepares $B_i^* = (x_1x_{n+i}P, x_2x_{n+i}P, \ldots, x_{i-1}x_{n+i}P, \perp, x_{i+1}x_{n+i}P, \ldots, x_nx_{n+i}P)$, for each $U_i^* \in \mathcal{U}^*$; that is,

$$\begin{array}{c|ccccccc} & U_1^* & U_2^* & \ldots & U_i^* & \ldots & U_n^* \\ \hline B_1^* & \perp & x_2x_{n+1}P & \ldots & x_ix_{n+1}P & \ldots & x_nx_{n+1}P \\ B_2^* & x_1x_{n+2}P & \perp & \ldots & x_ix_{n+2}P & \ldots & x_nx_{n+2}P \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ B_i^* & x_1x_{n+i}P & x_2x_{n+i}P & \ldots & \perp & \ldots & x_nx_{n+i}P \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ B_n^* & x_1x_{2n}P & x_2x_{2n}P & \ldots & x_ix_{2n}P & \ldots & \perp \end{array} \tag{11}$$

$\mathcal{B}$ sets $T_i^* = s\alpha^*Q_i + sx_iP$ and computes $\beta_i^* = H_2(T_i^*, \alpha^*)$ and SCK$^* = H_3(K^*)$, where $1 \leq i \leq n$. $\mathcal{B}$ responds every $(b_i^*, B_i^*, T_i^*, \beta_i^*)$ and SCK* to $\mathcal{A}$.

$\mathcal{A}$ can continue making the queries of *Execute*$(\Pi_{\mathcal{U}}^{t})$, *Send*$(\Pi_{\mathcal{U}}^{t})$, *Reveal*$(\Pi_{\mathcal{U}}^{t})$, and *Corrupt*(U_i), where $t \neq \ell$ and $U_i \notin \mathcal{U}^*$, until $\mathcal{A}$ outputs a bit b. If SCK* is the key in session ℓ from $\mathcal{A}$'s point of view, $\mathcal{A}$ will output $b = 1$; otherwise, $b = 0$. If $b = 1$, $\mathcal{B}$ outputs "YES"; that is, $Z = (x_{n+1}+x_{n+2}+\cdots+x_{2n})P$; otherwise, $\mathcal{B}$ outputs "NO".

If the adversary can compromise known session key security of the scheme with advantage at least ε, $\mathcal{B}$ can solve the n-DLDH problem with the advantage at least $\varepsilon' \geq (1/q_0)\ (1/2 + \varepsilon) + (1/2)((q_0 - 1)/q_0) - 1/2 = \varepsilon/q_0$. □

By Theorem 8, we can solve the n-DLDH problem in polynomial time with nonnegligible advantage if there exists a polynomial-time adversary that can break the known session key security with nonnegligible advantage of the proposed single domain key agreement scheme.

As for the proof of the interdomain case, we can let n be the number of the total users from all domains; that is, $n = N$. Then, by the proof of Theorem 8, the n-DLDH problem can be solved if the adversary can distinguish the session key from a random string in the proposed interdomain key agreement scheme.

6.3. Key-Compromise Impersonation. An adversary is given all users' long-term keys by making Corrupt queries except the one that he claims to impersonate. If no adversary can output the correct messages of the user with nonnegligible probability, the $\mathcal{IDCK}$ scheme can withstand key-compromise impersonation.

Definition 9 (key-compromise impersonation). An $\mathcal{IDCK}$ scheme can withstand key-compromise impersonation if no adversary can have nonnegligible probability to impersonate a user without the long-term private key of the user.

Lemma 10 (the forking lemma [35]). *Let $T_i = \alpha s Q_i + r_i^{-1} sP$ be a valid authentic parameter of user U_i, where sQ_i is U_i's private key, sP is TA's public key, r_i^{-1} is randomly chosen by U_i, and α is a hashed value of K shared by all users. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine. Given only the public data of the key agreement scheme as input, if $\mathcal{A}$ can find, with nonnegligible probability, a valid authentic parameter T_i with α, then, with nonnegligible probability, a replay of this machine, with the same random tape and a different value returned by the random oracle, can output two valid authentic parameters T_i with α and T_i' with α', such that $\alpha \neq \alpha'$.*

Lemma 11 (the splitting lemma [36]). *Let $A \subset X \times Y$ such that* $\Pr[(x, y) \in A] = |A|/(|X| \times |Y|) \geq \delta$. *For any $\rho < \delta$, define $B = \{x \in X \mid \Pr[(x, y) \in A] \geq \delta - \rho\}$ and $\overline{B} = X \setminus B$. Then the following statements hold:*

(1) $\Pr[x \in B] = |B|/|X| \geq \rho$,

(2) $\forall x \in B, \Pr[(x, y) \in A] \geq \delta - \rho$,

(3) $\Pr[x \in B \mid (x, y) \in A] \geq \rho/\delta$.

Lemma 12. *Assume that $e(\sum_{i=1}^{n} a_i b_i + \alpha \sum_{i=1}^{n} a_i Q_i, sP') = e(\sum_{i=1}^{n} a_i T_i, P')$. Let E be an event that occurs if there is at least one T_i such that $e(a_i b_i + \alpha a_i Q_i, sP') \neq e(a_i T_i, P')$. Then, the probability* $\Pr[E] \leq 1/2^q$, *where q is a security parameter.*

Proof. The proof is using the technique of the small exponents test in [37]. If $e(a_i b_i + \alpha a_i Q_i, sP') \neq e(a_i T_i, P')$ for some i, then $T_i \neq sb_i + s\alpha Q_i$. That is, there exists $c_i \neq 0 \pmod q$ such that $T_i = sb_i + s\alpha Q_i + c_i P$.

Let $T_j = sb_j + s\alpha Q_j + c_j P$ where $\forall j \in \{1, \ldots, n\} - \{i\}$ and $c_j \in \{0, \ldots, q-1\}$. As $e(\sum_{i=1}^{n} a_i b_i + \alpha \sum_{i=1}^{n} a_i Q_i, sP') = e(\sum_{i=1}^{n} a_i T_i, P')$, $\sum_{i=1}^{n} c_i a_i \equiv 0 \pmod q$. Hence, $a_i \equiv -c_i^{-1}(c_1 a_1 + c_2 a_2 + \cdots + c_{i-1} a_{i-1} + c_{i+1} a_{i+1} + \cdots + c_n a_n) \pmod q$. Since a_i is randomly chosen from $\mathbb{Z}_q^*$, $\Pr[E] \leq 1/2^q$. □

Theorem 13. *If an adversary $\mathcal{A}$ can $(t, q_S, q_C, q_R, q_H, q_{H_1}, \varepsilon)$-impersonate a user U_i^* without the long-term private key of U_i^* with probability at least ε, where t is the running time, q_S, q_C, q_R, q_H, and q_{H_1} are the numbers of making Send queries, Corrupt queries, Reveal queries, H queries, and H_1 queries, respectively, there exists an algorithm to solve the variant-CDH problem with probability at least ε' in time t', where*

$$\varepsilon' \geq \left(\frac{(1/|\mathcal{U}|)(1 - 1/|\mathcal{U}|)\left(1 - 1/2^h\right)(\varepsilon - 1/2^q)}{2}\right)^2,$$
$$t' \approx 2\left(t + q_S \mathcal{O}(t_S) + q_C \mathcal{O}(t_C) + q_R \mathcal{O}(t_R) + q_H \mathcal{O}(t_H) + q_{H_1} \mathcal{O}\left(t_{H_1}\right)\right) + \mathcal{O}\left(n^2\right), \quad (12)$$

h is the length of H_1's output, $\mathcal{U}$ is a set of users, and t_H is the computing time of H oracle.

Proof. At first, $\mathcal{B}$ inputs κ to generate pairing parameters and a variant-CDH tuple $(P, aP, bP, P', aP') \in \mathbb{G}_1$, where $a, b \in_R \mathbb{Z}_q^*$. We will show that $\mathcal{B}$ can solve the variant-CDH problem with the assistance of an adversary $\mathcal{A}$. $\mathcal{B}$'s task is to compute and output the value $cP = abP$.

$\mathcal{B}$ simulates the system as follows. We define (aP, aP') as the system master public key and $\mathcal{B}$ does not know the master private key a. H and H_1 are two random oracles simulated by $\mathcal{B}$ to respond the queries to H and H_1, respectively. $\mathcal{B}$ randomly chooses one user U_i^* and let bP be U_i^*'s long-term public key. Except U_i^*, $\mathcal{B}$ computes other users' long-term public/private key pairs by H. $\mathcal{B}$ allows $\mathcal{A}$ to make the following queries.

(i) $H(\cdot)$: after given $m \in \{(\text{TA's ID} \| U_i\text{'s ID}) \mid 1 \leq i \leq q\}$, $\mathcal{B}$ responds the query, $H(m)$, and maintains the H-list as follows.

If $m = (\text{TA's ID} \| U_i^*\text{'s ID})$, $\mathcal{B}$ returns $H(m) = bP$ and stores a record $(U_i^*, bP, \perp, \perp)$ in the H-list. Otherwise, $\mathcal{B}$ randomly chooses $u_i \in \mathbb{Z}_q^*$, returns $H(m) = u_i P$, and stores a record $(U_i, u_i P, u_i aP, u_i)$ in the H-list, where $u_i P$ and $u_i aP$ are the long-term public and private keys, respectively, of U_i.

(ii) $H_1(\cdot)$: after given $m \in \mathbb{G}_1$, $\mathcal{B}$ randomly chooses $\alpha \in \{0,1\}^{\ell}$, returns $H_1(m) = \alpha$, and stores a record (m, α) in the H_1-list.

(iii) *Execute*$(\Pi_{\mathcal{U}}^t)$: $\mathcal{A}$ can choose $\mathcal{U}$ and ask $\mathcal{B}$ to run the key agreement protocol. $\mathcal{B}$ returns the public messages (b_i, B_i, T_i, β_i)'s of all U_i's in $\mathcal{U}$ to $\mathcal{A}$. $\mathcal{B}$ produces the messages as follows.

If $U_i^* \in \mathcal{U}$, $\mathcal{B}$ picks $k_i, z, \alpha \in_R \mathbb{Z}_q^*$ and computes $b_i = zP - \alpha bP = (z - \alpha b)P$, $T_i = zaP$, and $\beta_i = H_2(T_i, \alpha)$. For each of the other U_j's $\in \mathcal{U}$, $\mathcal{B}$ randomly picks $r_j, k_j \in \mathbb{Z}_q^*$ and computes $b_j = r_j^{-1} P$, $T_j = \alpha u_j aP + r_j^{-1} aP$, and $\beta_j = H_2(T_j, \alpha)$. Thus, $\mathcal{B}$ can follow the protocol to compute B_i for each $U_j \in \mathscr{U}$ and sets $\alpha = H_1(K)$ where $K = \sum_{U_j \in \mathscr{U}} k_j P$.

(iv) *Send*$(\Pi^t_{\widetilde{\mathscr{U}}}, m)$: if $\mathscr{A}$ actively broadcasts the message of users $\widetilde{\mathscr{U}} \subset \mathscr{U}$ to run the key agreement protocol in session t, $\mathscr{B}$ can produce each (b_i, B_i, T_i, β_i) of the other $U_i \in \mathscr{U} - \widetilde{\mathscr{U}}$ as that in the *Execute* query. Once if $\mathscr{U} - \widetilde{\mathscr{U}} = \{U_i^*\}$, $\mathscr{B}$ can obtain K from H_1-list while $\mathscr{A}$ is making the H_1 query for α.

(v) *Reveal*$(\Pi^t_{\mathscr{U}})$: $\mathscr{B}$ looks up the H_1-list to obtain K by checking if $H_2(T_i, \alpha) = \beta_i$ and returns the session conference key SCK $= H_3(K)$.

(vi) *Corrupt*(U_i): if $U_i = U_i^*$, $\mathscr{B}$ returns $\perp$ and aborts the game. Otherwise, $\mathscr{B}$ searches U_i in the H-list. If U_i is not in the H-list, $\mathscr{B}$ calls H to produce and store $(U_i, u_i P, u_i aP, u_i)$ in the H-list. Then, $\mathscr{B}$ returns $u_i aP$.

If $\mathscr{A}$ impersonates U_i^* at some point, according to Lemma 12, it must send out $T_i = \alpha a Q_i^* + r_i^{-1} aP$ with the probability at least $\varepsilon - 1/2^q$, where $H_1(K) = \alpha$. Once $\mathscr{A}$ produces a correct T_i, $\mathscr{B}$ replays $\mathscr{A}$ with the same random tape by forking lemma. At this time, $\mathscr{A}$ gets two different hashed values α and α' and generates two valid $T_i = \alpha abP + r_i^{-1} aP$ and $T_i' = \alpha' abP + r_i^{-1} aP$. $\mathscr{B}$ can compute $cP = (\alpha - \alpha')^{-1}(T_i - T_i') = abP$. Finally, $\mathscr{B}$ outputs abP.

Let X and Y be the set of any possible input messages of the random tape and H_1, respectively. $\mathscr{A}$ fails in making *Corrupt* queries with probability $1/|\mathscr{U}|$. $\mathscr{A}$ guesses the value $\alpha = H_1(K)$ without making H_1 queries with probability 2^{-h}, where h is the length of H_1's output. Therefore, the probability is $\delta = (1/|\mathscr{U}|)(1 - 1/|\mathscr{U}|)(1 - 1/2^h)(\varepsilon - 1/2^q)$. By splitting lemma, we set $\rho = \delta/2$ such that $\delta - \rho = \delta - (\delta/2) = \delta/2$. Overall, $\mathscr{B}$ performs two executions of $\mathscr{A}$, so that we have

$$\begin{aligned} \varepsilon' &\geq \left(\frac{\delta}{2}\right)^2 \\ &= \left(\frac{(1/|\mathscr{U}|)(1 - 1/|\mathscr{U}|)\left(1 - 1/2^h\right)(\varepsilon - 1/2^q)}{2}\right)^2, \\ t' &\approx 2\left(t + q_S \mathscr{O}(t_S) + q_C \mathscr{O}(t_C) + q_R \mathscr{O}(t_R)\right. \\ &\quad \left. + q_H \mathscr{O}(t_H) + q_{H_1} \mathscr{O}\left(t_{H_1}\right)\right) + \mathscr{O}\left(n^2\right). \end{aligned} \tag{13}$$

□

As for the proof of the interdomain case, (aP, aP', bP), the parameters of the variant-CDH problem are the public keys of some TA and some user who belongs to the CA, respectively. The private keys of the other TAs and users are randomly generated by $\mathscr{B}$. Likewise, the proposed interdomain key agreement scheme can also withstand key-compromise impersonation.

6.4. Forward Secrecy. After given broadcast messages, session keys, and all users' long-term keys according to an $\mathscr{IDCK}$ scheme, an adversary makes a Test query and then receives a random string or a session key. The adversary can continue making queries. If no adversary can decide whether the received string is a session key or not with nonnegligible advantage, we say that the $\mathscr{IDCK}$ scheme satisfies forward secrecy.

Definition 14 (forward secrecy). An $\mathscr{IDCK}$ scheme has forward secrecy if any adversary who obtains the other session keys and all users' long term keys can distinguish a previous session key from a random string with the probability at least $(\varepsilon + 1/2)$ where ε, called the advantage, is negligible.

After running time at most t, making at most q_{H_1} queries to H_1, q_E *Execute* queries, q_S *Send* queries, q_C *Corrupt* queries, and q_R *Reveal* queries, an adversary $\mathscr{A}$ can obtain all users' long-term private keys and receives a string through a Test query. $\mathscr{A}(t, q_E, q_S, q_C, q_R, q_{H_1}, \varepsilon)$-FS-breaks our $\mathscr{IDCK}$ scheme if he can determine whether the received string is a previous session key negotiated by users with nonnegligible advantage at least ε.

Theorem 15. *If an adversary $\mathscr{A}$ can $(t, q_E, q_S, q_C, q_R, q_{H_1}, \varepsilon)$-FS-determine whether SCK is a previous session key or not, where t is the running time, q_E, q_S, q_C, q_R, and q_{H_1} are the numbers of making Send queries, Corrupt queries, Reveal queries, and hash queries to H_1, respectively, there exists an algorithm to solve the n-DLDH problem with nonnegligible advantage at least ε' in time t', where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{q_0}, \\ t' &\approx t + q_E \mathscr{O}(t_E) + q_S \mathscr{O}(t_S) + q_C \mathscr{O}(t_C) \\ &\quad + q_R \mathscr{O}(t_R) + q_{H_1} \mathscr{O}\left(t_{H_1}\right) + \mathscr{O}\left(n^2\right), \end{aligned} \tag{14}$$

and $q_0 = q_E + q_S + q_C + q_R + q_{H_1}$.

Proof. The simulation and the queries answered by $\mathscr{B}$ are the same as those in the proof of Theorem 8 except that $\mathscr{A}$ is allowed to make *Corrupt*(U_i) query, where $U_i \in \mathscr{U}^*$, after making *Test*$(\Pi^\ell_{\mathscr{U}^*})$. That is, $\mathscr{A}$ has to activate the ℓth session first before any $U_i \in \mathscr{U}^*$ is corrupted. In the end, $\mathscr{B}$ can decide whether $Z = (x_{n+1} + x_{n+2} + \cdots + x_{2n})P$ or not according to the value of b output by $\mathscr{A}$, where

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{q_0} \\ t' &\approx t + q_E \mathscr{O}(t_E) + q_S \mathscr{O}(t_S) + q_C \mathscr{O}(t_C) \\ &\quad + q_R \mathscr{O}(t_R) + q_{H_1} \mathscr{O}\left(t_{H_1}\right) + \mathscr{O}\left(n^2\right). \end{aligned} \tag{15}$$

□

Similarly, the proposed interdomain key agreement scheme can also achieve forward secrecy.

6.5. Key Control. An adversary who is one of the users can obtain broadcast messages, session keys, and the other users' long-term keys. Then, the adversary is given a preselected value. If the adversary cannot make the given preselected value become a new session key with nonnegligible probability, the $\mathcal{IDCK}$ scheme can withstand key control attacks.

Definition 16 (key control). An $\mathcal{IDCK}$ scheme can withstand key control attacks if no adversary can predict a session key or preselect a session key with nonnegligible probability.

Theorem 17. *If an adversary $\mathcal{A}$ can $(t, q_E, q_S, q_C, q_R, q_H, \varepsilon)$-predict a session key or preselect a session key with probability at least ε, where t is the running time, q_E, q_S, q_C, q_R, and q_H are the numbers of making Send queries, Corrupt queries, Reveal queries, and hash queries to H, respectively, there exists an algorithm to solve the CDH problem with nonnegligible probability at least ε' in time t', where*

$$\begin{aligned}\varepsilon' &\geq \frac{\varepsilon}{q_0},\\ t' &\approx t + q_E\mathcal{O}(t_E) + q_S\mathcal{O}(t_S) + q_C\mathcal{O}(t_C)\\ &\quad + q_R\mathcal{O}(t_R) + q_H\mathcal{O}(t_H) + \mathcal{O}(n^2),\end{aligned} \tag{16}$$

and $q_0 = q_E + q_S + q_C + q_R + q_H$.

Proof. At first, $\mathcal{B}$ inputs κ to generate pairing parameters and a CDH triple $(P, aP, bP) \in \mathbb{G}_1$, where $a, b \in_R \mathbb{Z}_q^*$. We show that an algorithm $\mathcal{B}$ can solve the CDH problem with the help of an adversary $\mathcal{A}$. $\mathcal{B}$'s task is to compute and output the value $cP = abP$.

$\mathcal{B}$ simulates the system as follows. We define that (sP, sP') is the system master public key by randomly choosing the master private key s. $\mathcal{B}$ allows $\mathcal{A}$ to make the following queries.

(i) $H(\cdot)$: after given $m \in \{(\text{TA's ID}\|U_i\text{'s ID}) \mid 1 \leq i \leq q\}$, $\mathcal{B}$ responds the query $H(m)$ and maintains the H-list as follows: $\mathcal{B}$ randomly chooses $u_i \in \mathbb{Z}_q^*$, returns $H(m) = u_iP$, and stores a record (U_i, u_iP, u_isP, u_i) in the H-list.

(ii) *Execute*$(\Pi^t_{\mathcal{U}})$: $\mathcal{A}$ can choose a set $\mathcal{U}$ of users to run the key agreement protocol in session t. $\mathcal{B}$ follows the protocol to produce every $(b_i, B_i, T_i, \beta_i, k_i)$ and computes $K = \sum_{U_i \in \mathcal{U}} k_iP$. Finally, $\mathcal{B}$ responds every (b_i, B_i, T_i, β_i) to $\mathcal{A}$.

(iii) *Send*$(\Pi^t_{\widetilde{\mathcal{U}}}, m)$: if $\mathcal{A}$ can actively send the message of users $\widetilde{\mathcal{U}} \subset \mathcal{U}$ to run the key agreement protocol in session t, $\mathcal{B}$ can produce (b_i, B_i, T_i, β_i) of each $U_i \in \mathcal{U} - \widetilde{\mathcal{U}}$ as that in the *Execute* query.

(iv) *Reveal*$(\Pi^t_{\mathcal{U}})$: $\mathcal{B}$ returns the session conference key $\text{SCK} = H_3(K)$.

(v) *Corrupt*(U_i): $\mathcal{B}$ searches U_i in H-list and returns u_isP. If U_i is not in H-list, $\mathcal{B}$ calls H oracle to produce (U_i, u_iP, u_isP, u_i) and returns u_isP.

When $\mathcal{A}$ finishes making the above queries, $\mathcal{B}$ sets $K = xP + bP$, where $x \in_R \mathbb{Z}_q^*$, and returns it to $\mathcal{A}$. Suppose that $\mathcal{A}$ is user U_i and $\mathcal{B}$ simulates other users $U_1, U_2, \ldots, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n$ to negotiate a session conference key with $\mathcal{A}$. $\mathcal{B}$ returns $\{(b_1, B_1, T_1, \beta_1), \ldots, (b_{i-1}, B_{i-1}, T_{i-1}, \beta_{i-1}), (b_{i+1}, B_{i+1}, T_{i+1}, \beta_{i+1}), \ldots, (b_n, B_n, T_n, \beta_n)\}$ to $\mathcal{A}$, where $x = k_1 + \cdots + k_{i-1} + k_{i+1} + \cdots + k_n$ and $b_1 = aP$. $\mathcal{A}$ must broadcast $B_i = (k_ib_1, \ldots, k_ib_{i-1}, k_ib_{i+1}, \ldots, k_ib_n)$ in round 2 such that $(k_1 + k_2 + \cdots + k_n)P = (x + b)P$. Therefore, $k_iP = bP$ and $k_ib_1 = abP$.

Overall, we have that

$$\begin{aligned}\varepsilon' &\geq \frac{\varepsilon}{q_0},\\ t' &\approx t + q_E\mathcal{O}(t_E) + q_S\mathcal{O}(t_S) + q_C\mathcal{O}(t_C)\\ &\quad + q_R\mathcal{O}(t_R) + q_H\mathcal{O}(t_H) + \mathcal{O}(n^2).\end{aligned} \tag{17}$$

□

Following a proof similar to that for the above theorem, we can show that the proposed interdomain key agreement scheme withstands key control attacks.

7. Discussions

In this section, we compare our schemes with the schemes of [15–19, 24–27] according to the properties shown in Table 1 and performance factors in Table 2. We prove that our scheme in single domain satisfies all security attributes in Section 6 and it can be easily extended to an interdomain version. Besides, we do our best to extend each scheme in [15–19, 24–27] to an interdomain one, but only the schemes [17, 18, 24, 25] can achieve this goal with little modification. Therefore, we use the straightforward way [38] to extend the remaining schemes [15, 16, 19, 26, 27] to interdomain versions for performance comparisons. The result is shown in Table 3.

In the design of a conference key agreement scheme, the other researchers always divide the protocol into two main parts: the authentication stage and the stage of the construction of a common secret value. These proposed schemes [15–19, 24–26] all perform the authentication stage first and then make a session conference key, so that they have some problems in security. We have showed that [15–19, 24–26] are insecure in Section 3 because an adversary $\mathcal{A}$ can get the valid parameters in the authentication stage and replay them in different sessions. In particular, we design our protocol in inverse order such that users share a common secret value first and then run the authentication stage. If we only consider a single TA, some schemes are more efficient than our proposed scheme. However, none of them are secure.

8. Conclusion

Many researchers proposed two- or three-party identity-based authenticated key agreement schemes, but general multiparty authenticated key agreement schemes are rare. When the number of the members is more than three, the multiparty scheme is difficult to achieve both security and efficiency at the same time. In this paper, we proposed a novel

Table 1: Properties.

	[24]	[18]	[15, 16]	[17]	[19]	[25]	[27]	[26]	Ours
Know session key security	YES	YES	YES	NO	YES	YES	YES	YES	YES
(Perfect) forward secrecy	YES	YES	Δ	NO	Δ	YES	YES	Δ	YES
No key-compromise impersonation	NO	NO	NO	YES	NO	NO	YES	NO	YES
No key control	YES	YES	YES	YES	YES	YES	YES	YES	YES

Δ: partial forward secrecy.

Table 2: Performance comparisons in single domain.

Communication cost	[24]	[18]	[15, 16]	[17]	[19]	[25]	[27]	[26] (1)	[26] (2)	Ours
Round	2	2	2	1	1	3	3	1	2	3
Bandwidth	3	3	3	3	$(n-1)$	4	$(n-1)$	$(n+1)$	3	$(n+1)$
Computation cost	[24]	[18]	[15, 16]	[17]	[19]	[25]	[27]	[26] (1)	[26] (2)	Ours
Point addition	9	6	$3n-3$	$2n-3$	$n-1$	$2n+3$	$n+1$	—	$n-2$	$4n-2$
Scalar multiplication $(\mathbb{G}_1, \mathbb{G}_2)$	8	4	$n+5$	$n+3$	n	$2n+6$	$3n-1$	—	2	$4n+3$
Scalar multiplication $(\mathbb{G}_T)$	$2n$	—	$n-1$	$2n-2$	—	—	—	$2n-2$	2	—
Pairing	4	4	4	$2n+1$	1	$n+5$	$4n-4$	$2n-2$	4	2
Exponentiation	—	—	$n-2$	—	—	—	—	—	—	—
Increment/decrement of computation time $n = 100$	↓22%	↓19%	↓78%	↓98%	↑48%	↓97%	↓196%	↓98%	↓19%	—

Bandwidth: the total number of messages (points) sent by a user.
n: the number of users.
From [28], 1 exponentiation ≈ 240 scalar multiplications, 1 point addition ≈ 0.12 scalar multiplication, 1 scalar multiplication in $\mathbb{G}_1$ ≈ 29 scalar multiplications, 1 scalar multiplication in $\mathbb{G}_2$ ≈ 1 scalar multiplication, and 1 pairing ≈ 10 exponentiations.

efficient identity-based conference key agreement scheme and proved its security via formal method. Furthermore, our scheme can be extended to an interdomain one.

Consider the application of key agreement among the employees in a company. The focus of the past papers is that how users interact under a single TA. It means that the employees of a company can negotiate a common session key when they want to organize a private conference. But now, our schemes provide more flexibility for the users. Even if the users register with different TAs, they can also negotiate a common session key easily. In other words, when two or more companies want to hold a conference, the employees from different companies can still compute a common session key by our interdomain key agreement scheme.

Our conference key agreement can be applied to the ad hoc networks, too. In the wireless environment, reliable communication and authentication is desired. By performing our method, it is unnecessary for ad hoc sensors to store a large amount of data in advance and they can still negotiate a session key under mutual authentication. We can ensure that the transmitted messages are reliable and also secure against malicious sensors in wireless networks.

Appendices

A. Shi et al.'s Scheme [19]

In [19], KGC gives each user, U_i, $Q_i = (I_i s_1 + s_2)P$ as a public key and $S_i = (I_i s_1 + s_2)^{-1}P$ as a private key, where $I_i = H(\mathrm{ID}_i)$ is the hashed value of the identity information ID_i; s_1 and s_2 are randomly chosen by KGC. Each U_i $(1 \le i \le n)$ computes the key K as follows.

Step 1. U_i randomly chooses a_i, computes $T_{i,j} = a_i Q_j$ $(1 \le j \le n, j \ne i)$, and then sends $T_{i,j}$ to U_j for each j.

Step 2. After receiving $T_{1,i}, T_{2,i}, \ldots, T_{i-1,i}, T_{i+1,i}, \ldots, T_{n,i}$ from the other users, U_i computes

$$\begin{aligned} K &= e\left(T_{1,i} + T_{2,i} + \cdots + T_{i-1,i} + a_i Q_i + T_{i+1,i} + \cdots + T_{n,i}, S_i\right) \\ &= e\left(Q_i, S_i\right)^{a_1 + a_2 + \cdots + a_n} \\ &= e\left(P, P\right)^{a_1 + a_2 + \cdots + a_n}. \end{aligned} \tag{A.1}$$

There are two security problems shown as follows.

Table 3: Performance comparisons in interdomain.

Communication cost	[24]	[18]	[15, 16]	[17]	[19]	[25]	[27]	[26] (1)	[26] (2)	Ours
Round	2	2	$\leq 2 + \lceil \log_3(m) \rceil$	1	$\leq 1 + \lceil \log_3(m) \rceil$	3	$\leq 3 + \lceil \log_3(m) \rceil$	$\leq 1 + \lceil \log_3(m) \rceil$	$\leq 2 + \lceil \log_3(m) \rceil$	3
Bandwidth	3	3	$< 3 + 5m(m-1)$	3	$< (n-1) + 5m(m-1)$	4	$< (n-1) + 5m(m-1)$	$< (n+1) + 5m(m-1)$	$< 3 + 5m(m-1)$	$(n+1)$
Computation cost	**[24]**	**[18]**	**[15, 16]**	**[17]**	**[19]**	**[25]**	**[27]**	**[26] (1)**	**[26] (2)**	**Ours**
Point addition	9	6	$3n-3$	$2(n-m)-1$	$n-1$	$2n+m+2$	$n+1$	—	$n-2$	$3n-3$
Scalar multiplication $(\mathbb{G}_1, \mathbb{G}_2)$	8	4	$n+5$	$n+3$	n	$2n+6$	$3n-1$	—	2	$n+4$
Scalar multiplication $(\mathbb{G}_T)$	$2n$	—	n_i-1	$2n-1$	—	—	—	$\leq (n-1)+k$	$\leq 2+k$	$m-1$
Pairing	4	4	$\leq 4+k$	$2(n+m)-1$	$\leq 1+k$	$n+5$	$\leq 2n-2+k$	$\leq (n-1)+k$	$\leq 4+k$	$m+1$
Exponentiation	—	—	n_i-2	—	—	—	—	—	—	—
Increment/decrement of computation time $n = 100, m = 10, n_i = 10$	↑191%	↑204%	↓89%	↓94%	↓88%	↓89%	↓158%	↓95%	↓92%	—

m: the number of TAs.
n: the number of users.
n_i: the number of users in the ith TA, where $i = 1, 2, \ldots, m$.
$k = 5m\lceil \log_3(m) \rceil$.

A.1. Key-Compromise Impersonation

Case 1. Let $U_1, U_2, \ldots, U_i, \ldots, U_t, \ldots$, and U_n be legal users who are going to negotiate a session conference key. Assume that the private key S_i of U_i is compromised. Thus, an adversary can act as U_i and impersonate another user U_t. The attack holds even if U_i has no U_t's long-term private key S_t. U_i picks two random integers $a_i, a_t \in \mathbb{Z}_q^*$ as her/his ephemeral private keys and then computes $T_{i,j} = a_i Q_j$ $(1 \le j \le n, j \ne i)$ and $T_{t,\hat{j}} = a_t Q_{\hat{j}}$ $(1 \le \hat{j} \le n, \hat{j} \ne t)$. The session conference key $K = e(P,P)^{(a_1+a_2+\cdots+a_i+\cdots+a_t+\cdots+a_n)}$ and U_t is impersonated by U_i in this session [26] and even showed that U_i can impersonate U_t without participating in session key negotiation.

Case 2. Suppose that an adversary $\mathscr{A}$ obtains U_i's and U_t's long-term private keys S_i and S_t, where $i < t$. $\mathscr{A}$ can impersonate the other users $U_1, U_2, \ldots, U_{i-1}, U_{i+1}, \ldots, U_t, \ldots$, and U_n to beguile U_i into negotiating a session conference key. $\mathscr{A}$ randomly picks integers $a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{Z}_q^*$ and computes $T_{j,i} = a_j Q_i$ $(1 \le j \le n, j \ne i)$. U_i randomly picks an integer $a_i \in \mathbb{Z}_q^*$ and computes $T_{i,j} = a_i Q_j$ $(1 \le j \le n, j \ne i)$. Following the protocol, U_i can compute the session conference key $K_i = e(a_i Q_i + \sum_{j=1,j\ne i}^{n} T_{j,i}, S_i) = e(P,P)^z$, where $z = a_1 + a_2 + \cdots + a_i + \cdots + a_n$. According to the message $T_{i,t} = a_i Q_t$, $\mathscr{A}$ computes $e(T_{i,t}, S_t) = e(a_i(I_t s_1 + s_2)P, (I_t s_1 + s_2)^{-1}P) = e(P,P)^{a_i}$. Hence, $\mathscr{A}$ can compute the session conference key $K = K_E = e(P,P)^{a_i} \cdot e(P,P)^{a_1+a_2+\cdots+a_{i-1}+a_{i+1}+\cdots+a_n} = K_i$.

A.2. Forward Secrecy. Suppose that an adversary $\mathscr{A}$ gets user U_i's long-term private key S_i. $\mathscr{A}$ can compute $e(\sum_{j=1,j\ne i}^{n} T_{j,i}, S_i) = e(P,P)^{\sum_{j=1,j\ne i}^{n} a_j}$ but cannot compute $e(P,P)^{a_i}$. If two or more users' long-term private keys are compromised, $\mathscr{A}$ can compute the previous session keys. Assume that user U_t's long-term private key S_t is also obtained by $\mathscr{A}$. $\mathscr{A}$ can compute $e(T_{i,t}, S_t) = e(a_i(I_t s_1 + s_2)P, (I_t s_1 + s_2)^{-1}P) = e(P,P)^{a_i}$ and derive the session key $e(P,P)^{a_1+a_2+\cdots+a_n}$. Therefore, this scheme offers partial forward secrecy only.

B. Du et al.'s Scheme [15, 16]

Let $\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_n$ be the identifiers of the users $U_1, U_2, \ldots, U_n$. Each U_i $(1 \le i \le n)$ has the public key $Q_i = H_1(\mathrm{ID}_i)$ and the private key $S_i = sQ_i$, where s is the master key of KGC. The protocol is described below.

Step 1. U_i randomly chooses N_i, computes $z_i = N_i P$ and $T_i = H(z_i)S_i + N_i P_{\text{pub}}$, where $P_{\text{pub}} = sP$, and then broadcasts (z_i, T_i).

Step2. If $e(\sum_{j=1,j\ne i}^{n} T_j, P) = e(\sum_{j=1,j\ne i}^{n}(H(z_j)Q_j + z_j), P_{\text{pub}})$, U_i broadcasts $X_i = e(P_{\text{pub}}, N_i(z_{i+1} - z_{i-1}))$. Besides, let $z_{n+1} = z_1$ and $z_0 = z_n$.

Step 3. Finally, U_i computes the session key $K = e(P_{\text{pub}}, nN_i z_{i-1}) \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_n^{i-1} \cdot X_1^{i-2} \cdot X_2^{i-3} \cdots X_{i-2} = e(P,P)^{(N_1N_2+N_2N_3+\cdots+N_{n-1}N_n+N_nN_1)s}$.

The following two attacks are valid.

B.1. Key-Compromise Impersonation [29]. When U_i wants to negotiate a session conference key with others, she/he must compute the authenticated factor (z_i, T_i) and broadcast it. Two adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$ can get (z_i, T_i) in a previous session. Suppose that $\mathscr{A}_1$ is U_{i-1} and $\mathscr{A}_2$ is U_{i+1}. Thus, $\mathscr{A}_1$ and $\mathscr{A}_2$ can impersonate U_i by rebroadcasting (z_i, T_i) that satisfies the verification formula in Step 2. When $\mathscr{A}_1$ and $\mathscr{A}_2$ collude, they can compute $X_i = e(z_i, (N_{i+1} - N_{i-1})P_{\text{pub}}) = e(N_i P, s(z_{i+1} - z_{i-1})) = e(P_{\text{pub}}, N_i(z_{i+1} - z_{i-1}))$.

Now, $\mathscr{A}_1$ and $\mathscr{A}_2$ have a valid message (z_i, T_i, X_i), so that they can impersonate U_i to construct a session conference key without being detected by other users.

B.2. Forward Secrecy. Assume that an adversary $\mathscr{A}$ gets TA's long-term private key s. $U_1, U_2, \ldots$, and U_n negotiated a key K in a previous session and $\mathscr{A}$ got all transmitted messages $(z_1, T_1, X_1), (z_2, T_2, X_2), \ldots$, and (z_n, T_n, X_n). Then $\mathscr{A}$ can compute the previous session key K as follows:

$$\begin{aligned} K &= e(z_i, snz_{i-1}) \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \\ &= e(P_{\text{pub}}, nN_i z_{i-1}) \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2}. \end{aligned} \tag{B.1}$$

Du et al. improved their scheme in [16]. They added a synchronous counter c which was held by all users. The initial value of c is 1 and c is increased by 1 after a successful session. In the improved scheme, they modified $T_i = H(z_i) cS_i + N_i P_{\text{pub}}$ in Step 1 and U_i verified if $e(\sum_{j=1,j\ne i}^{N} T_j, P) = e(\sum_{j=1,j\ne i}^{N}(H(z_j)cQ_j + z_j), P_{\text{pub}})$ in Step 2. However, [16] still has the same vulnerability as [15] to the key-compromise impersonation attack (because $\mathscr{A}_1$ and $\mathscr{A}_2$ who act as U_{i-1} and U_{i+1}, resp., know c, which is public among users) and lacks forward secrecy as shown above.

C. Zhang et al.'s Scheme [18]

Each U_i $(1 \le i \le n)$ has the public key $Q_i = H_1(\mathrm{ID}_i)$ and the private key $S_i = sQ_i$, which are the same as those of Du et al.'s scheme. The protocol is shown below.

Step 1. U_i randomly chooses N_i and computes $z_i = N_i P$ and $T_i = H(z_i)S_i + N_i P_{\text{pub}}$ and then sends z_i with the signature T_i to $U_{i-2}, U_{i-1}, U_{i+1}$, and U_{i+2}.

Step 2. After verifying each received z_i by checking that $e(T_i, P) = e((H(z_i)Q_i + z_i), P_{\text{pub}})$, U_1 and U_n compute $Y_{1R} = e(z_2, z_3)^{N_1}$, $Y_{1L} = e(z_n, z_2)^{N_1} = e(z_1, z_2)^{N_n} = Y_{nR}$, $Y_{nL} = e(z_{n-2}, z_{n-1})^{N_n}$ if n is odd; otherwise U_1 and U_{n-1} compute $Y_{1R} = e(z_2, z_3)^{N_1}$, $Y_{1L} = e(z_{n-1}, z_n)^{N_1} = e(z_n, z_1)^{N_{n-1}} = Y_{(n-1)R}$, $Y_{(n-1)L} = e(z_{n-3}, z_{n-2})^{N_{n-1}}$. The other U_i's compute $Y_{iL} = e(z_{i-2}, z_{i-1})^{N_i}$ and $Y_{iR} = e(z_{i+1}, z_{i+2})^{N_i}$ if i is odd; otherwise they do nothing. Finally, each U_i, with an odd i, broadcasts $X_i = H(Y_{iL}) \oplus H(Y_{iR})$ with the signature.

Step 3. Each user U_i (i is odd) and U_{i+1} can compute the session conference key $K = H(Y_{1L}) + H(Y_{3L}) + \cdots + H(Y_{nL})$ if n is odd where $H(Y_{jL}) = H(Y_{iR}) \oplus X_i \oplus X_{i-2} \oplus \cdots \oplus X_j$

(when $i \geq j$) or $H(Y_{jL}) = H(Y_{iR}) \oplus X_i \oplus X_{i-2} \oplus \cdots \oplus X_1 \oplus X_n \oplus X_{n-2} \oplus \cdots \oplus X_j$ (when $i < j$) for each odd j. Otherwise, $K = H(Y_{1L}) + H(Y_{3L}) + \cdots + H(Y_{(n-1)L})$ if n is even where $H(Y_{jL}) = H(Y_{iR}) \oplus X_i \oplus X_{i-2} \oplus \cdots \oplus X_j$ (when $i \geq j$) or $H(Y_{jL}) = H(Y_{iR}) \oplus X_i \oplus X_{i-2} \oplus \cdots \oplus X_1 \oplus X_{n-1} \oplus X_{n-3} \oplus \cdots \oplus X_j$ (when $i < j$) for each odd j.

It is easy to impersonate a user whose index is even in the scheme since the users only sends the message (z_i, T_i). We show an example as follows. Suppose that there are six users U_1, U_2, U_3, U_4, U_5, and U_6 who want to negotiate a conference key. By this scheme, we can know that U_4 only submits (z_4, T_4) to U_2, U_3, U_5, and U_6 in Step 1. If U_2, U_3, U_5, and U_6 collude, they can negotiate a randomly-chosen pair of (z_4', T_4') to cheat U_1 without U_4 joining in the session because U_1 does not need to verify (z_4', T_4'). Besides, even though U_i broadcasts the message (z_i, T_i) and every user can check the correctness of the message, the scheme still suffers from key-compromise impersonation by resending (z_i, T_i) as that of Du et al.'s scheme [29].

D. Kim et al.'s Scheme [17]

First, KGC sets up s as the master key and (P_{pub}, P) as public parameters, where $P_{\text{pub}} = sP$. Then, KGC generates a key pair for each user. Let $\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n$ be the identifiers of users $U_1, U_2, \ldots, U_n$, respectively. Each U_i $(1 \leq i \leq n)$ has the public key $Q_i = H_1(\text{ID}_i)$ and the private key $S_i = sQ_i$.

Step 1. U_i broadcasts $a_i P_{\text{pub}}$ with the signature (P_i, T_i), where $P_i = aP$, $T_i = H(P_i, a_i P_{\text{pub}}) S_i + a a_i P_{\text{pub}}$, a_i and a are randomly chosen by U_i.

Step 2. After verifying the signature by checking that $e(T_j, P) = e(H(P_j, a_j P_{\text{pub}}) Q_j, P_{\text{pub}}) \cdot e(a_j P_{\text{pub}}, P_j)$, the conference key K_s is computed as follows: $K_s = H_2(K)$ where $K = \prod_{i=1}^{N} e(Q_i, a_i P_{\text{pub}})$.

This scheme has a serious problem. In the protocol, since every user U_i broadcasts the message $(a_i P_{\text{pub}}, P_i, T_i)$, one can collect all $a_i P_{\text{pub}}$'s and compute all Q_i's to derive $K = \prod_{i=1}^{N} e(Q_i, a_i P_{\text{pub}})$ even if she/he does not join the session.

E. Choi et al.'s Scheme [24]

First, KGC sets up s as the master key and (P_{pub}, P) as public parameters, where $P_{\text{pub}} = sP$. Then, KGC generates a key pair for each user. Let $\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n$ be the identifiers of users $U_1, U_2, \ldots, U_n$, respectively. Each U_i $(1 \leq i \leq n)$ has the public key $Q_i = H_1(\text{ID}_i)$ and the private key $S_i = sQ_i$.

Step 1. U_i broadcasts the signature (P_i, T_i), where $P_i = a_i P$, $T_i = H(P_i) S_i + a_i P_{\text{pub}}$, a_i is randomly chosen by U_i.

Step 2. After receiving (P_{i-1}, T_{i-1}), (P_{i+1}, T_{i+1}), and (P_{i+2}, T_{i+2}), U_i verifies the messages by checking whether $e(T_{i-1} + T_{i+1} + T_{i+2}, P) = e(P_{i-1} + P_{i+1} + P_{i+2} + H(P_{i-1}) Q_{i-1} + H(P_{i+1}) Q_{i+1} + H(P_{i+2}) Q_{i+2}, P_{\text{pub}})$. U_i then broadcasts $D_i = e(a_i(P_{i+2} - P_{i-1}), P_{i+1})$.

Step 3. U_i can compute the session key $K_i = e(a_i P_{i-1}, P_{i+1})^n D_i^{n-1} D_{i+1}^{n-2} \cdots D_n^{i-1} D_1^{i-2} D_2^{i-3} \cdots D_{i-2}$.

Reference [30] has shown that [24] is vulnerable to key-compromise impersonation as follows.

Suppose that attacker $\mathscr{A}_1$ is U_{i-1} and attacker $\mathscr{A}_2$ is U_{i+2}. Thus, $\mathscr{A}_1$ and $\mathscr{A}_2$ can rebroadcast (P_i, T_i) and then broadcast $D_i = e(P_i, P_{i+1})^{-a_{i-1} + a_{i+2}} = e(a_i(P_{i+2} - P_{i-1}), P_{i+1})$ to impersonate U_i.

F. Zhou et al.'s Schemes [26]

Zhou et al. proposed two group key agreement schemes. One is a one-round scheme with security proofs. The other based on the former is a two-round scheme with lower communication cost. These schemes are depicted as below:

KGC sets up s as the master key and (P_{pub}, P) as public parameters, where $P_{\text{pub}} = sP$. Then, KGC generates a key pair for each user. Let $\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n$ be the identifiers of users $U_1, U_2, \ldots, U_n$, respectively. Each U_i $(1 \leq i \leq n)$ has the public key $Q_i = H_1(\text{ID}_i)$ and the private key $S_i = sQ_i$.

In the former scheme, each user U_i randomly picks δ_i, r_i, and k_i, computes $P_i^j = H_2(e(S_i, Q_j) \cdot \delta_i) \oplus r_i$ for each $1 \leq j \leq n$ and $j \neq i$, and broadcasts $D_i = (\delta_i, P_i^1, \ldots, P_i^{i-1}, P_i^{i+1}, \ldots, P_i^n, H_3(r_i) \oplus k_i)$. After receiving all $D_j = (R_j, P_j^1, \ldots, P_j^{j-1}, P_j^{j+1}, \ldots, P_j^n, V_j)$'s, U_i computes $k_j' = H_3(H_2(e(Q_j, S_i) \cdot R_j) \oplus P_j^i) \oplus V_j$ for each $1 \leq j \leq n$ and $j \neq i$. Thus, the session key $K_i = k_1' \oplus k_2' \oplus \cdots \oplus k_{i-1}' \oplus k_i \oplus k_{i+1}' \oplus \cdots \oplus k_n'$.

In the latter scheme, only U_1 randomly picks δ, r, and k_1 and broadcasts $D_1 = (R, P_2, \ldots P_n, V, W)$ such that $D_1 = (\delta, r \oplus H_4(e(S_1, Q_2) \cdot \delta), \ldots, r \oplus H_4(e(S_1, Q_n) \cdot \delta), H_5(r) k_1 P, k_1 P_{\text{pub}})$. When each U_i $(2 \leq i \leq n)$ receives D_1, he computes $r' = H_4(e(Q_1, S_i) \cdot R) \oplus P_i = r$ and broadcasts $D_i = (X_i, Y_i) = (H_5(r) k_i P, k_i P_{\text{pub}})$, where k_i is randomly chosen by U_i. Finally, all U_i's $(1 \leq i \leq n)$ compute $z_1 = H_5(r)^{-1} V$ and $z_j = H_5(r)^{-1} X_j$, $2 \leq j \leq n$, and verify $z_1, \ldots, z_n$ by checking $e(P, \sum_{j=2}^{n} Y_j + W) = e(P_{\text{pub}}, \sum_{j=1}^{n} z_j)$. Finally, all users can compute the session key $K_i = H_6(z_1) \oplus \cdots \oplus H_6(z_n)$.

Both of the schemes only achieve partial forward secrecy. Once all of the private keys S_i's of the users are revealed or KGC's secret key s is corrupted, an attacker is able to compute previous session keys by intercepting the previous broadcast messages. Moreover, the schemes cannot withstand key-compromise impersonation. In the first scheme, if the private keys $S_1, \ldots, S_{i-1}, S_{i+1}, \ldots S_n$ are revealed, an attacker $\mathscr{A}$ can impersonate U_i by broadcasting the message D_i with random strings $\{\delta_i, r_i, k_i\}$ and $P_i^j = H_2(e(S_j, Q_i) \cdot \delta_i) \oplus r_i$. In the second scheme, if U_1's private key S_1 is revealed, an attacker $\mathscr{A}$ can impersonate any other U_i by computing and broadcasting $D_i = (H_5(r) k_i P, k_i P_{\text{pub}})$, $2 \leq i \leq n$, where δ, r, k_1, and k_i are randomly chosen.

G. Yao et al.'s Scheme [25]

KGC sets up s as the master key and (R, P) as public parameters, where $R = sP$. Then, KGC generates a key pair for each user. Let $\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_n$ be the identifiers of users

$U_1, U_2, \ldots, U_n$, respectively. Each U_i ($1 \leq i \leq n$) has the public key $Q_i = H_0(\mathrm{ID}_i)$ and the private key $S_i = sQ_i$.

Step 1. Each U_i generates a random string r_i, computes $E_i = r_iP$ and $F_i = H(U, e(E_i, R))S_i + r_iR$, where $U = \mathrm{ID}_1 \| \mathrm{ID}_2 \| \cdots \| \mathrm{ID}_n$, and broadcasts (E_i, F_i).

Step 2. Each U_i verifies the received (E_j, F_j), $1 \leq j \leq n$ and $j \neq i$, by checking whether $e(\sum_{j \neq i} F_j, P) = e(\sum_{j \neq i} H(U, e(E_j, R))Q_j + E_j, R)$. If true, U_i computes and broadcasts $Y_i = r_iT$ and $X_i = r_i(E_{i+1} - E_{i-1} + T)$, where $T = H_0(\mathrm{ID}_1 \| E_1 \| \cdots \| \mathrm{ID}_n \| E_n)$.

Step 3. After receiving (X_j, Y_j)'s, U_i checks if $e(\sum_{j \neq i} Y_j, P) = e(\sum_{j \neq i} E_j, T)$. Then, U_i computes $Z_i = e(nr_iE_{i-1} + \sum_{j=0}^{n-1}(n-1-j)(X_{i+j} - Y_{i+j}), R)$ and broadcasts $C_i = H(i \| U \| E_1 \| \cdots \| E_n \| X_1 \| \cdots \| X_n \| Y_1 \| \cdots \| Y_n \| Z_i)$.

Step 4. U_i checks whether each received C_j equals $H(j \| U \| E_1 \| \cdots \| E_n \| X_1 \| \cdots \| X_n \| Y_1 \| \cdots \| Y_n \| Z_i)$, where $Z_i = Z_j$. If the condition holds, the session key $K_i = H(U \| E_1 \| \cdots \| E_n \| X_1 \| \cdots \| X_n \| Y_1 \| \cdots \| Y_n \| Z_i \| C_1 \| \cdots \| C_n)$.

The scheme is not immune to key-compromise impersonation, either. Suppose that an adversary $\mathscr{A}$ who obtains U_i's private key S_i attempts to impersonate U_t. $\mathscr{A}$ can rebroadcast previous messages (E_t', F_t') and (X_t', Y_t') in Step 1 and Step 2, respectively. Then, $\mathscr{A}$ broadcasts $C_t = H(t \| U \| E_1 \| \cdots \| E_n \| X_1 \| \cdots \| X_n \| Y_1 \| \cdots \| Y_n \| Z_i)$ in Step 3.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was partially supported by the Ministry of Science and Technology of Taiwan under Grant MOST 103-2221-E-110-057, NSYSU-KMU Joint Research Project (NSYSUKMU 103-I001), and Aim for the Top University Plan of the National Sun Yat-sen University and Ministry of Education, Taiwan.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] R. Dutta and R. Barua, "Overview of key agreement protocols," Tech. Rep. 2005/289, Cryptology ePrint Archive, 2005, http://eprint.iacr.org/2005/289.pdf.

[3] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advance in Cryptology*, vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, Berlin, Germany, 1985.

[4] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[5] G. Frey, M. Müller, and H. Rüuck, "The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1717–1719, 1999.

[6] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *Journal of Cryptology*, vol. 17, no. 4, pp. 263–276, 2004.

[7] N. P. Smart, "Identity-based authenticated key agreement protocol based on Weil pairing," *Electronics Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[8] L. Chen, Z. Cheng, and N. P. Smart, "Identity-Based Key Agreement Protocols from Pairings," Cryptology, Report 2006/199, 2006, http://eprint.iacr.org/2006/199.pdf.

[9] L. Chen and C. Kudla, "Identity based key agreement protocols from pairings," in *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pp. 219–233, Pacific Grove, Calif, USA, 2003.

[10] Y. J. Choie, E. Jeong, and E. Lee, "Efficient identity-based authenticated key agreement protocol from pairings," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 179–188, 2005.

[11] N. McCullagh and P. S. L. M. Barreto, "A new two-party identity-based authenticated key agreement," in *Topics in Cryptology—CT-RSA 2005*, vol. 3376 of *Lecture Notes in Computer Science*, pp. 262–274, Springer, Berlin, Germany, 2005.

[12] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptology-CRYPTO 93*, vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Springer, 1994.

[13] S. S. Al-Riyami and K. G. Paterson, "Tripartite authenticated key agreement protocols from pairings," in *Cryptography and Coding*, vol. 2898 of *Lecture Notes in Computer Science*, pp. 332–359, Springer, 2003.

[14] K. Shim and S. S. Woo, "Cryptanalysis of tripartite and multi-party authenticated key agreement protocols," *Information Sciences*, vol. 177, no. 4, pp. 1143–1151, 2007.

[15] X. Du, Y. Wang, J. Ge, and Y. Wang, "An ID-based authenticated two round multi-party key agreement," Cryptology ePrint Archive, Report 2003/247, 2003, http://eprint.iacr.org/2003/247.pdf.

[16] X. Du, Y. Wang, J. Ge, and Y. Wang, "An improved ID-based authenticated group key agreement scheme," Cryptology ePrint Archive, Report 2003/260, 2003, http://eprint.iacr.org/2003/260.pdf.

[17] J. S. Kim, H. C. Kim, K. J. Ha, and K. Y. Yoo, "One round identity-based authenticated conference key agreement protocol," in *Proceedings of the 3rd European Conference (ECUMN '04)*, vol. 3262 of *Lecture Notes in Computer Science*, pp. 407–416, 2004.

[18] P. Zhang, C. Ye, X. Li, and X. Ma, "An efficient keys agreement for multi-party's communication," in *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '05)*, pp. 267–269, Dalian, China, December 2005.

[19] Y. Shi, G. Chen, and J. Li, "ID-based one round authenticated group key agreement protocol with bilinear pairings," in *Proceeding of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, vol. 1, pp. 757–761, April 2005.

[20] I. F. Blake, G. Seroussi, and N. P. Smart, *Advances in Elliptic Curve Cryptography*, London Mathematical Society Lecture Note Series, No. 317, 2005.

[21] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Infromation and Communications Security*, vol. 2836 of *Lecture Notes in Computer Science*, pp. 301–312, 2003.

[22] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology-CRYPTO 04*, vol. 3152 of *Lecture Notes in Computer Science*, pp. 41–55, Springer, 2004.

[23] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Advances in Cryptology—CRYPTO 2006*, vol. 4117 of *Lecture Notes in Computer Science*, pp. 209–307, Springer, Berlin, Germany, 2006.

[24] K. Y. Choi, J. Y. Hwang, and D. H. Lee, "Efficient ID-based group key agreement with bilinear maps," in *Public Key Cryptography—PKC 2004*, vol. 2947 of *Lecture Notes in Computer Science*, pp. 130–144, Springer, Berlin, Germany, 2004.

[25] G. Yao, H. Wang, and Q. Jiang, "An authenticated 3-round identity-based group key agreement protocol," in *Proceedings of the 3rd International Conference on Availability, Security, and Reliability (ARES '08)*, pp. 538–543, Barcelona, Spain, March 2008.

[26] L. Zhou, W. Susilo, and Y. Mu, "Efficient ID-based authenticated group key agreement from bilinear pairings," in *Proceedings of the 2nd International Conference (MSN '06)*, vol. 4325 of *Lecture Notes in Computer Science*, pp. 521–532, 2006.

[27] W. Yuan, L. Hu, H. Li, and J. Chu, "An enhanced authenticated 3-round identity-based group key agreement protocol," in *Green Communications and Networks*, vol. 113 of *Lecture Notes in Electrical Engineering*, pp. 549–556, Springer, Amsterdam, The Netherlands, 2012.

[28] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Designs, Codes and Cryptography*, vol. 19, no. 2-3, pp. 173–193, 2000.

[29] F. G. Zhang and X. F. Chen, "Attack on Two ID-based Authenticated Group Key Agreement Schemes," Cryptology, Report 2003/259.

[30] F. Zhang and X. Chen, "Attack on an ID-based authenticated group key agreement scheme from PKC 2004," *Information Processing Letters*, vol. 91, no. 4, pp. 191–193, 2004.

[31] H. Park, T. Asano, and K. Kim, "Improved ID-based authenticated Group key agreement secure against impersonation attack by insider," in *Proceedings of the Symposium on Cryptography and Information Security*, Otsu, Japan, 2009.

[32] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology—EUROCRYPT '97*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 256–266, 1997.

[33] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73, November 1993.

[34] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in *Crytography and Coding: Proceedings of the 6th IMA International Conference on Cryptography and Coding*, vol. 1355 of *Lecture Notes in Computer Science*, pp. 30–45, 1997.

[35] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *Lecture Notes in Computer Science*, pp. 387–398, Springer, Berlin, Germany, 1996.

[36] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[37] M. Bellare, J. A. Garary, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Advances in Cryptology—EUROCRYPT'98*, vol. 1403 of *Lecture Notes in Computer Science*, pp. 236–250, Springer, Berlin, Germany, 1998.

[38] R. Barua, R. Dutta, and P. Sarkar, "Extending Joux's protocol to multi party key agreement," in *Progress in Cryptology-INDOCRYPT 03*, vol. 2904 of *Lecture Notes in Computer Science*, pp. 205–217, 2003.