IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2019, accepted October 8, 2019, date of publication October 10, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946874

Key Escrow Protocol Based on a Tripartite Authenticated Key Agreement and Threshold Cryptography

ZHEN WANG, ZHAOFENG MA, SHOUSHAN LUO, AND HONGMIN GAO
School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Zhaofeng Ma (mzf@bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61272519, and in part by the Research Funds of Blockchain and Security Joint Lab of BUPT & BCT and BUPT & CAPSTONE under Grant B2018005 and Grant S2017060.

ABSTRACT While instant messaging systems bring convenience to people's lives and work, they also make it easier for malicious users to discuss and plot illegal activities. Therefore, determining how to balance the privacy protection requirements of user communication in the network with the authorized monitoring requirements of law enforcement agencies (LEAs) is a meaningful task. To solve this problem, a new tripartite authenticated key agreement (Tri-AKA) protocol and a session key escrow scheme based on threshold cryptography and the new Tri-AKA protocol were proposed. In the proposed scheme, the LEA participates as a normal user in the key agreement process of two users and uses (t, n) threshold cryptography to share its ephemeral private key with n key escrow agents (KEAs). When necessary, the LEA can combine t KEAs to recover the specified session key and decrypt the communications, thereby preventing malicious administrators in the LEA from arbitrarily monitoring user communications. Finally, we proved the security of the proposed Tri-AKA protocol under the Computational Diffie-Hellman (CDH) assumption with the Random Oracle Model and the security of the proposed key escrow scheme under the Elliptic Curve Discrete Logarithm (ECDL) assumption. Analysis of our session key escrow scheme and comparison with other schemes show that our scheme can avoid the ''once monitor, monitor forever'' scenario and achieve fine-grained control in each session. Moreover, our scheme has low storage overhead for each KEA.

INDEX TERMS Instant messaging, authorized monitoring, key escrow, threshold cryptography, tripartite authenticated key agreement.

I. INTRODUCTION

With the popularity of mobile intelligent devices and the rapid development of mobile Internet, IM (instant messaging) apps have become a very frequently used tool in people's daily lives and work. IM systems integrate short message communication, voice and video calls, document transmission, and even payment functions, which make up for the shortcomings of traditional communication (such as E-mail and voice communication). Utilizing an IM system, people can not only communicate with friends conveniently but also talk about professional work with coworkers or even conduct business with their commercial partners [1]. Professional and business contacts may include some commercial secrets in their communication through an IM system. Meanwhile, it is easy for attackers on the Internet to eavesdrop or intercept plaintext. To protect against this threat, IM systems usually scramble plaintext communications with an encryption algorithm. Only the users who have the correct secret key can decrypt the scrambled data and then obtain the original content.

As shown in Fig. 1, there are normally two types of architecture for an encrypted IM system. Suppose that there are two users (user A, the message sender, and user B, as the recipient) and an IM server in the system, that message M is a plaintext communication, and that functions Enc() and Dec() are encryption and decryption algorithms in symmetric cryptography. To protect message M, if the system as is of

The associate editor coordinating the review of this manuscript and approving it for publication was Remigiusz Wisniewski.

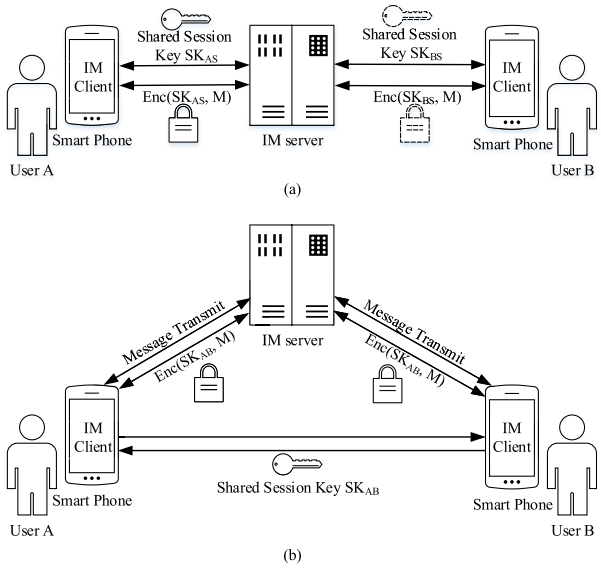


FIGURE 1. Two types of architecture for the encrypted IM system.

type (a) in Fig. 1, the IM server will obtain two different secret session keys, marked SK_{AS} and SK_{BS}, for users A and B, respectively. Although this procedure can prevent attackers from eavesdropping on messages, it is not sufficiently safe for some users because the IM server can obtain plaintext message M by evaluating Dec(SK_{AS}, Enc(SK_{AS}, M)). For a system of type (b) in Fig. 1, the message sender will negotiate the secret session key (marked as SK_{AB}) only with the specified recipient. The IM server in this type of system cannot do anything but transmit the encrypted data Enc(SK_{AB}, M) accurately [2]. There is no doubt that architecture of type (b) is sufficiently safe for the most fastidious users. However, it should be noted that sometimes there are criminal or even terrorist activities that utilize the IM system. Therefore, it is necessary for the government to wiretap communications at the appropriate time to find and prevent these illegal activities [3], [4]. Obviously, it is hard for authorized law enforcement agencies (LEAs) to wiretap communications in IM systems of type (b). Therefore, determining how to balance the requirements between user privacy protection and the need for government-authorized monitoring is an important branch in cryptography.

A. RELATED WORKS

In this paper, whose purpose is to balance users' communication security and government-authorized supervision, a key escrow protocol for mobile instant messaging systems based on tripartite key agreement and threshold cryptography was proposed that was developed using difficult calculation problems on elliptic curves. Related studies are described as follows.

1) KEY ESCROW SCHEME

Key escrow scheme is a technology that can acquire decrypted information in an emergency. One of the purposes of a key escrow scheme is to monitor or audit encrypted data with authorization; the other is to recover the lost private key of a user [3]–[5]. Here, we only discuss how to monitor or audit encrypted data with authorization.

In 1993, the US government proposed a concept of key escrow that is based on a special encryption chip [6]. With this scheme, the US government can decrypt all communications encrypted by the special chip. Since then, various key escrow schemes have been proposed by different researchers. Azfar A used (t, n) threshold cryptography to escrow a session master key with multiple agents in a VoIP system [7] by dividing the session master key into n parts and escrowing the n parts to n escrow agents; at least t escrow agents must hand over their hosting parts to enable reconstruction the session master key, which reduces the risk that the session master key will be compromised if a single escrow agent is attacked. Long Y et al. proposed a dynamic threshold key escrow scheme that escrows the user's private key based on conic [3], which allows the system to add or remove a key escrow agent without changing the secret shares of other key escrow agents. Moreover, the scheme avoids the "once monitor, monitor forever" scenario by updating the user's private key periodically. Fan Q et al. designed a key escrow program with a cooperation mechanism for multi-group escrow agents based on threshold cryptography [8]. Furthermore, they also proposed a special key escrow scheme, which allows an important escrow agent to have the right of denial [9]; other escrow agents cannot reconstruct the secret key without the participation of the important escrow agent. Since PKI cryptography suffers from issues of certificate management and withdrawal, Shamir proposed identity-based asymmetric cryptosystems [10], which have the inner property of key escrowing. Therefore, Wang S B, Gao Z G and Ni L et al. proposed different identity-based key escrow schemes based on different mathematical problems and security models [11]–[14]. All of these schemes escrowed the master key of the system.

In summary, the above key escrow schemes, which escrow the master key of the system, must ensure that the key escrow agent is absolutely safe and trustworthy; otherwise, a malicious administrator can freely wiretap the communication data of any user in the system without authorization, which addressed some researchers proposed key agreement protocols that avoid key escrow [11], [12], [15], [16] to protect user privacy. Other schemes that escrow users' private key or session master key will consume a lot of storage resources and computing resources as the number of users in the system increases.

2) SECRET SHARING SCHEME

Shamir and Blakley almost simultaneously proposed their secret sharing schemes with different methods in 1979 [17], [18]. Shamir's scheme used Lagrange interpolation, while Blakley's scheme was based on the intersection of multiple multidimensional spaces. Subsequently, researchers have designed secret sharing schemes based on the Chinese Remainder Theorem (CRT) [19], [20] and Attribute-based

Encryption (ABE) [21]. The above schemes are also referred to as (t, n) threshold cryptography. Eslami Z et al. proposed ideal social secret sharing using Birkhoff interpolation [22], which allows participants to have different authorizations and allows the number of participants can change dynamically. Classified by secret type, secret sharing schemes can be divided into schemes for number/text and schemes for images [23]–[25]. In addition, there are researchers who study linguistic techniques for cryptographic data sharing algorithms, with which the secret information will be divided into secret information and linguistic information, which will then be escrowed to two different group participants [26]. In traditional secret sharing schemes, it is impossible to verify the correctness of the hosting parts handed over by participants when reconstructing the secret. If there are malicious participants in the system, the sharing secret may not be reconstructed correctly, and the malicious participants cannot be identified. To solve this problem, some verifiable secret sharing schemes have been proposed by different researchers [27]. In addition to key escrow, secret sharing schemes can also be used for group session key sharing and cloud data sharing or computing [27]–[29]. Data sharing in cloud computing is primarily implemented by the ABE cryptosystem. Users who satisfy certain conditions can decrypt the secret data and obtain the original plaintext data.

3) TRIPARTITE KEY AGREEMENT PROTOCOL

The key agreement (KA) protocol is one of the most important protocols in Internet communication: it ensures that protocol participants negotiate a common session key in an open and insecure channel. Depending on the number of participants, the key agreement protocol is classified as a bipartite key agreement (Bi-KA) protocol, a tripartite key agreement (Tri-KA) protocol or a group key agreement (G-KA) protocol. Using the Weil and Tate pairings on elliptic curves in cryptography, Joux A proposed an efficient tripartite key agreement protocol with one round of communication [30] that lacked authentication for the identity of the participants. Xiong H et al. proposed two different authenticated tripartite key agreement protocols based on certificateless public key cryptography [31] and identity-based public key cryptography [32] and also proved the security of these protocols in the random oracle model. With the digital signature algorithm, Tan Z also proposed an identity-based tripartite authenticated key agreement (ID-AKA) protocol [33]. To protect against the leakage of participants' ephemeral secret keys, Manulis M et al. proposed a security model for a group key agreement protocol based on the extended Canetti-Krawczyk (eCK) model, named the g-eCK model [34], and then proposed a tripartite key agreement protocol (treated as a special case of group key agreement protocol) under the g-eCK model. Bayat M et al. proposed the first attribute-based tripartite key agreement protocol and an improved security model for this kind of protocol [35]. To improve the security of the tripartite AKA protocol, Suzuki K et al. proposed a one-round tripartite AKA protocol in the standard model [36]. To avoid denial-of-service attacks, Gupta D S et al. proposed an identity-based tripartite key agreement protocol with timestamp [37]. Utilizing the tripartite authenticated key agreement (Tri-AKA) protocol, Chien H Y et al. proposed a protocol that enables a pair of registered clients to establish a session key with the help of a trusted server [38].

B. OUR CONTRIBUTION

The contributions of this paper as follows.

(1) For the tripartite key agreement protocol, a security model which supports the adversary to obtain the user's ephemeral private key was presented.
(2) To reduce the computational overhead, a new tripartite authenticated key agreement protocol without pairing operation was proposed.
(3) A session key escrow scheme based on threshold cryptography and the new Tri-AKA protocol was proposed, which fully considers the security of authorized monitoring and session key.
(4) The proposed new key escrow scheme has low storage overhead, as the LEA uses (t, n) threshold cryptography to share its ephemeral private key LEA may generate only one ephemeral private key over the whole life of the IM system.
(5) We proved the security of the proposed Tri-AKA protocol under the Computational Diffie-Hellman (CDH) assumption with the Random Oracle Model.
(6) We conducted some experiments on a laptop and a smart phone based on Java code, and compared our scheme with others comprehensively.
(7) The proposed new key escrow scheme can avoid the ''once monitor, monitor forever'' scenario and achieve fine-grained control in each session.

C. ROADMAP OF THIS PAPER

The rest of this paper is as follows. Section II described the preliminaries and security model used in this paper. Section III presented the architecture and detailed processing at each phase for the proposed session key escrow protocol. Section IV gave the security proof for the proposed Tri-AKA protocol and session key escrow protocol. Section V described the experiments result and compared the proposed scheme with others in terms of security properties, performance and so on. Section VI gave the conclusions of this paper.

II. PRELIMINARIES

A. ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM (ECDL)

An elliptic curve E defined over a field F_p is a set of points $P = (x_n, y_n)$ where x_n and y_n are elements of F_p that satisfy a certain equation; the curve is also denoted $E(F_p)$.

Given two points on $E(F_p)$ and $Q = n \cdot P$, where n is unknown and $n \in Z_q^*$, it is difficult to calculate n with polynomial-time algorithms.

B. COMPUTATIONAL DIFFIE-HELLMAN PROBLEM (CDH)

Given three points P, $m \cdot P$ and $n \cdot P$, where m and n are unknown and $m, n \in Z_q^*$, it is difficult to calculate $m \cdot n \cdot P$ with polynomial-time algorithms.

C. BILINEAR PAIRING

Suppose that G_1 and G_T are cyclic groups of prime order q. Suppose that P is a generator of G_1. A bilinear pairing is a function $e : G_1 \times G_1 \rightarrow G_T$ that satisfies the following properties:

(1) Bilinearity: For any $a, b \in Z_q^*$, $e(a \cdot P, b \cdot P) = e(P, P)^{a \cdot b}$.
(2) Nondegeneracy: $e(P, P) \neq 1$.
(3) Computability: The function e is a polynomial-time algorithm.

D. LAGRANGE INTERPOLATION POLYNOMIAL

Given a polynomial of degree n in Fp:

$$Pn(x) = a_0 + a_1 x + \cdots + a_n x^n, \quad a_i \in Fp, \quad (i = 0, 1, 2, \cdots, n), \tag{1}$$

There are $n + 1$ different points (x_0, y_0), $(x_1, y_1), \cdots, (x_n, y_n)$ that satisfy the equation:

$$y_i = Pn(x_i), \quad (i = 0, 1, \cdots, n) \tag{2}$$

With the $n + 1$ different points (x_0, y_0), $(x_1, y_1), \cdots, (x_n, y_n)$, the Lagrange interpolation polynomial $Ln(x)$ is:

$$Ln(x) = \sum_{j=0}^{n} l_j(x) y_j \tag{3}$$

where $l_j(x)$ is:

$$l_j(x) = \frac{(x - x_0) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)} = \prod_{i=0, i \neq j}^{n} \frac{x - x_i}{x_j - x_i} \tag{4}$$

That is, $L_n(x)$ can be expressed as:

$$Ln(x) = \sum_{j=0}^{n} l_j(x) y_j = \sum_{j=0}^{n} \left(\prod_{i=0, i \neq j}^{n} \frac{x - x_i}{x_j - x_i} \right) y_j \tag{5}$$

Moreover, $P_n(x) = L_n(x)$, which means for an unknown polynomial $P_n(x)$ with degree n, using $n + 1$ different points on the graph of polynomial $P_n(x)$ and equation (5) enables the reconstruction of the polynomial $P_n(x)$.

E. SECURITY MODEL FOR TRIPARTITE KEY AGREEMENT PROTOCOL

Motivated by the g-eCK model [34] of Manulis et al., the G-CK$^+$ model [36] of Suzuki et al. and the 3-IDAKA model [33] of Tan, a security model for the tripartite key agreement (Tri-KA) protocol is proposed in this paper. The proposed security model not only allows the adversary to obtain the user's private key but also allows the adversary to obtain the user's ephemeral private key. In this security model, any session instance in the execution of the protocol is treated as an oracle $\prod$. The oracle $\prod_{i,j,k}^{s}$ represents the s-th session instance among participants U_i, U_j and U_k, where U_i is the initiator of this instance. An adversary $\mathcal{A}$ controls all the communication channels among those participants. In addition, the adversary can randomly execute the queries Send, Reveal, EphemeralKeyReveal and Corrupt.

Send ($\prod_{i,j,k}^{s}$,*M*): M is the message that is sent by the adversary $\mathcal{A}$ to oracle $\prod_{i,j,k}^{s}$. Meanwhile, the adversary $\mathcal{A}$ obtains a feedback result from oracle $\prod_{i,j,k}^{s}$. During the execution of the protocol, since the adversary $\mathcal{A}$ completely controls the communication network, it can eavesdrop, cancel or modify messages sent by other protocol participants, or impersonate other participants to create a message.

Reveal ($\prod_{i,j,k}^{s}$, *i*): The adversary $\mathcal{A}$ queries the oracle $\prod_{i,j,k}^{s}$ and obtains the s-th session key of participant U_i in order to participate in and complete the agreement.

EphemeralKeyReveal ($\prod_{i,j,k}^{s}$, *i*): The adversary $\mathcal{A}$ queries and obtains the ephemeral private key of participant U_i in key agreement instance $\prod_{i,j,k}^{s}$.

Corrupt (i): The adversary $\mathcal{A}$ queries and obtains the private key of protocol participant U_i.

The game in the security model is divided into two phases. In the first phase of the game, the adversary $\mathcal{A}$ can perform the above queries in any order. Once adversary $\mathcal{A}$ determines that the first phase is over, it starts the second phase of the game, then selects a fresh oracle $\prod_{i,j,k}^{s}$ and executes a Test ($\prod_{i,j,k}^{s}$) query to the fresh oracle $\prod_{i,j,k}^{s}$.

The following are definitions of the fresh oracle and matching sessions.

Definition 1 (Fresh Oracle): If an oracle $\prod_{i,j,k}^{s}$ meets all of the following conditions, then it is a fresh oracle:

(1) The oracle $\prod_{i,j,k}^{s}$ has not been queried by Reveal ().
(2) For protocol participant U_i, U_j or U_k, the Corrupt () query and EphemeralKeyReveal () query have not been performed simultaneously.
(3) If the oracles $\prod_{i,j,k}^{s}$ and $\prod_{i,j,k}^{t}$ exist in matching sessions, then the oracle $\prod_{i,j,k}^{t}$ has not been queried by Reveal ().

Definition 2 (Matching Session): When a tripartite key agreement protocol instance is activated, the protocol participant with this instance will be assigned a unique session identifier *sid*. For any given protocol participant, the model does not allow the same session identifier to be assigned in different protocol instances. The sessions holding the same session identifier are called matching sessions.

Test ($\prod_{i,j,k}^{s}$): For a fresh oracle $\prod_{i,j,k}^{s}$, depending on the result b of a random coin flip, the Test query feeds back a real session key if $b = 1$ or gives a randomly generated session key based on the system definition if $b = 0$. The adversary $\mathcal{A}$ can terminate the first phase of the game and perform a Test query at any time and can only perform one Test query for a given oracle.

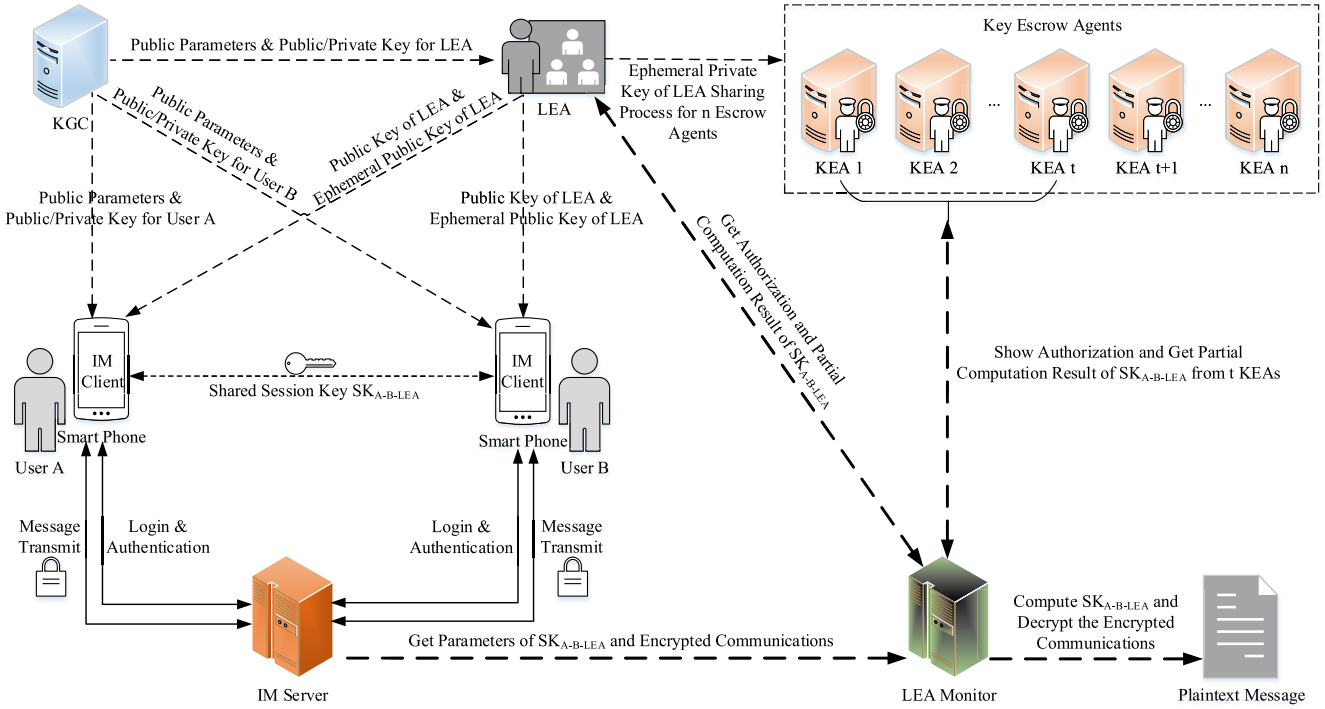


FIGURE 2. Architecture of the proposed session key escrow protocol.

After the Test query is finished, the adversary $\mathcal{A}$ can continue to perform other queries on the oracle $\prod_{i,j,k}^{s}$, but cannot perform the Reveal query on the oracle $\prod_{i,j,k}^{s}$ or on an oracle $\prod_{i,j,k}^{t}$ that has a matching session with the oracle $\prod_{i,j,k}^{s}$.

When the adversary $\mathcal{A}$ terminates the game, the output is the judgment b' of the feedback result of the Test query. The adversary $\mathcal{A}$ is considered to win the game if $b' = b$. The advantage of adversary $\mathcal{A}$ winning in the game is defined as:

$$Adv^{Tri-KA}(\mathcal{A}) = \Pr[b' = b] - \frac{1}{2} \tag{6}$$

Definition 3: Secure tripartite key agreement protocol: Based on the above definition, if a tripartite key agreement protocol satisfies the following conditions, the protocol is considered to be a secure tripartite key agreement protocol:

(1) Protocol participants in matching sessions calculate and obtain the same final session key.
(2) For any adversary $\mathcal{A}$, the advantage $Adv^{Tri-KA}(\mathcal{A})$ of winning the game in polynomial time is negligible.

III. PROPOSED SESSION KEY ESCROW PROTOCOL BASED ON TRIPARTITE KEY AGREEMENT

A. ARCHITECTURE OF THE PROPOSED SESSION KEY ESCROW PROTOCOL

As shown in Fig. 2, the architecture of the proposed session key escrow protocol has six modules: Key Generation Center (KGC), Law Enforcement Agency (LEA), Key Escrow Agents (KEAs), LEA Monitor, IM server and IM client.

The functions of these modules are described in detail as follows:

1) KEY GENERATION CENTER

KGC is responsible for the initialization and maintenance of the public key cryptosystem in the IM system, selecting the appropriate elliptic curve and parameters to ensure that the public key cryptosystem is sufficiently secure. Then, based on a user's unique identifier (such as ID card number, email address or cellphone number), KGC generates a public and private key for the user. After that, KGC sends the user's public/private key to the user in a secure way.

2) LAW ENFORCEMENT AGENCY

As an independent organization, LEA has a unique identifier like a normal user and applies to the KGC for a public/private key. Then LEA holds its private key and publishes its public key and ephemeral public key to all users. After that, LEA shares its ephemeral private key to n KEAs with (t, n) threshold cryptography and then deletes the ephemeral public key. When two ordinary users perform a key agreement protocol, LEA participates in the session key agreement process as an ordinary user by using its own public key and ephemeral public key. Thus, the session key agreement process between two ordinary users and LEA is a tripartite session key agreement protocol.

When it is necessary to monitor the communications of a user, LEA applies for authorization from the government or regulatory agency. After being authorized, LEA uses its private key to compute the partial computation result of the user's session key and sends this partial computation result to LEA Monitor.

3) KEY ESCROW AGENTS

Each of the n KEAs secretly holds a subkey of LEA's ephemeral private key. Based on (t, n) threshold cryptography, t KEAs can use their holding subkeys to recover the ephemeral private key of LEA. After receiving the monitor's authorization,, t KEAs use their holding secret subkeys to compute partial computation results of the user's session key and send the partial computation results to LEA Monitor.

4) LEA MONITOR

After receiving the partial computation results of the user's session key from LEA and the KEAs, LEA Monitor will recover the user's session key. With the recovered session key, LEA Monitor can monitor user communications by decrypting the ciphertexts it has received from IM server.

5) IM SERVER

IM server serves as a message transfer station in this scheme. This means that any message received by the user will be transferred through IM server, including messages for the session key agreement phase and messages for the normal session phase. Therefore, IM server can store public messages for computing secret session keys and ciphertext communications from users, which can be used for monitoring and auditing when necessary.

6) IM CLIENT

The IM client stores the user's public/private key and executes a key agreement protocol with other IM clients. It also encrypts sent messages and decrypts received messages with the negotiated session key.

B. SETUP OF THE PROPOSED SESSION KEY ESCROW PROTOCOL

1) SETUP OF KGC

a) KGC selects a suitable elliptic curve $E(F_p)$ which can meet the security requirements for a secure cryptosystem.

b) KGC selects a point P on $E(F_p)$ as a generator, where the order of point P is a large prime number denoted q. Then, KGC can obtain the additive cyclic group G generated by point P, such that the order of group G is also q.

c) KGC randomly selects a number $s_0 \in Z_q^*$ as the master private key of KGC and stores it in secret. Then, KGC calculates its public key as $P_{KGC} = s_0 \cdot P$.

d) KGC selects three cryptographic hash functions H_0, H_1 and H_2, as follows:

$$\begin{aligned} &H_0 : G \to Z_q^* \\ &H_1 : \{0,1\}^* \times G \to Z_q^* \\ &H_2 : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times G \times G \times G \\ &\quad \times G \times G \times G \times G \times G \times G \times G \to \{0,1\}^k \end{aligned} \tag{7}$$

e) KGC exposes the system parameters P_{pub} to all users in the system:

$$P_{pub} = \{E(F_p), P, q, G, P_{KGC}, H_0, H_1, H_2\} \tag{8}$$

2) EXTRACT PRIVATE KEY FOR USER

For any user A in the system, we use ID_A to indicate his/her unique identifier. KGC selects a number $r_A \in Z_q^*$ randomly and calculates $R_A = r_A \cdot P$ and $h_A = H_1(ID_A, R_A)$. Then, KGC makes R_A and ID_A known to the public. Next, KGC calculates the private key $Q_A = r_A + h_A \cdot s_0$ of user A and sends Q_A to user A in a secure way. Finally, the public key P_A of user A is calculated as in equation (9):

$$P_A = Q_A \cdot P = R_A + H_1(ID_A, R_A) \cdot P_{KGC} \tag{9}$$

3) SETUP OF LEA AND KEAS

After initialization of KGC has been completed, LEA immediately applies to KGC for a public/private key as a normal user with its unique identifier ID_{LEA}. Then, LEA holds its private key Q_{LEA}, while R_{LEA} and ID_{LEA} are publicly released by KGC. After that, LEA performs the following process:

a) LEA selects a number $E_{LEA} \in Z_q^*$ randomly as its ephemeral private key and calculates its ephemeral public key T_{LEA} as $T_{LEA} = E_{LEA} \cdot P$. Then, LEA makes T_{LEA} known to the public.

b) LEA randomly selects a polynomial of degree $t - 1$in Z_q^* with its ephemeral private key E_{LEA}:

$$P_{t-1}(x) = E_{LEA} + a_1 x + \cdots + a_{t-1} x^{t-1}, \quad a_i \in Z_q^* \tag{10}$$

c) LEA selects n different numbers $x_i \in Z_q^*, (i = 1, \cdots, n)$ and calculates $y_i (i = 1, \cdots, n)$ as shown in equation (11):

$$y_i = P_{t-1}(x_i) \bmod q, \quad (i = 1, \cdots, n) \tag{11}$$

LEA thus obtains n different points $(x_1, y_1), \cdots, (x_n, y_n)$. Each point is a subkey for the ephemeral private key E_{LEA} of LEA.

d) For n different points and n different KEAs, LEA assigns a unique point $\{(x_i, y_i), i = 1, \cdots, n\}$ to each of the n different KEAs in a secure way and deletes its ephemeral private key E_{LEA}.

e) Each KEA holds its assigned point (x_i, y_i) in secret.

4) SESSION KEY AGREEMENT PHASE

Suppose there are two normal users A and B preparing to execute the proposed session key agreement protocol. As in the above description, ID_A, ID_B, ID_{LEA}, R_A, R_B and R_{LEA} have been published by KGC and T_{LEA} has been published by LEA. In addition, we use the elliptic curve digital signature algorithm (ECDSA) [39] to sign and verify messages transferred in the protocol to achieve identity authentication.

For User A:

a) User A calculates the public keys P_{LEA} and P_B as:

$$\begin{aligned} P_{LEA} &= R_{LEA} + H_1(ID_{LEA}, R_{LEA}) \cdot P_{KGC} \\ P_B &= R_B + H_1(ID_B, R_B) \cdot P_{KGC} \end{aligned} \tag{12}$$

b) User A selects a number $E_A \in Z_q^*$ randomly as its ephemeral private key and calculates some parameters as in (13):

$$\begin{aligned} T_{A1} &= (Q_A + E_A) \cdot (P_{LEA} + T_{LEA}) \\ T_{A2} &= E_A \cdot T_{LEA} \\ T_{A3} &= E_A \cdot P \end{aligned} \tag{13}$$

c) With the ECDSA algorithm, user A calculates $m_A = H_0(T_{A1} + T_{A2} + T_{A3})$. Then, user A selects a number $k_A \in Z_q^*$ randomly and calculates$(x_A, y_A) = k_A \cdot P$. Let $s_{A1} = x_A$ and calculate s_{A2} as:

$$s_{A2} = k_A^{-1}(m_A + Q_A \cdot s_{A1}) \mathrm{mod} q \tag{14}$$

If $s_{A1} = 0$ or $s_{A2} = 0$, user A reselects k_A and recalculates s_{A1} and s_{A2}.

d) User A sends the message $\langle ID_A, T_{A1}, T_{A2}, T_{A3}, s_{A1}, s_{A2} \rangle$ to user B.

For User B:

a) After user B receives the message from A, user B calculates the public keys P_{LEA} and P_A as:

$$\begin{aligned} P_{LEA} &= R_{LEA} + H_1(ID_{LEA}, R_{LEA}) \cdot P_{KGC} \\ P_A &= R_A + H_1(ID_A, R_A) \cdot P_{KGC} \end{aligned} \tag{15}$$

b) User B calculates $m_{A1} = H_0(T_{A1} + T_{A2} + T_{A3})$ and $(x_{A1}, y_{A1}) = s_{A2}^{-1} \cdot (m_{A1} \cdot P + s_{A1} \cdot P_A)$. If $x_{A1} = s_{A1}$, user B accepts the message and executes the following steps; otherwise, user B rejects the message and stops the protocol.

c) User B selects a number $E_B \in Z_q^*$ randomly as its ephemeral private key and calculates some parameters as in (16):

$$\begin{aligned} T_{B1} &= (Q_B + E_B) \cdot (P_{LEA} + T_{LEA}) \\ T_{B2} &= E_B \cdot T_{LEA} \\ T_{B3} &= (Q_B + E_B) \cdot (P_A + T_{A3}) \\ T_{B4} &= E_B \cdot T_{A3} \end{aligned} \tag{16}$$

d) With the ECDSA algorithm, user B calculates $m_B = H_0(T_{B1} + T_{B2} + T_{B3} + T_{B4})$. Then, user B selects a number $k_B \in Z_q^*$ randomly and calculates $(x_B, y_B) = k_B \cdot P$. Let $s_{B1} = x_B$ and calculate s_{B2} as:

$$s_{B2} = k_B^{-1}(m_B + Q_B \cdot s_{B1}) \mathrm{mod}\ q \tag{17}$$

If $s_{B1} = 0$ or $s_{B2} = 0$, user B reselects k_B and recalculates s_{B1} and s_{B2}.

e) User B sends the message $\langle ID_B, T_{B1}, T_{B2}, T_{B3}, T_{B4}, s_{B1}, s_{B2} \rangle$ to user A.

After user A receives the message from B, user A calculates $m_{B1} = H_0(T_{B1} + T_{B2} + T_{B3} + T_{B4})$ and $(x_{B1}, y_{B1}) = s_{B2}^{-1} \cdot (m_{B1} \cdot P + s_{B1} \cdot P_B)$. If $x_{B1} = s_{B1}$, user A accepts the message; otherwise, user A rejects the message and stops the protocol.

Finally, if user A and user B both accept these messages, user A calculates $SK_{A-B-LEA}^1$, $SK_{A-B-LEA}^2$ and session key $SK_{A-B-LEA}$ as:

$$\begin{aligned} SK_{A-B-LEA}^1 &= (Q_A + E_A) \cdot T_{B1} \\ SK_{A-B-LEA}^2 &= E_A \cdot T_{B2} \\ SK_{A-B-LEA} &= H_2(ID_A, ID_B, ID_{LEA}, T_{A1}, T_{A2}, T_{A3}, T_{B1}, T_{B2}, \\ &\qquad T_{B3}, T_{B4}, T_{LEA}, SK_{A-B-LEA}^1, SK_{A-B-LEA}^2) \end{aligned} \tag{18}$$

User B calculates $SK_{B-A-LEA}^1$, $SK_{B-A-LEA}^2$ and session key $SK_{B-A-LEA}$ as:

$$\begin{aligned} SK_{B-A-LEA}^1 &= (Q_B + E_B) \cdot T_{A1} \\ SK_{B-A-LEA}^2 &= E_B \cdot T_{A2} \\ SK_{B-A-LEA} &= H_2(ID_B, ID_A, ID_{LEA}, T_{B1}, T_{B2}, T_{B3}, T_{B4}, T_{A1}, \\ &\qquad T_{A2}, T_{A3}, T_{LEA}, SK_{B-A-LEA}^1, SK_{B-A-LEA}^2) \end{aligned} \tag{19}$$

Morever:

$$\begin{aligned} SK_{A-B-LEA}^1 &= (Q_A + E_A) \cdot T_{B1} \\ &= (Q_A + E_A) \cdot (Q_B + E_B) \cdot (P_{LEA} + T_{LEA}) \\ &= (Q_B + E_B) \cdot (Q_A + E_A) \cdot (P_{LEA} + T_{LEA}) \\ &= SK_{B-A-LEA}^1 \\ &= (Q_A + E_A) \cdot (Q_B + E_B) \cdot (Q_{LEA} + E_{LEA}) \cdot P \\ SK_{A-B-LEA}^2 &= E_A \cdot T_{B2} \\ &= E_A \cdot E_B \cdot T_{LEA} \\ &= E_B \cdot E_A \cdot T_{LEA} \\ &= SK_{B-A-LEA}^2 \\ &= E_A \cdot E_B \cdot E_{LEA} \cdot P \end{aligned} \tag{20}$$

Thus, $SK_{A-B-LEA} = SK_{B-A-LEA}$, i.e., user A and B have the same session key with which to encrypt/decrypt communications between them using a symmetric cryptographic algorithm such as AES or DES.

5) MONITORING PHASE

As mentioned above, IM server serves as a message transfer station, so it will store public messages in the session key agreement phase and ciphertext communications in this secure session. Suppose that the plaintext message M will be encrypted as $M_{Enc} = Enc(SK_{A-B-LEA}, M)$ with the shared session key. When LEA Monitor needs to monitor user communications, LEA Monitor will first apply to LEA for authorization. After obtaining the authorization from LEA, LEA Monitor gets data from IM server as $\langle ID_A, ID_B, T_{A1}, T_{A2}, T_{A3}, T_{B1}, T_{B2}, T_{B3}, T_{B4}, M_{Enc} \rangle$.

Then, for LEA:

a) LEA Monitor sends message $\langle ID_A, ID_B, T_{B3} \rangle$ to LEA.

b) LEA calculates $SK_{LEA-A-B}^{1-LEA}$ with its private key as:

$$SK_{LEA-A-B}^{1-LEA} = Q_{LEA} \cdot T_{B3} \tag{21}$$

c) LEA sends message $\langle ID_A, ID_B, SK_{LEA-A-B}^{1-LEA}\rangle$ to LEA Monitor.

For KEAs:

a) LEA Monitor sends message $\langle ID_A, ID_B, T_{B3}, T_{B4}\rangle$ and authorization to KEAs.

b) Each of t KEAs uses its holding subkey (x_i, y_i) to calculate $SK_{LEA-A-B}^{1-KEA-i}$ and $SK_{LEA-A-B}^{2-KEA-i}$ as:

$$\begin{aligned} SK_{LEA-A-B}^{1-KEA-i} &= y_i \cdot T_{B3} \\ SK_{LEA-A-B}^{2-KEA-i} &= y_i \cdot T_{B4}, (i = 1, 2, \cdots, t) \end{aligned} \quad (22)$$

c) Each of the t KEAs sends message $\langle ID_A, ID_B, x_i, SK_{LEA-A-B}^{1-KEA-i}, SK_{LEA-A-B}^{2-KEA-i}\rangle$ to LEA Monitor.

For LEA Monitor:

After receiving the above messages from LEA and the t KEAs, LEA Monitor calculates $SK_{LEA-A-B}^{1}$ and $SK_{LEA-A-B}^{2}$ as:

$$\begin{aligned} &SK_{LEA-A-B}^{1} \\ &= SK_{LEA-A-B}^{1-LEA} + (\prod_{i=2}^{t} \frac{x_i}{x_i - x_1}) \cdot SK_{LEA-A-B}^{1-KEA-1} \\ &\quad + (\prod_{i=1, i\neq 2}^{t} \frac{x_i}{x_i - x_2}) \cdot SK_{LEA-A-B}^{1-KEA-2} + \cdots \\ &\quad + (\prod_{i=1}^{t-1} \frac{x_i}{x_i - x_t}) \cdot SK_{LEA-A-B}^{1-KEA-t} \\ &= SK_{LEA-A-B}^{1-LEA} + \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot SK_{LEA-A-B}^{1-KEA-j} \\ &SK_{LEA-A-B}^{2} \\ &= (\prod_{i=2}^{t} \frac{x_i}{x_i - x_1}) \cdot SK_{LEA-A-B}^{2-KEA-1} \\ &\quad + (\prod_{i=1, i\neq 2}^{t} \frac{x_i}{x_i - x_2}) \cdot SK_{LEA-A-B}^{2-KEA-2} + \cdots \\ &\quad + (\prod_{i=1}^{t-1} \frac{x_i}{x_i - x_t}) \cdot SK_{LEA-A-B}^{2-KEA-t} \\ &= \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot SK_{LEA-A-B}^{2-KEA-j} \end{aligned} \quad (23)$$

LEA Monitor calculates session key $SK_{LEA-A-B}$ as:

$$\begin{aligned} SK_{LEA-A-B} = H_2(&ID_A, ID_B, ID_{LEA}, T_{A1}, T_{A2}, T_{A3}, T_{B1}, T_{B2}, \\ &T_{B3}, T_{B4}, T_{LEA}, SK_{LEA-A-B}^{1}, SK_{LEA-A-B}^{2}) \end{aligned} \quad (24)$$

Based on equations (5) and (9), the t different subkeys can be used in the Lagrange interpolation polynomial to reconstruct the polynomial $P_{t-1}(x)$:

$$P_{t-1}(x) = L_{t-1}(x) = \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x - x_i}{x_j - x_i}) y_j \quad (25)$$

Based on equation (10), we can see that:

$$\begin{aligned} E_{LEA} = Pt - 1(0) &= \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{-x_i}{x_j - x_i}) y_j \\ &= \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) y_j \end{aligned} \quad (26)$$

Then, the equations in (23) can be transformed as follows:

$$\begin{aligned} &SK_{LEA-A-B}^{1} \\ &= SK_{LEA-A-B}^{1-LEA} + \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot SK_{LEA-A-B}^{1-KEA-j} \\ &= Q_{LEA} \cdot T_{B3} + \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot y_j \cdot T_{B3} \\ &= Q_{LEA} \cdot T_{B3} + E_{LEA} \cdot T_{B3} \\ &= (Q_{LEA} + E_{LEA}) \cdot (Q_B + E_B) \cdot (Q_A + E_A) \cdot P \\ &SK_{LEA-A-B}^{2} \\ &= \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot SK_{LEA-A-B}^{2-KEA-j} \\ &= \sum_{j=1}^{t} (\prod_{i=1, i\neq j}^{t} \frac{x_i}{x_i - x_j}) \cdot y_j \cdot T_{B4} \\ &= E_{LEA} \cdot T_{B4} = E_{LEA} \cdot E_A \cdot E_B \cdot P \end{aligned} \quad (27)$$

Thus, $SK_{LEA-A-B}^{1} = SK_{A-B-LEA}^{1}$, $SK_{LEA-A-B}^{2} = SK_{A-B-LEA}^{2}$, and $SK_{LEA-A-B} = SK_{A-B-LEA}$. Finally, LEA Monitor can obtain the plaintext M by decrypting the ciphertext message M_{Enc} via $M = Dec(SK_{LEA-A-B}, M_{Enc})$.

IV. SECURITY PROOF AND ANALYSIS OF THE PROPOSED SESSION KEY ESCROW PROTOCOL

As the proposed session key escrow protocol is based on threshold cryptography and a new tripartite authenticated key agreement protocol, we will prove the security of the proposed session key escrow protocol in two steps. First, we will prove that the new tripartite authenticated key agreement protocol is secure. Second, we will prove that the proposed session key escrow scheme, which is based on the new tripartite authenticated key agreement protocol and threshold cryptography, is secure and thus can avoid the ''once monitor, monitor forever'' scenario.

A. SECURITY PROOF OF THE PROPOSED TRIPARTITE AUTHENTICATED KEY AGREEMENT PROTOCOL

Suppose that the hash functions H_1 and H_2 are modeled as two random oracles in the security game. The adversary $\mathcal{A}$ can make q_1 queries for H_1 and q_2 queries for H_2 and perform q_0 protocol instances (i.e., generate q_0 oracles $\prod$). The probability that the adversary $\mathcal{A}$ wins the game is $\varepsilon(k)$.

Lemma 1: If the CDH assumption is true on an elliptic curve, then the proposed tripartite authenticated key agreement protocol in this paper is a secure Tri-KA protocol.

TABLE 1. The worst-case analysis when the adversary controls the private key and ephemeral private key of the protocol participants.

	Q_A	E_A	Q_B	E_B	Q_{LEA}	E_{LEA}
E_1	×	√	×	√	×	√
E_2	×	√	×	√	√	×
E_3	×	√	√	×	×	√
E_4	×	√	√	×	√	×
E_5	√	×	×	√	×	√
E_6	√	×	×	√	√	×
E_7	√	×	√	×	×	√
E_8	√	×	√	×	√	×

Q_A,Q_B and Q_{LEA} are the private keys of U_A,U_B and LEA, respectively, and E_A, E_B and E_{LEA} are the ephemeral private keys of U_A,U_B and LEA, respectively. " √ " means that the private key or ephemeral private key has not been controlled by the adversary $\mathcal{A}$. In contrast, "×" means that the private key or ephemeral private key has been controlled by the adversary $\mathcal{A}$.

Proof: From the equations in (20) and (27), we can determine that protocol participants in matching sessions obtain the same final session key, which meets the first condition in definition 3. Next, we should prove that the proposed tripartite authenticated key agreement protocol meets the second condition in definition 3. We will prove this by using the reduction to absurdity method in the following. Suppose there is an algorithm that can finish the game in polynomial time, and the adversary $\mathcal{A}$ can use the algorithm to win in the secure game with a nonnegligible advantage.

Since the ECDSA algorithm is a mature signature algorithm, its security has been proven and verified. Therefore, we only need to prove that the proposed Tri-AKA protocol is a secure Tri-KA protocol under the CDH assumption when it removes the ECDSA signature algorithm.

Given three CDH problem instances (P, $a \cdot P$, $b \cdot P$), (P, $c \cdot P$, $d \cdot P$) and (P, $m \cdot P$, $n \cdot P$), called CDH_1, CDH_2 and CDH_3, respectively, suppose that there is a polynomial time algorithm ALG that the adversary $\mathcal{A}$ can use to solve the CDH problem. According to the security model given in Chapter 3 and the definition of the fresh oracle, if the security game can on-going then it will eventually enter the Test ($\prod_{i,j,k}^{s}$) query phase. For the oracle $\prod_{i,j,k}^{s}$,$\prod_{i,j,k}^{s}$ now represents the s-th protocol execution instance in the attack process (it no longer specifically refers to the s-th protocol execution instance in those protocol instances initiated only by user U_i). Algorithm ALG randomly selects U_A, U_B, U_{LEA} and $\prod_{A,B,LEA}^{T}$ as the targets of the adversary $\mathcal{A}$, where U_A, U_B, U_{LEA} $\in$[1,q_1], $T \in [1, q_0]$, and $\prod_{A,B,LEA}^{T}$ is the final test session. The worst-case analysis when the adversary $\mathcal{A}$ controls the private key and ephemeral private key of the protocol participants is shown in Table 1.

To finish the proof, here *Suc* represents the adversary $\mathcal{A}$ winning in the security game. We will continue our proof of the second condition in definition 3 by analyzing each case as follows. Since there are many similarities in the analysis process of the eight cases, this article only analyzes the first case in detail, and then analyzes the last seven cases briefly.

1) $E_1 \wedge Suc$

Setup Phase: The adversary $\mathcal{A}$ uses the ALG algorithm to simulate the system initialization process. Algorithm ALG randomly selects P_{KGC} as the system's public key. The hash functions $H_1$and H_2 are instantiated by the algorithm ALG into two random oracles. Algorithm ALG randomly selects U_i, U_j and U_k to simulate the real protocol process and then allows the adversary $\mathcal{A}$ to attack. In the security game, U_i and U_j simulate normal users U_A and U_B and U_k simulates LEA. According to the security model in Chapter 3, the adversary $\mathcal{A}$ conducts the queries in the first phase of the security game in any order, and the algorithm ALG answers.

H_1- *Query:* Algorithm ALG maintains a list L_{H1} that is initialized to be empty. The format of each tuple in the list is (ID_i, R_i, Q_i, h_i), where Q_i represents the private key of ID_i. Then, the public key of ID_i can be expressed as:

$$P_i = Q_i \cdot P_{KGC} = R_i + h_i \cdot P_{KGC} \tag{28}$$

Algorithm ALG answers this query as follows:

a) If there is a tuple matching ID_i in the list L_{H1}, directly return the answer Q_i.
b) Otherwise, algorithm ALG randomly selects h_i, $Q_i \in Z_q^*$. Let Q_i represent the private key of ID_i, let $H_1(ID_i, R_i) = h_i$, and calculate $R_i = Q_i \cdot P - h_i \cdot P_{KGC}$. Return Q_i as the response and insert the tuple (ID_i, R_i, Q_i, h_i) into the list L_{H1}.

Corrupt - Query: Algorithm ALG queries the list L_{H1}. If there is no tuple matching ID_i in the list L_{H1}, ALG performs $H_1(ID_i)$ query with ID_i as the index. Otherwise, return Q_i as the response.

Send - Query: Algorithm ALG maintains a list L_S that is initialized to be empty. The format of each tuple in the list is (ID_i, $\prod_{i,j,k}^{s}$, $r_{i,j,k}^{s}$, $tran_{i,j,k}^{s}$), where $tran_{i,j,k}^{s}$ represents the message sent by ID_i in the protocol instance $\prod_{i,j,k}^{s}$, and $r_{i,j,k}^{s} \in Z_q^*$ is randomly generated by the oracle and uses the ephemeral private key of ID_i to generate the message $tran_{i,j,k}^{s}$. Algorithm ALG answers this query as follows:

a) Observing the key agreement protocol of Chapter 4, a complete protocol instance execution process will only generate three messages in the channel (LEA generates the first message as a normal user, and then user A generates the second message and user B generates the third message). If $tran_{i,j,k}^{s}$ is the third message in a protocol instance, the adversary $\mathcal{A}$ accepts the oracle after message $tran_{i,j,k}^{s}$ has been received.
b) If $tran_{i,j,k}^{s}$ is the first message, to facilitate the understanding of the proof process, let ID_k replace ID_i: If $ID_k = ID_{LEA}$, let $r_{i,j,k}^{s} = \perp$ and

$$tran_{i,j,k}^{s} = T_{LEA} = c \cdot P \tag{29}$$

where $c \cdot P$ is in the CDH_2 instance, and return $c \cdot P$ as the answer.
Otherwise, ALG randomly selects $r_{i,j,k}^{s} \in Z_q^*$ and calculates

$$tran_{i,j,k}^{s} = T_k = r_{i,j,k}^{s} \cdot P \tag{30}$$

Return $tran^s_{i,j,k}$ as the answer.
Finally, *ALG* inserts the tuple $(ID_k, \prod^s_{i,j,k}, r^s_{i,j,k}, tran^s_{i,j,k})$ into list L_S.

c) If $tran^s_{i,j,k}$ is the second message: If $ID_i = ID_A$, let $r^s_{i,j,k} = \perp$, $T_{i2} = a \cdot P$ in the CDH_1 instance, $T_{i3} = n \cdot P$ in the CDH_3 instance, and

$$T_{i1} = Q_A \cdot Q_k \cdot P + Q_A \cdot T_k + Q_k \cdot T_{i3} + T_{i2} \quad (31)$$

where Q_A and Q_k can be obtained through Corrupt (ID_A) query and Corrupt (ID_k) query.
Otherwise, *ALG* randomly selects $r^s_{i,j,k} \in Z^*_q$ and calculates:

$$\begin{aligned} T_{i3} &= r^s_{i,j,k} \cdot P \\ T_{i2} &= r^s_{i,j,k} \cdot T_k \\ T_{i1} &= (Q_i + r^s_{i,j,k}) \cdot (Q_k \cdot P + T_k) \end{aligned} \quad (32)$$

where Q_i,Q_k and T_k can be obtained through Corrupt (ID_i) query, Corrupt (ID_k) query and Send $(ID_k, \prod^s_{i,j,k})$ query.
Finally, whether or not $ID_i = ID_A$, let $tran^s_{i,j,k} = (T_{i1}, T_{i2}, T_{i3})$ be the answer to return, and insert the tuple $(ID_i, \prod^s_{i,j,k}, r^s_{i,j,k}, tran^s_{i,j,k})$ into list L_S.

d) If $tran^s_{i,j,k}$ is the third message (to complete the security proof, we add a parameter T_{j5} in the third message), to facilitate the understanding of the proof process, let ID_j replace ID_i: If $ID_j = ID_B$, let $r^s_{i,j,k} = \perp$, $T_{j2} = m \cdot P$ in the CDH_3 instance, $T_{j4} = d \cdot P$ in the CDH_2 instance, $T_{j5} = b \cdot P$ in the CDH_1 instance, and

$$\begin{aligned} T_{j1} &= Q_j \cdot Q_k \cdot P + Q_j \cdot T_k + T_{j2} + Q_k \cdot T_{j5} \\ T_{j3} &= Q_j \cdot Q_i \cdot P + Q_j \cdot T_{i3} + Q_i \cdot T_{j5} + T_{j4} \end{aligned} \quad (33)$$

where Q_i, Q_j and Q_k can be obtained through Corrupt (ID_i) query, Corrupt (ID_j) query and Corrupt (ID_k) query and T_k and T_{i3} can be obtained through Send $(ID_k, \prod^s_{i,j,k})$ query and Send $(ID_i, \prod^s_{i,j,k})$ query.

Otherwise, *ALG* randomly selects $r^s_{i,j,k} \in Z^*_q$ and calculates

$$\begin{aligned} T_{j1} &= (Q_j + r^s_{i,j,k}) \cdot (Q_k \cdot P + T_k) \\ T_{j2} &= r^s_{i,j,k} \cdot T_k \\ T_{j3} &= (Q_j + r^s_{i,j,k}) \cdot (Q_i \cdot P + T_{i3}) \\ T_{j4} &= r^s_{i,j,k} \cdot T_{i3} \\ T_{j5} &= r^s_{i,j,k} \cdot P \end{aligned} \quad (34)$$

where Q_i, Q_j, Q_k, T_k and T_{i3} are obtained as mentioned above (through Corrupt () query and Send () query).

Finally, whether or not $ID_j = ID_B$, let $tran^s_{i,j,k} = (T_{j1}, T_{j2}, T_{j3}, T_{j4}, T_{j5})$ be the answer to return, and insert the tuple $(ID_j, \prod^s_{i,j,k}, r^s_{i,j,k}, tran^s_{i,j,k})$ into list L_S.

EphemeralKeyReveal - Query: If $ID_i \neq ID_{LEA}$, $ID_i \neq ID_A$ and $ID_i \neq ID_B$, query the list L_S with index $(ID_i, \prod^s_{i,j,k})$ and return $r^s_{i,j,k}$ in tuple as the answer. Otherwise, the algorithm *ALG* aborts the security game.

Reveal - Query: Algorithm *ALG* maintains a list L_R that is initialized to be empty. The format of each tuple in the list is $(\prod^s_{i,j,k}, ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k}, SK^s_{i-j-k})$, where $tran^{s-i}_{i,j,k}$, $tran^{s-j}_{i,j,k}$ and $tran^{s-k}_{i,j,k}$ are messages sent by participants U_i, U_j and U_k in the protocol and SK^s_{i-j-k} is the final agreement session key for $\prod^s_{i,j,k}$. Algorithm *ALG* answers this query as follows:

a) Algorithm *ALG* queries list L_S with index $\prod^s_{i,j,k}$; if oracle $\prod^s_{i,j,k}$ has not been accepted, return $\perp$ as the answer.

b) If $s = T$ or $\prod^s_{i,j,k}$ is the matching session of $\prod^T_{A,B,LEA}$, algorithm *ALG* aborts the security game.

c) Look for a tuple in list L_R with index $(\prod^s_{i,j,k}, ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k})$; if such a tuple exists and $SK^s_{i-j-k} \neq \perp$, return SK^s_{i-j-k} as the answer. Otherwise, execute step d) or e).

d) If $ID_k = ID_{LEA}$, $ID_i = ID_A$ and $ID_j = ID_B$: This means that $E_A = \perp$, $E_B = \perp$ and $E_{LEA} = \perp$, and SK^{s-1}_{i-j-k} and SK^{s-2}_{i-j-k} cannot be directly calculated. Therefore, find a tuple in list L_{H2} with index $(ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k})$, where $tran^{s-i}_{i,j,k}$, $tran^{s-j}_{i,j,k}$ and $tran^{s-k}_{i,j,k}$ can be obtained in the list L_S with index $(\prod^s_{i,j,k}, ID_i)$, $(\prod^s_{i,j,k}, ID_j)$ and $(\prod^s_{i,j,k}, ID_k)$ respectively. If the tuple exists and satisfies:

$$\begin{aligned} e(SK^{s-2}_{i-j-k}, P) &= e(a \cdot P, b \cdot P) = e(c \cdot P, d \cdot P) \\ &= e(m \cdot P, n \cdot P) \\ e(SK^{s-1}_{i-j-k}, P) &= e(T_{A1}, Q_B \cdot P + b \cdot P) \\ &= e(T_{B1}, Q_A \cdot P + n \cdot P) \\ &= e(T_{B3}, Q_{LEA} \cdot P + c \cdot P) \end{aligned} \quad (35)$$

then let $SK^s_{i-j-k} = h_K$. (Note that Q_A, Q_B and Q_{LEA} can be obtained through list L_{H1} and h_K is explained in $H2$-Query below.) Otherwise, *ALG* randomly generates $SK^s_{i-j-k} \in \{0, 1\}^k$ and inserts the tuple $(\prod^s_{i,j,k}, ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k}, SK^s_{i-j-k})$ into list L_R, where $k$is the system-specified length of the session key. Finally, *ALG* returns SK^s_{i-j-k} as the answer.

e) Otherwise, $Q_i \neq \perp$ and $E_i \neq \perp$, where private key Q_i and ephemeral private key E_i can be obtained through $H_1(ID_i)$ query and Send $(ID_i, \prod^s_{i,j,k})$ query. In this case, *ALG* computes $SK^{s-2}_{i-j-k} = E_i \cdot T_{j2}$ and $SK^{s-1}_{i-j-k} = (Q_i + E_i) \cdot T_{j1}$, where T_{j1} and T_{j2} can be obtained through Send $(ID_j, \prod^s_{i,j,k})$ query. Then, query the list L_{H2} with index $(ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k}, SK^{s-1}_{i-j-k}, SK^{s-2}_{i-j-k})$ to obtain the session key $SK^s_{i-j-k} = h_K$. Finally, *ALG* returns SK^s_{i-j-k} as the answer and inserts the tuple $(\prod^s_{i,j,k}, ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k}, SK^s_{i-j-k})$ into list L_R.

H_2 - Query: Algorithm *ALG* maintains a list L_{H2} that is initialized to be empty. The format of each tuple in the list is $(ID_i, ID_j, ID_k, tran^{s-i}_{i,j,k}, tran^{s-j}_{i,j,k}, tran^{s-k}_{i,j,k}, SK^{s-1}_{i-j-k}, SK^{s-2}_{i-j-k}, h_K)$, where h_K is the agreement session key for this protocol instance. Algorithm *ALG* answers this query as follows:

a) Algorithm *ALG* looks for a tuple in list L_{H2} with index (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}). If such a tuple exists and $h_K \neq \perp$, return h_K as the answer.

b) Otherwise, *ALG* looks for a tuple in list L_R with index (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$). If such a tuple exists, let $h_k = SK_{i-j-k}^{s}$ and delete the tuple in list L_R. Then, check whether the parameters in the tuple satisfy

$$\begin{aligned} e(SK_{i-j-k}^{s-2}, P) &= e(T_{i2}, T_{j5}) = e(T_k, T_{j4}) \\ &= e(T_{j2}, T_{i3}) \\ e(SK_{i-j-k}^{s-1}, P) &= e(T_{i1}, P_j + T_{j5}) \\ &= e(T_{j1}, P_i + T_{i3}) \\ &= e(T_{j3}, P_k + T_k) \end{aligned} \tag{36}$$

where $P_i = R_i + h_i \cdot P_{KGC}$, $P_j = R_j + h_j \cdot P_{KGC}$, $P_k = R_k + h_k \cdot P_{KGC}$, and R_i, h_i, R_j, h_j, and R_k, h_k can be obtained through H_1 query. If the equations in (36) are true, insert the tuple (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}, h_K) into list L_{H2} and return h_k as the answer. Otherwise, randomly regenerate $h'_k \in \{0,1\}^k$, insert the tuple (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}, h'_k) into list L_{H2}, and return h'_k as the answer.

c) If there is not a tuple in list L_R with index (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$), randomly generate $h_k \in \{0,1\}^k$, insert the tuple (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}, h_k) into list L_{H2}, and return h_k as the answer.

With this, the first phase is completed. According to the security model, the adversary $\mathcal{A}$ can perform only one Test query in the second phase of the security game.

Test - Query: Algorithm *ALG* answers this query as follows:

a) If $s \neq T$ (which means that oracle $\prod_{A,B,LEA}^{T}$ is not selected as the final test session) or the oracle that has a matching session with oracle $\prod_{A,B,LEA}^{T}$ has been corrupted, *ALG* aborts the security game.

b) Otherwise, *ALG* randomly selects $h_\varepsilon \in \{0,1\}^k$ as the answer. Now $E_k = \perp$ and $T_k = c \cdot P$; $E_i = \perp$ and $T_{i2} = a \cdot P$ and $T_{i3} = n \cdot P$; and $E_j = \perp$ and $T_{j2} = m \cdot P$, $T_{j4} = d \cdot P$, and $T_{j5} = b \cdot P$.

Suppose that the adversary $\mathcal{A}$ finally wins the game and the probability of winning, expressed as $\varepsilon(k)$, is not negligible; then *ALG* must not abort the security game, H_2 query with index (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}) has been executed, and a tuple (ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s-1}, SK_{i-j-k}^{s-2}, h_K) has been obtained whose parameters satisfy equation (36). To solve the three given CDH problems, the solutions of CDH instances CDH_1, CDH_2 and CDH_3 can be derived from L_{H2} as follows:

$$a \cdot b \cdot P = c \cdot d \cdot P = m \cdot n \cdot P = SK_{i-j-k}^{s-2} \tag{37}$$

Then, the probability that *ALG* solves these CDH problems is:

$$Adv_{ALG}^{CDH}(k) \geq \frac{\varepsilon(k)}{3 \cdot q_0 \cdot q_1^3 \cdot q_2} \tag{38}$$

which means that solving one of the three given CDH instances leads to winning the security game. Now, if $\varepsilon(k)$ is nonnegligible, then $Adv_{ALG}^{CDH}(k)$ is obviously nonnegligible, which contradicts the CDH problem in Chapter 2. Therefore, the probability that the adversary $\mathcal{A}$ wins the game is negligible.

2) $E_2 \wedge Suc$

In this case, $E_A = \perp$, $E_B = \perp$, and $Q_{LEA} = \perp$. In contrast to the first case, for the tuple ($\prod_{i,j,k}^{s}$, ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s}) in list L_R, if $ID_k = ID_{LEA}$, *ALG* randomly selects $E_k = r_{i,j,k}^{s-k} \in Z_q^*$ and calculates $T_k = E_k \cdot P$, and then lets $Q_k = \perp$ and $P_k = c \cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, *ALG* randomly selects $Q_A \in Z_q^*$ and lets $E_A = \perp$, lets $E_A \cdot P_k$ be expressed as $m \cdot P$ in the CDH instance CDH_3, lets $T_{A3} = a \cdot P$ in the CDH instance CDH_1, and lets T_{A1} and T_{A2} be calculated as:

$$\begin{aligned} T_{A2} &= E_k \cdot T_{A3} \\ T_{A1} &= Q_A \cdot P_k + Q_A \cdot T_k + m \cdot P + T_{A2} \end{aligned} \tag{39}$$

If $ID_j = ID_B$, *ALG* randomly selects $Q_B \in Z_q^*$ and lets $E_B = \perp$, lets $E_B \cdot P_k$ be expressed as $b \cdot P$ in the CDH instance CDH_1, lets $T_{B5} = n \cdot P$ in the CDH instance CDH_3, lets $T_{B4} = d \cdot P$ in the CDH instance CDH_2, and lets T_{B2}, T_{B1} and T_{B3} be calculated as:

$$\begin{aligned} T_{B2} &= E_k \cdot T_{B5} \\ T_{B1} &= Q_B \cdot P_k + Q_B \cdot T_k + b \cdot P + T_{B2} \\ T_{B3} &= Q_B \cdot P_A + Q_B \cdot T_{A3} + Q_A \cdot T_{B5} + T_{B4} \end{aligned} \tag{40}$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances (P, $a \cdot P$, $b \cdot P$), (P, $c \cdot P$, $d \cdot P$) and (P, $m \cdot P$, $n \cdot P$) can be derived from L_{H2} as:

$$\begin{aligned} a \cdot b \cdot P &= c \cdot d \cdot P = m \cdot n \cdot P \\ &= SK_{i-j-k}^{s-1} - Q_A \cdot T_{B1} - Q_B \cdot (T_{A1} - Q_A \cdot (P_k + T_k)) \\ &\quad - SK_{i-j-k}^{s-2} \end{aligned} \tag{41}$$

Then, the probability that *ALG* solves these CDH problems is as in equation (38).

3) $E_3 \wedge Suc$

In this case, $E_A = \perp$, $Q_B = \perp$, and $E_{LEA} = \perp$. For the tuple ($\prod_{i,j,k}^{s}$, ID_i, ID_j, ID_k, $tran_{i,j,k}^{s-i}$, $tran_{i,j,k}^{s-j}$, $tran_{i,j,k}^{s-k}$, SK_{i-j-k}^{s}) in list L_R, if $ID_k = ID_{LEA}$, *ALG* randomly selects $Q_k \in Z_q^*$ and lets $E_k = \perp$ and $T_k = c \cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, *ALG* randomly selects $Q_A \in Z_q^*$ and lets $E_A = \perp$ and $T_{A2} = m \cdot P$ in the CDH instance CDH_3, lets

$T_{A3} = a \cdot P$ in the CDH instance CDH_1, and calculates T_{A1} as:

$$T_{A1} = Q_A \cdot P_k + Q_A \cdot T_k + Q_k \cdot T_{A3} + T_{A2} \quad (42)$$

If $ID_j = ID_B$, ALG lets $Q_B = \perp$ and $P_B = n \cdot P$ in the CDH instance CDH_3, lets $E_A \cdot P_B$ be expressed by $d \cdot P$ in the CDH instance CDH_2, lets $E_k \cdot P_B$ be expressed by $b \cdot P$ in the CDH instance CDH_1, and then randomly selects $E_B = r_{i,j,k}^{s-k} \in Z_q^*$ and calculates T_{B5}, T_{B4}, T_{B2}, T_{B1} and T_{B3} as:

$$\begin{aligned} T_{B5} &= E_B \cdot P \\ T_{B4} &= E_B \cdot T_{A3}, T_{B2} = E_B \cdot T_k \\ T_{B1} &= Q_k \cdot P_B + b \cdot P + E_B \cdot P_k + T_{B2} \\ T_{B3} &= Q_A \cdot P_B + d \cdot P + E_B \cdot P_A + T_{B4} \end{aligned} \quad (43)$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a \cdot P, b \cdot P)$, $(P, c \cdot P, d \cdot P)$ and $(P, m \cdot P, n \cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a \cdot b \cdot P &= c \cdot d \cdot P = m \cdot n \cdot P \\ &= SK_{i-j-k}^{s-1} - Q_A \cdot T_{B1} - Q_k \cdot (T_{A1} - Q_A \cdot (P_B + T_{B5})) \\ &\quad - SK_{i-j-k}^{s-2} \end{aligned} \quad (44)$$

Then the probability that ALG solves these CDH problems is as in equation (38).

4) $E_4 \wedge Suc$

In this case, $E_A = \perp$, $Q_B = \perp$, and $Q_{LEA} = \perp$. For the tuple $(\prod_{i,j,k}^{s}, ID_i, ID_j, ID_k, tran_{i,j,k}^{s-i}, tran_{i,j,k}^{s-j}, tran_{i,j,k}^{s-k}, SK_{i-j-k}^{s})$ in list L_R, if $ID_k = ID_{LEA}$, ALG randomly selects $E_k = r_{i,j,k}^{s-k} \in Z_q^*$, lets $T_k = E_k \cdot P$, and lets $Q_k = \perp$ and $P_k = c \cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, ALG randomly selects $Q_A \in Z_q^*$ and lets $E_A = \perp$, lets $E_A \cdot P_k$ be expressed by $m \cdot P$ in the CDH instance CDH_3, lets $T_{A3} = a \cdot P$ in the CDH instance CDH_1, and calculates T_{A1} and T_{A2} as in (39).

If $ID_j = ID_B$, ALG lets $Q_B = \perp$ and $P_B = n \cdot P$ in the CDH instance CDH_3, lets $E_A \cdot P_B$ be expressed by $d \cdot P$ in the CDH instance CDH_2, lets $Q_k \cdot P_B$ be expressed by $b \cdot P$ in the CDH instance CDH_1, and then randomly selects $E_B = r_{i,j,k}^{s-k} \in Z_q^*$ and calculates T_{B1} as in (45) and T_{B2}, T_{B3}, T_{B4}, and T_{B5} as in (43):

$$T_{B1} = b \cdot P + E_k \cdot P_B + E_B \cdot P_k + T_{B2} \quad (45)$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a \cdot P, b \cdot P)$, $(P, c \cdot P, d \cdot P)$ and $(P, m \cdot P, n \cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a \cdot b \cdot P &= c \cdot d \cdot P = m \cdot n \cdot P \\ &= SK_{i-j-k}^{s-1} - Q_A \cdot T_{B1} - E_k \cdot U - E_B \cdot V - SK_{i-j-k}^{s-2} \\ U &= T_{B3} - Q_A \cdot P_B - E_B \cdot P_A - T_{B4} = d \cdot P \\ V &= T_{A1} - Q_A \cdot P_k - Q_A \cdot T_k - T_{A2} = m \cdot P \end{aligned} \quad (46)$$

Then, the probability that ALG solves these CDH problems is as in equation (29).

5) $E_5 \wedge Suc$

In this case, $Q_A = \perp$, $E_B = \perp$, and $E_{LEA} = \perp$. For the tuple $(\prod_{i,j,k}^{s}, ID_i, ID_j, ID_k, tran_{i,j,k}^{s-i}, tran_{i,j,k}^{s-j}, tran_{i,j,k}^{s-k}, SK_{i-j-k}^{s})$ in list L_R, if $ID_k = ID_{LEA}$, ALG randomly selects $Q_k \in Z_q^*$ and lets $E_k = \perp$ and $T_k = c \cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, ALG randomly selects $E_A \in Z_q^*$ and lets $Q_A = \perp$ and $P_A = a \cdot P$ in the CDH instance CDH_1, lets $Q_A \cdot T_k$ be expressed by $m \cdot P$ in the CDH instance CDH_3, and calculates T_{A1}, T_{A2} and T_{A3} as:

$$\begin{aligned} T_{A3} &= E_A \cdot P, \quad T_{A2} = E_A \cdot T_k \\ T_{A1} &= Q_k \cdot P_A + m \cdot P + E_A \cdot P_k + T_{A2} \end{aligned} \quad (47)$$

If $ID_j = ID_B$, ALG randomly selects $Q_B \in Z_q^*$ and lets $E_B = \perp$ and $T_{B5} = n \cdot P$ in the CDH instance CDH_3, lets $E_B \cdot P_A$ be expressed by $d \cdot P$ in the CDH instance CDH_2, lets $T_{B2} = b \cdot P$ in the CDH instance CDH_1, and calculates T_{B1}, T_{B3} and T_{B4} as:

$$\begin{aligned} T_{B4} &= E_A \cdot T_{B5} \\ T_{B1} &= Q_B \cdot P_k + Q_B \cdot T_k + Q_k \cdot T_{B5} + T_{B2} \\ T_{B3} &= Q_B \cdot P_A + Q_B \cdot T_{A3} + d \cdot P + T_{B4} \end{aligned} \quad (48)$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a \cdot P, b \cdot P)$, $(P, c \cdot P, d \cdot P)$ and $(P, m \cdot P, n \cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a \cdot b \cdot P &= c \cdot d \cdot P = m \cdot n \cdot P \\ &= SK_{i-j-k}^{s-1} - Q_B \cdot T_{A1} - Q_k \cdot (T_{B3} - Q_B \cdot (P_A + T_{A3})) \\ &\quad - SK_{i-j-k}^{s-2} \end{aligned} \quad (49)$$

Then, the probability that ALG solves these CDH problems is as in equation (38).

6) $E_6 \wedge Suc$

In this case, $Q_A = \perp$, $E_B = \perp$, $Q_{LEA} = \perp$. For the tuple $(\prod_{i,j,k}^{s}, ID_i, ID_j, ID_k, tran_{i,j,k}^{s-i}, tran_{i,j,k}^{s-j}, tran_{i,j,k}^{s-k}, SK_{i-j-k}^{s})$ in list L_R, if $ID_k = ID_{LEA}$, ALG randomly selects $E_k = r_{i,j,k}^{s-k} \in Z_q^*$ and $T_k = E_k \cdot P$ and lets $Q_k = \perp$ and $P_k = c \cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, ALG randomly selects $E_A \in Z_q^*$ and lets $Q_A = \perp$ and $P_A = a \cdot P$ in the CDH instance CDH_1, lets $Q_A \cdot P_k$ be expressed by $m \cdot P$ in the CDH instance CDH_3, calculates T_{A2} and T_{A3} as in (47), and calculates T_{A1} as:

$$T_{A1} = m \cdot P + E_k \cdot P_A + E_A \cdot P_k + T_{A2} \quad (50)$$

If $ID_j = ID_B$, ALG randomly selects $Q_B \in Z_q^*$ and lets $E_B = \perp$ and $T_{B5} = n \cdot P$ in the CDH instance CDH_3, lets $E_B \cdot P_A$ be expressed by $d \cdot P$ in the CDH instance CDH_2, lets $E_B \cdot P_k$ be expressed by $b \cdot P$ in the CDH instance CDH_1, and then calculates T_{B1}, T_{B2}, T_{B3} and T_{B4} as:

$$\begin{aligned} T_{B4} &= E_A \cdot T_{B5} \\ T_{B2} &= E_k \cdot T_{B5} \\ T_{B1} &= Q_B \cdot P_k + Q_B \cdot T_k + b \cdot P + T_{B2} \\ T_{B3} &= Q_B \cdot P_A + Q_B \cdot T_{A3} + d \cdot P + T_{B4} \end{aligned} \quad (51)$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a\cdot P, b\cdot P)$, $(P, c\cdot P, d\cdot P)$ and $(P, m\cdot P, n\cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a\cdot b\cdot P &= c\cdot d\cdot P = m\cdot n\cdot P \\ &= SK_{i-j-k}^{s-1} - Q_B\cdot T_{A1} - E_k\cdot U - E_A\cdot V - SK_{i-j-k}^{s-2} \\ U &= T_{B3} - Q_B\cdot P_A - Q_B\cdot T_{A3} - T_{B4} = d\cdot P \\ V &= T_{B1} - Q_B\cdot P_k - Q_B\cdot T_k - T_{B2} = b\cdot P \end{aligned} \tag{52}$$

Then, the probability that *ALG* solves these CDH problems is as in equation (38).

7) $E_7 \wedge Suc$

In this case, $Q_A = \perp$, $Q_B = \perp$, and $E_{LEA} = \perp$. For the tuple $(\prod_{i,j,k}^{s}, ID_i, ID_j, ID_k, tran_{i,j,k}^{s-i}, tran_{i,j,k}^{s-j}, tran_{i,j,k}^{s-k}, SK_{i-j-k}^{s})$ in list L_R, if $ID_k = ID_{LEA}$, *ALG* randomly selects $Q_k \in Z_q^*$ and lets $E_k = \perp$ and $T_k = c\cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, *ALG* randomly selects $E_A \in Z_q^*$ and lets $Q_A = \perp$ and $P_A = a\cdot P$ in the CDH instance CDH_1, lets $Q_A\cdot T_k$ be expressed by $m\cdot P$ in the CDH instance CDH_3, and calculates T_{A1}, $T_{A2}$and T_{A3}as in (47).

If $ID_j = ID_B$, *ALG* lets $Q_B = \perp$ and $P_B = n\cdot P$ in the CDH instance CDH_3, lets $Q_A\cdot P_B$ be expressed by $d\cdot P$ in the CDH instance CDH_2, lets $E_k\cdot P_B$ be expressed by $b\cdot P$ in the CDH instance CDH_1, and then randomly selects $E_B = r_{i,j,k}^{s-k} \in Z_q^*$ and calculates T_{B1}, T_{B2}, T_{B4} and T_{B5} as in (43) and T_{B3}as:

$$T_{B3} = d\cdot P + E_A\cdot P_B + E_B\cdot P_A + T_{B4} \tag{53}$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a\cdot P, b\cdot P)$, $(P, c\cdot P, d\cdot P)$ and $(P, m\cdot P, n\cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a\cdot b\cdot P &= c\cdot d\cdot P = m\cdot n\cdot P \\ &= SK_{i-j-k}^{s-1} - Q_k\cdot T_{B3} - E_A\cdot U - E_B\cdot V - SK_{i-j-k}^{s-2} \\ U &= T_{B1} - Q_k\cdot P_B - E_B\cdot P_k - T_{B2} = b\cdot P \\ V &= T_{A1} - Q_k\cdot P_A - E_A\cdot P_k - T_{A2} = m\cdot P \end{aligned} \tag{54}$$

Then, the probability that *ALG* solves these CDH problems is as in equation (38).

8) $E_8 \wedge Suc$

In this case, $Q_A = \perp$, $Q_B = \perp$, and $Q_{LEA} = \perp$. For the tuple $(\prod_{i,j,k}^{s}, ID_i, ID_j, ID_k, tran_{i,j,k}^{s-i}, tran_{i,j,k}^{s-j}, tran_{i,j,k}^{s-k}, SK_{i-j-k}^{s})$ in list L_R, if $ID_k = ID_{LEA}$, *ALG* randomly selects $E_k = r_{i,j,k}^{s-k} \in Z_q^*$ and $T_k = E_k\cdot P$ and lets $Q_k = \perp$ and $P_k = c\cdot P$ in the CDH instance CDH_2.

If $ID_i = ID_A$, *ALG* randomly selects $E_A \in Z_q^*$ and lets $Q_A = \perp$ and $P_A = a\cdot P$ in the CDH instance CDH_1, lets $Q_A\cdot P_k$ be expressed by $m\cdot P$ in the CDH instance CDH_3, and calculates T_{A2} and T_{A3} as in (47) and T_{A1} as in (50).

If $ID_j = ID_B$, *ALG* lets $Q_B = \perp$ and $P_B = n\cdot P$ in the CDH instance CDH_3, lets $Q_B\cdot P_A$ be expressed by $d\cdot P$ in the CDH instance CDH_2, lets $Q_B\cdot P_k$ be expressed by $b\cdot P$ in the CDH instance CDH_1, and then randomly selects $E_B = r_{i,j,k}^{s-k} \in Z_q^*$ and calculates T_{B5}, T_{B4}, T_{B2}, T_{B1} and T_{B3} as:

$$\begin{aligned} T_{B5} &= E_B\cdot P \\ T_{B4} &= E_B\cdot T_{A3} \\ T_{B2} &= E_B\cdot T_k \\ T_{B1} &= b\cdot P + E_k\cdot P_B + E_B\cdot P_k + T_{B2} \\ T_{B3} &= d\cdot P + E_A\cdot P_B + E_B\cdot P_A + T_{B4} \end{aligned} \tag{55}$$

Finally, check whether these parameters satisfy the equations in (36), and then the solutions of CDH instances $(P, a\cdot P, b\cdot P)$, $(P, c\cdot P, d\cdot P)$ and $(P, m\cdot P, n\cdot P)$ can be derived from L_{H2} as:

$$\begin{aligned} a\cdot b\cdot P &= c\cdot d\cdot P = m\cdot n\cdot P \\ &= SK_{i-j-k}^{s-1} - E_A\cdot T_{B1} - E_B\cdot (T_{A1} - E_A\cdot (P_k + T_k)) \\ &\quad - E_k\cdot U \\ U &= T_{B3} - E_A\cdot P_B - E_B\cdot P_A - T_{B4} = d\cdot P \end{aligned} \tag{56}$$

Then, the probability that *ALG* solves these CDH problems is as in equation (38).

Based on the analyses of the above 8 cases, if the CDH assumption holds, the probability $Adv^{Tri-KA}(\mathcal{A})$ that the adversary $\mathcal{A}$ wins the game is negligible, so the proposed Tri-AKA protocol is a secure Tri-KA protocol under the CDH assumption when it removes the ECDSA signature algorithm.

Because we use the ECDSA signature algorithm to sign and verify messages transferred in the protocol to achieve identity authentication, the new proposed tripartite authenticated key agreement protocol user for session key escrow scheme in this paper is secure.

B. SECURITY ANALYSIS OF THE PROPOSED SESSION KEY ESCROW PROTOCOL

Lemma 2: If the ECDL assumption is true, then the proposed session key escrow scheme in this paper can avoid the ''once monitor, monitor forever'' scenario.

Analysis: For the proposed session key escrow scheme in this paper, LEA participates as a normal user in the tripartite key agreement protocol between users A and B. In the monitoring phase, if LEA holds its private key Q_{LEA} and ephemeral private key E_{LEA}, it can calculate $SK_{LEA-A-B}^{1}$, $SK_{LEA-A-B}^{2}$ and session key $SK_{LEA-A-B}$ directly as:

$$\begin{aligned} SK_{LEA-A-B}^{1} &= (Q_{LEA} + E_{LEA})\cdot T_{B3} \\ SK_{LEA-A-B}^{2} &= E_{LEA}\cdot T_{B4} \\ SK_{LEA-A-B} &= H_2(ID_A, ID_B, ID_{LEA}, T_{A1}, T_{A2}, T_{A3}, T_{B1}, T_{B2}, \\ &\qquad T_{B3}, T_{B4}, T_{LEA}, SK_{LEA-A-B}^{1}, SK_{LEA-A-B}^{2}) \end{aligned} \tag{57}$$

However, as described in the section ''Setup of LEA and KEAs'', in the initialization phase of the system, LEA will share the ephemeral private key E_{LEA} with n KEAs by means of (t, n) threshold cryptography and then delete the E_{LEA}. Therefore, LEA cannot calculate the session key directly; it can reconstruct the session key only by combining at least t

KEAs with equations (21)-(24), which limits LEA abuse of its power and prevents LEA from arbitrarily monitoring user communications.

At the monitoring phase, for the specified session key to be reconstructed, each of the t KEAs should calculate $SK_{LEA-A-B}^{1-KEA-i} = y_i \cdot T_{B3}$ and $SK_{LEA-A-B}^{2-KEA-i} = y_i \cdot T_{B4}$ with its secret subkey (x_i, y_i) of E_{LEA}, and then send message $< ID_A, ID_B, x_i, SK_{LEA-A-B}^{1-KEA-i}, SK_{LEA-A-B}^{2-KEA-i} >$ to LEA Monitor. As $y_i \cdot T_{B3}$ and $y_i \cdot T_{B4}$ are ECDL problems, LEA Monitor cannot obtain $y_i$from these messages, which means it cannot reconstruct another session key of message $\langle ID_A, ID_B, T_{A1}', T_{A2}', T_{A3}', T_{B1}', T_{B2}', T_{B3}', T_{B4}', M_{Enc}\rangle$ unless KEAs are authorized to calculate $SK_{LEA-A-B}^{1-LEA-i}$ and $SK_{LEA-A-B}^{2-KEA-i\prime}$ as:

$$\begin{aligned} SK_{LEA-A-B}^{1-KEA-i\prime} &= y_i \cdot T_{B3}' \\ SK_{LEA-A-B}^{2-KEA-i\prime} &= y_i \cdot T_{B4}', (i = 1, 2, \cdots, t) \end{aligned} \tag{58}$$

This means that the proposed key escrow scheme can avoid the ''once monitor, monitor forever'' scenario.

According to the above analyses, the proposed session key escrow scheme in this paper can prevent malicious administrators in LEA from arbitrarily monitoring user communications and avoid the ''once monitor, monitor forever'' scenario.

V. COMPARISON OF THE PROPOSED SCHEME WITH OTHERS

As the proposed session key escrow scheme is based on a new tripartite authenticated key agreement protocol and (t, n) threshold cryptography, we will analyze the proposed tripartite AKA protocol and session key escrow scheme and compare them with other protocols and schemes, respectively. First, we present a comparison of the proposed tripartite AKA protocol with others, and then we present a comparison of the proposed session key escrow scheme with others.

A. COMPARISON OF THE PROPOSED TRIPARTITE AKA PROTOCOL WITH OTHERS

The comparison result of the proposed tripartite AKA protocol with others is shown in Table 2. In this table, ''ESRR'' stands for Ephemeral Secret Reveal Resistance, which means the attacker cannot obtain the session key even if all users' ephemeral private keys have been revealed; ''P'' is a point on the elliptic curve; ''h'' is the result of a hash function; ''n'' is a large number that belongs to Z_q^*; ''Pa'' is a pairing computation; ''M'' is a multiplication computation; ''E'' is an exponentiation computation; ''Std'' is the Standard Model; and ''RO'' is the Random Oracle Model. It is generally believed that the time overhead of multiplication is equivalent to that of exponentiation, but the time overhead of pairing is indeed approximately 20 times greater than these [40].

Table 2 shows that Ref. [34] has only one communication round but has 14 pairing operations; Ref. [36] has the smallest computational overhead but has 4 communication rounds and cannot support ESRR in terms of security, while our protocol has 2 communication rounds and 9 multiplication operations, making it suitable for mobile clients (who have poor computational performance) because it is a compromise solution between communication rounds and computational overhead. Although our protocol has a higher communication overhead than others, with the popularity of 4G networks and the development of 5G networks, we believe that communication overhead will no longer be an important factor in the design of the AKA protocol.

TABLE 2. Comparison of the proposed tripartite aka protocol with others.

	Ref. [30]	Ref. [34]	Ref. [35]	Ref. [36]	Our
Assumption	BDH & CDH	DBDH	BDH & CDH	Modified CDH	CDH
ESRR	×	√	×	×	√
Communication Round	4	1	4	4	2
Communication Overhead	2P	4P	4P	2P+1h+ 3n	4.5P +1n
Computational Overhead	1Pa+5M +1E	14Pa+ 2M+15E	2Pa+5M	2E	9M
Std or RO	RO	Std	RO	RO	RO

B. COMPARISON OF THE PROPOSED SESSION KEY ESCROW SCHEME WITH OTHERS

Table 3 shows computational overhead for different operations of elliptic curves and big integer on a laptop and a mobile device. To explain the performance efficiency of the new scheme and others more intuitively, we conduct some experiments on a laptop and a smart phone based on Java code and Java Pairing Based Cryptography Library (JPBC) [28].

TABLE 3. Computational overhead for different operations of elliptic curves and big integer on a laptop and a mobile device.

Operation	Description	Average Time (ms)	
	Elliptic curve	d(159)	d(201)
P_{EC}^{M}	Pairing on P30	199.04	230.49
P_{EC}^{PC}	Pairing on laptop	20.654	30.55
E_{G1}^{M}	Exponentiation on P30 in G_1	7.22	8.84
E_{G1}^{PC}	Exponentiation on laptop in G_1	2.469	3.757
M_{G1}^{M}	Multiplication on P30 in G_1	7.26	8.76
M_{G1}^{PC}	Multiplication on laptop in G_1	2.465	3.767
A_{G1}^{M}	Addition on P30 in G_1	0.014	0.017
A_{G1}^{PC}	Addition on laptop in G_1	0.032	0.036
E_{GT}^{M}	Exponentiation on P30 in G_T	70.83	80.15
E_{GT}^{PC}	Exponentiation on laptop in G_T	6.305	8.755
M_{GT}^{M}	Multiplication on P30 in G_T	70.07	80.78
M_{GT}^{PC}	Multiplication on laptop in G_T	6.318	8.755
	Big integer	1024 bits	
M_{BI}^{M}	Multiplication on P30	0.0061	
M_{BI}^{PC}	Multiplication on laptop	0.0056	
A_{BI}^{M}	Addition on P30	0.0024	
A_{BI}^{PC}	Addition on laptop	0.0012	
MI_{BI}^{M}	Modular inversion on P30	0.134	
MI_{BI}^{PC}	Modular inversion on laptop	0.128	
ME_{BI}^{M}	Modular exponentiation on P30	1.424	
ME_{BI}^{PC}	Modular exponentiation on laptop	3.162	

TABLE 4. Comparison of the proposed session key escrow scheme with others.

	Escrow Target	Avoid OM-MF	Storage Overhead	Dynamic Agents	Protocol phrase	Execution time (ms) Operations	Execution time (ms) Total	CC(bits)
Ref. [7]	SK	×	|Users|	×	SK agreement	NM	NM	NM
					SK escrow	42 ME_{BI}^{M} +56 M_{BI}^{M} +56 A_{BI}^{M}	60.284	14336
					SK recovery	70 M_{BI}^{PC} +30 A_{BI}^{PC} +10 MI_{BI}^{PC}	1.708	5120
Ref. [12]	s_{KGC}	×	1	×	SK agreement	2 P_{EC}^{M} +4 E_{G1}^{M} + M_{GT}^{M}	577.12	804
					SK escrow	0	0	0
					SK recovery	P_{EC}^{PC} + M_{G1}^{PC}	34.317	804
Ref. [13]	s_{KGC}	×	1	×	SK agreement	2 P_{EC}^{M} +4 A_{G1}^{M} +4 M_{G1}^{M}	496.09	402
					SK escrow	0	0	0
					SK recovery	2 P_{EC}^{PC} +4 A_{G1}^{PC} +2 E_{GT}^{PC}	96.264	2412
Ref. [14]	s_{KGC}	×	1	×	SK agreement	P_{EC}^{M} +4 E_{G1}^{M} + E_{GT}^{M}	346	804
					SK escrow	0	0	0
					SK recovery	P_{EC}^{PC} +2 MI_{BI}^{PC} +2 E_{G1}^{PC} + E_{GT}^{PC}	47.075	804
Ref. [3]	s_{Ui}	√	|Users|	√	SK agreement	NM	NM	NM
					SK escrow	7 MI_{BI}^{PC} +7 M_{BI}^{PC} +7 M_{G1}^{PC} +16 MI_{BI}^{M} +23 ME_{BI}^{M} +23 M_{BI}^{M} +15 M_{G1}^{M} +42 A_{BI}^{M}	193.84	22528
					SK recovery	65 MI_{BI}^{PC} +85 M_{BI}^{PC} +5 M_{G1}^{PC} +26 A_{BI}^{PC}	27.662	6144
Our	E_{LEA}	√	1	×	SK agreement	9 M_{G1}^{M} +12 A_{G1}^{M} +4 MI_{BI}^{M} +2 M_{BI}^{M}	79.592	2010
					SK escrow	21 ME_{BI}^{PC} +28 M_{BI}^{PC} +28 A_{BI}^{PC}	66.592	7168
					SK recovery	21 M_{G1}^{PC} +9 A_{G1}^{PC} +30 M_{BI}^{PC} +15 A_{BI}^{PC} +5 MI_{BI}^{PC}	80.257	4422

The laptop has an Intel Core-i7 CPU at 2.4 GHz, 12 GB memory. The smart phone is Huawei P30 with an 8-core processor, 8 GB memory, and Android 9.0. Table 3 shows the computational overheads for different elliptic curve operations on the laptop and the mobile device. We use the d(n)-type elliptic curve group, which is good for cryptosystems when group elements must be as short as possible. In addition, d(n) denotes that the base field size is n bits and can provide security strength of the equivalent of RSA 6∗n bits keys [41].

Table 4 shows the results of the comparison of the proposed session key escrow scheme with others. In this table, "SK" is the session key between users; "s_{KGC}" is the secure master key of KGC; "E_{LEA}" is the ephemeral private key of LEA; |Users| stands for the number of users in system; "OM-OF" stands for "once monitor, monitor forever"; "ET" stands for the execution time; "CC" stands for communication cost; "NM" stands for not mentioned in the reference paper. We unified select the instance (5, 7) for these schemes which used (t, n) threshold cryptography. For equality, we use the d(201)-type elliptic curve group (an EC point to be 402 bits) and 1024 bits big integer to compare the execution time in different phrases for SK between different session key escrow schemes. Assume that the program for SK agreement phase runs on the smart phone, the program for KEAs and SK recovery phase runs on the laptop. While the SK escrow phase program runs on the smart phone or laptop depends on the reference scheme.

Ref. [12]–[14] focused on the security of the AKA protocol rather than on the practicability of the escrow protocol and user privacy protection. Therefore, although they have higher security in key agreement, KGC can use s_{KGC} to monitor the communications of all users directly, and therefore cannot avoid OM-MF. In addition, letting KGC take on additional monitoring work will increase the management difficulty and risk of being corrupted of the secure master key s_{KGC}. Ref. [3] supports dynamic add/delete escrow agents during system running, but to avoid OM-MF, the user has to update his or her private key s_{Ui} periodically. This means that LEA can obtain all the session keys encrypted by s_{Ui} if it is authorized to recover s_{Ui}. In contrast, in our session key escrow scheme, based on the section above entitled "Monitoring Phase", if LEA is authorized, it can only recover the specified session key; therefore, our scheme has higher fine-grained control on avoiding OM-MF than Ref. [3]. With regard to storage overhead, based on the section above entitled "Setup of LEA and KEAs", LEA will share its ephemeral private key E_{LEA} with n KEAs at the initialization phase of the IM system and will send its ephemeral public key T_{LEA} to a new user after the user has registered in the IM system. In the real running life of the IM system, LEA may generate only one pair E_{LEA} and T_{LEA} over the whole life of the IM system. Thus each KEA must store only one subkey of E_{LEA} over the whole life of the IM system, which is more convenient for KEAs compared with Ref. [3], [7]. Furthermore, the fact that LEA sends its ephemeral public key T_{LEA} to a new user after the user has registered can reduce the communication round to 1 round at the session key agreement phase in our scheme.

In summary, our key escrow scheme fully considers user privacy protection and authorized monitoring, has low storage overhead, and can achieve fine-grained control in every session on avoiding OM-MF.

VI. CONCLUSION

To balance the requirements of user privacy protection and government-authorized monitoring in instant messaging systems, this paper proposed a session key escrow scheme based on threshold cryptography and a new tripartite authenticated key agreement protocol. The proposed session key escrow scheme takes into account the security of both key agreement and key escrow, and, unlike other solutions, does not focus on the security of just one of them. To achieve authorization monitoring, the new scheme adopts a method that escrows the ephemeral private key of LEA instead of the secure master key s_{KGC} of KGC, and can therefore achieve fine-grained control in every session on avoiding the ''once monitor, monitor forever'' scenario and reduce the management difficulty and risk of being corrupted of s_{KGC}. In addition, LEA will generate and use only one pair E_{LEA} and T_{LEA} over the whole life of the IM system, which allows the proposed scheme to have low storage overhead for each KEA.

REFERENCES

[1] A. P. Oghuma, C. F. Libaque-Saenz, S. F. Wong, and Y. Chang, ''An expectation-confirmation model of continuance intention to use mobile instant messaging,'' *Telematics Informat.*, vol. 33, no. 1, pp. 34–47, 2016.

[2] S. Muftic, N. B. Abdullah, and I. Kounelis, ''Business information exchange system with security, privacy, and anonymity,'' *J. Elect. Comput. Eng.*, vol. 2016, Feb. 2016, Art. no. 7093642.

[3] Y. Long, Z. Cao, and K. Chen, ''A dynamic threshold commercial key escrow scheme based on conic,'' *Appl. Math. Comput.*, vol. 171, no. 2, pp. 972–982, 2005.

[4] Z. Yingli and L. Dan, ''A key escrow scheme to IOT based on Shamir,'' in *Proc. Int. Conf. Commun. Circuits Syst. (ICCCAS)*. Chengdu, China, Nov. 2013, pp. 94–97.

[5] J. Lin, W.-T. Zhu, Q. Wang, N. Zhang, J. Jing, and N. Gao, ''RIKE+ : Using revocable identities to support key escrow in public key infrastructures with flexibility,'' *IET Inf. Secur.*, vol. 9, no. 2, pp. 136–147, 2015.

[6] D. E. Denning and M. Smid, ''Key escrowing today,'' *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 58–68, Sep. 1994.

[7] A. Azfar, ''Implementation and performance of threshold cryptography for multiple escrow agents in VoIP,'' in *Proc. 1st Int. Joint Conf. Adv. Signal Process Inf. Technol.*, Amsterdam, The Netherlands, 2011, pp. 143–150.

[8] Q. Fan, M. Zhang, and Y. Zhang, ''Key escrow scheme with the cooperation mechanism of multiple escrow agents,'' *Przegląd Elektrotechniczny*, vol. 88, no. 5b, pp. 116–118, 2012.

[9] Q. Fan, X. C. Liu, H. Y. Liu, and T. M. Zhang, ''A key escrow scheme of the escrow agent with the denial right,'' *Inf. Technol. J.*, vol. 12, no. 16, pp. 3825–3830, 2013.

[10] A. Shamir, ''Identity-based cryptosystems and signature schemes,'' in *Advances in Cryptology*, vol. 1984. Berlin, Germany: Springe, 1985, pp. 47–53.

[11] S. Wang, Z. Cao, and X. L. Dong, ''Provably secure identity-based authenticated key agreement protocols in the standard model,'' *Chin. J. Comput.-Chin. Ed.*, vol. 30, no. 10, pp. 1842–1852, 2007.

[12] Z.-G. Gao and D.-G. Feng, ''Efficient identity-based authenticated key agreement protocol in the standard model,'' *J. Softw.*, vol. 22, no. 5, pp. 1031–1040, 2011.

[13] L. Ni, G. Chen, and J. Li, ''Escrowable identity-based authenticated key agreement protocol with strong security,'' *Comput. Math. With Appl.*, vol. 65, no. 9, pp. 1339–1349, 2013.

[14] M. Chen, ''Escrowable identity-based authenticated key agreement in the standard model,'' *Chin. J. Electron.*, vol. 43, no. 10, pp. 1954–1962, 2015.

[15] D. Abbasinezhad-Mood and M. Nikooghadam, ''An anonymous ECC-based self-certified key distribution scheme for the smart grid,'' *IEEE Trans. Ind. Electron.*, vol. 65, no. 10, pp. 7996–8004, Oct. 2018.

[16] D. Abbasinezhad-Mood, A. Ostad-Sharif, M. Nikooghadam, and S. M. Mazinani, ''A secure and efficient key establishment scheme for communications of smart meters and service providers in smart grid,'' *IEEE Trans. Ind. Informat.*, to be published. doi: 10.1109/TII.2019.2927512.

[17] A. Shamir, ''How to share a secret,'' *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[18] G. R. Blakley, ''Safeguarding cryptographic keys,'' in *Proc. Nat. Comput. Conf.*, 1979, pp. 313–317.

[19] T. B. Pedersen, K. Kaya, and E. Anarim, ''A CRT-based verifiable secret sharing scheme secure against unbounded adversaries,'' *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4416–4427, Nov. 2016.

[20] J. Zarepour-Ahmadabadi, M. Shiri-Ahmadabadi, A. Miri, and A. Lati, ''A new gradual secret sharing scheme with diverse access structure,'' *Wireless Pers. Commun.*, vol. 99, no. 3, pp. 1329–1344, 2018.

[21] H. Hong and Z. Sun, ''Achieving secure data access control and efficient key updating in mobile multimedia sensor networks,'' *Multimedia Tools Appl.*, vol. 77, no. 4, pp. 4477–4490, 2018.

[22] Z. Eslami, N. Pakniat, and M. Nojoumian, ''Ideal social secret sharing using Birkhoff interpolation method,'' *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 4973–4982, 2016.

[23] Y.-N. Liu, Q. Zhong, M. Xie, and Z.-B. Chen, ''A novel multiple-level secret image sharing scheme,'' *Multimedia Tools Appl.*, vol. 77, no. 10, pp. 6017–6031, 2017.

[24] Y. Cheng, Z. Fu, and B. Yu, ''Improved visual secret sharing scheme for QR code applications,'' *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2393–2403, Sep. 2018.

[25] S. Shivendra, ''Multi secret sharing with unexpanded meaningful shares,'' *Multimedia Tools Appl.*, vol. 77, no. 5, pp. 6287–6310, 2017.

[26] U. Ogiela and L. Ogiela, ''Linguistic techniques for cryptographic data sharing algorithms,'' *Concurrency Comput. Pract. Exper.*, vol. 30, no. 3, p. e4275, 2018. doi: 10.1002/cpe.4275.

[27] V. Attasena, J. Darmont, and N. Harbi, ''Secret sharing for cloud data security: A survey,'' *VLDB Int. J. Very Large Data Bases*, vol. 26, no. 5, pp. 657–681, 2017.

[28] Z. Wang, ''A privacy-preserving and accountable authentication protocol for IoT end-devices with weaker identity,'' *Future Gener. Comput. Syst.*, vol. 82, pp. 342–348, May 2018.

[29] S. Xu, G. Yang, Y. Mu, and R. H. Deng, ''Secure fine-grained access control and data sharing for dynamic groups in the cloud,'' *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2101–2113, Aug. 2018.

[30] A. Joux, ''A one round protocol for tripartite Diffie-Hellman,'' in *Proc. Int. Algorithmic Number Theory Symp.*, Leiden, The Netherlands, 2000, pp. 385–393.

[31] H. Xiong, Z. Chen, and F. Li, ''Provably secure and efficient certificateless authenticated tripartite key agreement protocol,'' *Math. Comput. Model.*, vol. 55, nos. 3–4, pp. 1213–1221, 2012.

[32] H. Xiong, Z. Chen, and F. Li, ''New identity-based three-party authenticated key agreement protocol with provable security,'' *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 927–932, 2013.

[33] Z. Tan, ''An efficient identity-based tripartite authenticated key agreement protocol,'' *Electron. Commerce Res.*, vol. 12, no. 4, pp. 505–518, 2012.

[34] M. Manulis, K. Suzuki, and B. Ustaoglu, ''Modeling leakage of ephemeral secrets in tripartite/group key exchange,'' *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 96, no. 1, pp. 101–110, 2013.

[35] M. Bayat and M. R. Aref, ''An attribute-based tripartite key agreement protocol,'' *Int. J. Communication Syst.*, vol. 28, no. 8, pp. 1419–1431, 2015.

[36] K. Suzuki and K. Yoneyama, ''Exposure-resilient one-round tripartite key exchange without random oracles,'' *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 97, no. 6, pp. 1345–1355, 2014.

[37] D. S. Gupta and G. P. Biswas, ''On securing Bi- and Tri-partite session key agreement protocol using IBE framework,'' *Wireless Pers. Commun.*, vol. 96, no. 3, pp. 4505–4524, 2017.

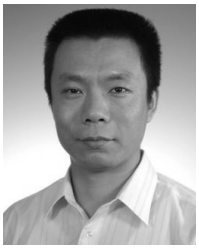
[38] H.-Y. Chien, ''Using the modified Diffie-Hellman problem to enhance client computational performance in a three-party authenticated key agreement,'' *Arabian J. Sci. Eng.*, vol. 43, no. 2, pp. 637–644, 2018.

[39] D. Johnson, A. Menezes, and S. Vanstone, ''The elliptic curve digital signature algorithm (ECDSA),'' *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.

[40] Z. Wang, Z. Ma, S. Luo, and H. Gao, ''Enhanced instant message security and privacy protection scheme for mobile social network systems,'' *IEEE Access*, vol. 6, pp. 13706–13715, 2018.

[41] (2013). *The Java Pairing Based Cryptography Library (JPBC)*. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc

ZHEN WANG was born in 1989. He is currently pursuing the Ph.D. degree with the School of Cyber Security, Beijing University of Posts and Telecommunications. His research interests include mobile network security and digital rights management.



SHOUSHAN LUO was born in 1962. He is currently a Professor and Ph.D. Supervisor with the School of Cyber Security, Beijing University of Posts and Telecommunications. His research interests include encode cryptography and networks, and information security.

ZHAOFENG MA was born in 1974. He received the Ph.D. degree from Xi'an Jiaotong University, in 2004. He did his postdoctoral research work in Tsinghua University from 2005 to 2007. Since 2007, he has been involved in education and research work with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include information security, digital rights management, and blockchain.

HONGMIN GAO was born in 1987. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing University of Posts and Telecommunications. His research interests include mobile network security and digital rights management.