Hindawi
Security and Communication Networks
Volume 2019, Article ID 7871067, 13 pages
https://doi.org/10.1155/2019/7871067

Research Article

A Provably Secure and Lightweight Identity-Based Two-Party Authenticated Key Agreement Protocol for Vehicular Ad Hoc Networks

Quanrun Li,[1,2] Ching-Fang Hsu,[1,3] Kim-Kwang Raymond Choo,[4] and Debiao He[3,5]

[1]*Information Security Lab, Computer School, Central China Normal University, Wuhan, China*
[2]*Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, China*
[3]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China*
[4]*Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, USA*
[5]*School of Cyber Science and Engineering, Wuhan University, Wuhan, China*

Correspondence should be addressed to Ching-Fang Hsu; cherryjingfang@gmail.com

Received 25 January 2019; Accepted 12 September 2019; Published 4 December 2019

Academic Editor: Bela Genge

As an important part of smart cities, vehicle ad hoc networks (VANETs) have attracted much attention from both industry and academia. In a VANET, generating a secure session key to facilitate subsequent data-in-transit transfer between two or more vehicles is crucial, which can be achieved by using an authenticated key agreement protocol. However, most of the existing identity-based two-party authenticated key agreement protocols have significant computational requirements or are known to be insecure. Thus, in this paper, a secure and efficient identity-based two-party authenticated key agreement protocol is presented by us. This protocol does not involve complex bilinear pairing computations and can generate a valid session key in two rounds. The security of the proposed protocol is proved in the eCK model which has better capability to describe a protocol's security than the famous CK model, and it has been widely used in the security proof of ID-based key agreement protocols currently. Additionally, we also evaluate its performance for potential utility in a VANET.

1. Introduction

As smart cities become a reality, vehicle ad hoc networks (VANETs) will become increasingly crucial. Therefore, data communications in a VANET are no longer restricted to a small number of vehicles, as such communications can occur among a wide range of vehicles (including driverless vehicles and unmanned aerial vehicles), roadside units (e.g., smart traffic lights), and other supporting infrastructure (e.g., IP-based CCTV). This allows the collection of traffic and other environmental information that can be analyzed to facilitate a smooth city operation. For example, information gathered from hurricane sensors and traffic monitoring devices can help to alert nearby vehicles to avoid a certain route.

In general, a typical VANET setup comprises a trusted authority, some roadside infrastructures and some smart vehicles. VANETs can provide connectivity among vehicles and other Internet-connected entities and devices (e.g., via other local networks or the Internet). For instance, it can realize efficient vehicle-to-vehicle communications in the Internet Transportation System (ITS) [1], and so on.

Two kinds of communication modes are included in a typical VANET (see also Figure 1), namely, vehicle-to-vehicle (V2V) communication and vehicle-to-infrastructure

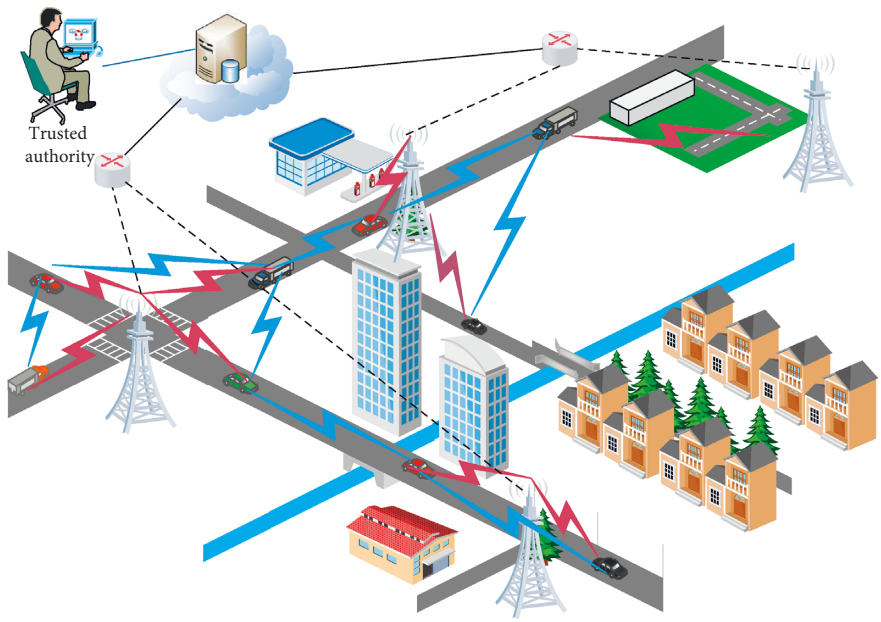

Figure 1: A typical VANET setup.

(V2I) communication. Since the increasing devices and electronic products around us are digitalized and Internet-connected, vehicle to everything(V2X) security has been an essential security attribute in our daily life [2].

In VANETs, communication channels between vehicles and nearby roadside infrastructures are usually established using dedicated short-range communication (DSRC) protocols [3]. By using these channels, a vehicle can transmit messages, such as traffic information or conditions, to nearby vehicles and roadside infrastructures at a uniform time period. Such information can be used by drivers to plan, revise, and optimize their routes. Depending on a city's connectivity level, local traffic control center (a trusted authority) may be able to reroute traffic, make certain adjustments to improve the traffic flow, and hence reduce traffic build up.

As more vehicles and devices join the network, there are operational challenges, for example, to deal with latency (e.g., communication delay) and minimize computational costs. It is known that computing capability of smart vehicles and roadside units usually is limited in comparison to other computationally powerful devices such as a dedicated laptop or server. In time-critical application such as VANETs within a smart city, a large volume of traffic and other related information may need to be handled in time for making accurate traffic decision and timely instructions. In addition, messages exchanged between the different entities (e.g., vehicles and/or devices) in the VANETs can be sensitive and private. Hence, security and privacy are both two key properties. However, due to the open nature of VANETs, an adversary can easily obtain sensitive user messages through various attacks such as replay, masquerading, impersonation, and password guessing. The leakage of such messages may have real-world consequences, such as facilitating the planning and execution of a kidnapping or assassination attempt.

Therefore, one fundamental design feature is to build a fast and secure communication channel between the different entities in a VANET, such as using two-party authenticated key agreement (2PAKA) protocols or group authenticated key agreement protocols. Specifically, in VANETs, a reliable 2PAKA protocol can help two communication entities to realize mutual authentication and get a valid session key. Unsurprisingly, a large number of 2PAKA protocols have been proposed to facilitate secure message exchange in VANETs. Simultaneously, such protocols are broadly divided into public key infrastructure (PKI)-based 2PAKA protocols, identity (ID)-based 2PAKA protocols, and certificateless 2PAKA protocols (i.e., based on how public keys are generated in these protocols). One limitation associated with PKI-based protocols is the surprising cost incurred in maintaining, issuing, and authenticating a large number of certificates. To mitigate such a limitation, we could use 2PAKA protocol based on identity-based cryptography (IBC) [4–11]. While ID-based 2PAKA protocols, such as those presented in [5–7], could overcome certain shortcomings associated with PKI-based 2PAKA protocols, bilinear pairing used in these protocols makes them unrealistic for deployment on lightweight devices. Hence, to overcome inefficiency caused by bilinear pairing, Zhu et al. [12] presented an ID-based 2PAKA protocol including no pairings in 2007. Nevertheless, the protocol suffers from limitations, such as the requirement for significant bandwidth. In recent times, we can find that a large number of similar protocols have been designed in the literature. However, most of these protocols provide no security proof, use a weak model to prove safety, take more than two communication rounds, or are found to be insecure. Based on gap Diffie–Hellman assumption, for example, Dang et al. [13] designed a two-round ID-based 2PAKA protocol in 2018, which has security proof in the eCK model. However, we reveal in this paper that their protocol could suffer from the man-in-the-middle attack, contrary to their claim. In our paper, we will build on their work and introduce a two-round

ID-based 2PAKA protocol. We demonstrate that our protocol requires less computation and communication costs, in comparison to the protocol of Dang et al. [13].

The key properties of our proposed protocol are summarized as below:

(1) Mutual authentication of the two parties and negotiation of the session key can be realized by our ID-based two-party AKA protocol.

(2) We show our protocol can get strong security in the eCK model, unlike most other existing protocols.

(3) The proposed protocol is two-round and pairing-free. Hence, it is more superior to other competing protocols in terms of performance.

The rest of this paper is organized as follows. Some related works on ID-based 2PAKA protocol and background materials (i.e., mathematical assumptions and security attributes relating to ID-based AKA protocols) are introduced in Sections 2 and 3, respectively. Our new ID-based 2PAKA protocol is shown in Section 4. In Sections 5 and 6, we demonstrate the security of the protocol in the eCK model and give out the corresponding performance analysis. A comparative summary of the performance between the proposed protocol and the ID-based 2PAKA protocols of [13, 14] is also presented in Section 6. In the end, the last section shows our paper's conclusion.

2. Related Work

This section mainly shows some related works on the ID-based two-party AKA protocols. At first, we divide these protocols into two types: ID-based 2PAKA protocols with pairings and ID-based 2PAKA protocols without bilinear functions [15, 16]. Next, we respectively review the related works about the two different types of protocols:

2.1. ID-Based 2PAKA Protocols with Pairings. The first key agreement protocol employing pairings was presented by Joux [17] in 2000. Then Boneh and Franklin used bilinear pairing to construct the first ID-based encryption scheme in 2001 [18]. After Boneh and Franklin's work, a lot of ID-based authenticated key agreement protocols with pairings have been presented. According to this ID-based encryption scheme, the first ID-based 2PAKA protocol with pairings was presented by Smart [4]. Unfortunately, Shim [19] found that the protocol presented by Smart [4] had some security flaws and constructed another one ID-based 2PAKA protocol with stronger security, which had lesser quantity of bilinear pairings. In Shim's protocol [19], only one Weil pairing and scalar multiplication were used in the computation of session key. Meanwhile, Shim declared that his protocol could resist the general attacks. However, the protocol of Shim [20] was shown that it suffers from man-in-the-middle attack in the paper of Hsieh [19].

2.2. ID-Based 2PAKA Protocols without Bilinear Pairing Operations. To eliminate efficiency flaw in ID-based 2PAKA protocols with pairings, all kinds of ID-based 2PAKA protocols using no bilinear functions have been presented in the last decade. In 2007, the first ID-based 2PAKA protocol using no bilinear operations was presented by Zhu et al. [12] based on an ID-based signature scheme. Nevertheless, their protocol was still inefficient and needed three message exchanges. To reduce communication traffic, Fiore and Gennaro [21] used exponentiation operation to make an ID-based 2PAKA protocol in 2010. Besides, this protocol's security was proved by them in the CK model. But this weak security model could not describe the ability of real adversary well. In the same year, Cao et al. [22] proposed a new ID-based 2PAKA protocol employing no pairings to reduce message exchange. Unfortunately, Cao et al.'s protocol was vulnerable to ephemeral key revealed attack. After Cao et al.'s work, lots of ID-based 2PAKA protocols using no bilinear functions were proposed, but these protocols still could not deal with the efficiency problem and security issue effectively.

But because ID-based 2PAKA protocol without pairings can fit real-time application environment such as VANETs perfectly, cryptologists still put a lot of effort into improving these protocols' performance and security. Until recently, some responding protocols with better properties have been presented. In 2015, Sun et al. [23] presented an improved 2PAKA protocol based on the identity with security proof in the eCK security model. But disadvantages were that this protocol used six scalar multiplications on elliptic curve and security proof was incomplete because only passive adversary was taken into consideration in the security model. After the Sun et al.'s work, Ni et al. [24] designed other new ID-based 2PAKA protocol that only needed five scalar multiplications in 2016. In addition, it was proved secure in the eCK security model completely. Although this protocol was far more efficient than previously proposed protocols, the communication traffic was still very large. Then, in 2017, an ID-based 2PAKA protocol including no pairings based on the BAN logic model was constructed by Islam and Biswas [25]. Sadly, their protocol was unsafe.

3. Preliminaries

In this section, we show several difficult mathematical problems and indispensable security attributes in the ID-based AKA protocol.

3.1. Mathematical Assumptions. The following difficult mathematical problems are some basic tools used to analyze the security of AKA protocol.

We assume that q is the order for a finite cyclic additive group $\mathbb{G}$, where q is a big prime number. Meanwhile, $\mathbb{G}$ has a generator P.

3.1.1. Elliptic Curve Discrete Logarithm (ECDL) Problem. Given two elements $P, Q \in \mathbb{G}$, it is hard to calculate a value $a \in \mathbb{Z}_p^*$ such that $Q = aP$ for any adversary in probability polynomial time.

3.1.2. Gap Diffie–Hellman (GDH) Problem. Given three points (P, aP, bP) and a DDH oracle, for any probability polynomial time algorithm, the advantage of making the calculation of abP can be ignored, where $a, b \in \mathbb{Z}_p^*$ are unknown.

3.1.3. Decisional Diffie–Hellman (DDH) Problem. For unknown $a, b, c \in \mathbb{Z}_p^*$, if an adversary is given $P, aP, bP, cP \in \mathbb{G}$, it is still difficult to determine whether $c = ab \pmod q$ or not.

3.1.4. Computational Diffie–Hellman (CDH) Problem. The numbers $a, b \in \mathbb{Z}_p^*$ are unknown. Knowing three points $P, aP, bP \in \mathbb{G}$, it is also impossible to compute abP by any PTT algorithm.

3.2. Essential Security Attributes. If an AKA protocol is safe and reliable, it must have some essential security attributes, because these security attributes show that the proposed protocol is capable of resisting corresponding attacks. So security attribute is an important index to measure quality of a protocol. The following security attributes are some basic conditions that a secure ID-based 2PAKA protocol needs to meet [26–28].

3.2.1. Known-Key Security (K-SKS). Even though the adversary has known a protocol's all previous session keys, this protocol can still keep the current session key secure.

3.2.2. Forward Secrecy (FS). The leakage of users' long-term private keys has no impact on the security of preceding session keys. Generally, forward secrecy mainly includes the following two different categories:

(1) *Partial forward secrecy*. Even though the adversary has known some users' long-term secret keys, session keys made in preceding sessions still can keep safe.

(2) *Perfect forward secrecy*. For any probability polynomial time adversary, learning all long-term secrets has little help to make session keys known.

3.2.3. Key Compromise Impersonation (KCI) Resistance. Even though an adversary knows entity A's long-term private key, he still cannot masquerade as any other user to A.

3.2.4. Unknown Key Share (UKS) Resistance. An adversary makes a group of users believe that they are sharing a secret with him. Actually, this secret should be shared by them and another user (e.g., B holds the viewpoint that a session key is established by itself and an adversary E. In fact, this key is generated by A and B together).

3.2.5. No Key Control (NKC). No entity can enforce a session key to be preselected or predetermined.

3.2.6. Basic Impersonation (BI) Resistance. If a party A's long-term secret key is leaked to an adversary, he can make full use of this key to disguise himself as A.

3.2.7. Ephemeral Key Reveal (EKR) Resistance. Even if an adversary acquires the ephemeral private keys of all participants in a session, the session key is kept private as before.

4. Our Presented Protocol

We describe our ID-based two-round 2PAKA protocol Π without bilinear pairings in this section. There are three main algorithms included in this protocol Π, namely, setup algorithm, key generation algorithm, and key agreement algorithm. In our protocol, it is worth noting that the trusted authority plays the role of KGC.

4.1. Setup Phase. At this stage, all vital system parameters are generated by the trusted authority performing this setup algorithm with security parameter k. The specific implementation steps are as follows:

(1) Chooses one additive group $\mathbb{G}$ with a generator P. Meanwhile, p is a prime order of $\mathbb{G}$

(2) Selects randomly a number $s \in \mathbb{Z}_p^*$ as KGC's private key, and then calculates $P_{\text{pub}} = sP$ as its public key

(3) Picks three high-efficiency hash functions, where $h_1 : \{0,1\}^* \times \mathbb{G} \longrightarrow \mathbb{Z}_p^*$, $h_2 : \mathbb{G} \longrightarrow \mathbb{Z}_p^*$, and $H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \longrightarrow \{0,1\}^k$

(4) Makes $\text{params} = \{\mathbb{G}, p, P, P_{\text{pub}}, h_1, h_2, H_3\}$ public information and preserves the confidentiality of key s

4.2. Extract Key Phase. Here, we always suppose every vehicle has its own $\text{ID}_i \in \{0,1\}^*$. The trusted authority (acts as KGC) uses the Schnorr signature algorithm to compute these vehicles' long-term private keys. Besides, the trusted authority distributes these key pairs to the corresponding vehicles. The trusted authority does as below:

(1) Picks randomly a number $r_i \in \mathbb{Z}_p^*$ for each vehicle, and then does calculations about $R_i = r_i P$ and $h_i = h_1(\text{ID}_i, R_i)$.

(2) Calculates $s_i = r_i + h_i s \pmod p$ and a vehicle's long-term secret key actually is (s_i, R_i).

(3) The corresponding vehicle could receive this pair (s_i, R_i) sent by using a secure channel. Then, the vehicle can check the validity of its long-term private key after receiving the pair because they can verify whether the equation $s_i P = R_i + h_1(\text{ID}_i, R_i) P_{\text{pub}}$ is satisfied or not. If it passes, the vehicle sets $\text{PK}_i = s_i P$ as its long-term public key.

4.3. Key Agreement Phase. After the extract key phase, two vehicles A and B have their own key pair (s_i, R_i) and relevant ID_i. Now, A and B want to establish a session key through mutual communication, which is used to keep latter data secure. Our new protocol is displayed in Figure 2.

(1) Firstly, the ephemeral private key $a \in \mathbb{Z}_p^*$ of vehicle A is randomly chosen, and its ephemeral public key is set as $T_A = aP$. Then, A sends ID_A, R_A, and T_A to vehicle B:

$$A \longrightarrow B : ID_A, R_A, T_A. \quad (1)$$

(2) Meanwhile, a random value $b \in \mathbb{Z}_p^*$ is selected as vehicle B's ephemeral private key, similarly. Then, B also computes $T_B = bP$ as the ephemeral public key. B transmits ID_B, R_B, and T_B to A:

$$B \longrightarrow A : ID_B, R_B, T_B. \quad (2)$$

(3) After receiving messages from B, A can calculate $PK_B = R_B + h_1(ID_B, R_B)P_{pub}$ and $SK_A = h_2(s_A PK_B)aT_B$. So, A can compute the session key $sk_{AB} = H_3(ID_A, ID_B, SK_A, aPK_B, s_A T_B)$.

(4) In the same way, when B gets data that A sends, vehicle B can compute $PK_A = R_A + h_1(ID_A, R_A)P_{pub}$ and $SK_B = h_2(s_B PK_A)bT_A$. B can also calculate the session key $sk_{BA} = H_3(ID_A, ID_B, SK_B, bPK_A, s_B T_A)$.

Correctness. The correctness of our protocol is shown as below:

$$\begin{aligned} SK_A &= h_2(s_A PK_B)aT_B \\ &= h_2(s_A \cdot s_B \cdot P)a \cdot b \cdot P \\ &= h_2(s_B PK_A)bT_A \\ &= SK_B, \\ aPK_B &= a \cdot s_B \cdot P = s_B T_A, \\ s_A T_B &= s_A \cdot b \cdot P = bPK_A. \end{aligned} \quad (3)$$

5. Security Analysis

In the following content, the security of our protocol Π is displayed in detail. Firstly, it is necessary to give out the eCK security model that we use in our protocol. After that, we give the security proof and some security attributes of our protocol in detail.

5.1. Protocol Participants. There is a set $\mathcal{U}$ that is composed of all protocol participants. Every party in set $\mathcal{U}$ has a unique ID_i and corresponding private and public key pair (s_i, PK_i). Besides, PK_i is relative with its ID_i and s_i always is generated by the trusted authority (acts as KGC). In security proof, the ability of each participant is usually described by a probability polynomial time (PPT) algorithm. We consider that a polynomial number of sessions are the maximum value that every participant can take part in at the same time. Furthermore, the sth session of party ID_A is denoted as $\Pi_{A,B}^s$ that is established by party ID_A and party ID_B. If A finally gets a valid session key by communicating with B, we think participant A completes the session $\Pi_{A,B}^s$.

5.2. eCK Model. Without loss of generality, an adversary C always is deemed as a PPT algorithm in the security model. Moreover, C is considered to have the ability to control the whole communication network. It means messages may be arbitrarily replayed, eavesdropped, modified, suspended, and injected by the adversary. The ability of an adversary C always is described through a series of queries. Here, we only give out the simple information about the eCK model, and more details can be found in the literature of Huang and Cao [29].

(i) *EphemeraKeyReveal*($\Pi_{A,B}^s$). Adversary C can get the ephemeral private key of protocol participant ID_A in session $\Pi_{A,B}^s$.

(ii) *SessionKeyReveal*($\Pi_{A,B}^s$). If session $\Pi_{A,B}^s$ is completed, adversary C can obtain the session key. Otherwise, a null value will be returned to C.

(iii) *StaticKeyReveal*(ID_A). Adversary C could obtain ID_A's long-term private key sent by this query. But C cannot control ID_A completely.

(iv) *PKGstaticKeyReveal*. The adversary can know PKG's master private key by this query. This query usually simulates PKG-FS.

(v) *EstablishParty*(ID_A). After requesting this query, a legitimate user ID_A can be registered. But the adversary can acquire ID_A's private key. In addition, party ID_A is considered to be dishonest.

(vi) *Send*($\Pi_{A,B}^s$, M). A message M is transmitted to $\Pi_{A,B}^s$ by adversary C. After receiving this message, $\Pi_{A,B}^s$ responds to the message according to the protocol regulation. If M is null, this session $\Pi_{A,B}^s$ will initiate a session as an initiator. Otherwise, it will be a responder.

(vii) *Test*($\Pi_{A,B}^s$). This query usually takes place during the experiment. The adversary C can make this query to a finished fresh session $\Pi_{A,B}^s$ only once. The session picks a random value $b \in \{0, 1\}$ when getting a test query from the adversary C. If $b = 1$, this session $\Pi_{A,B}^s$ returns session key to C. Otherwise, a random number that is indistinguishable from the session key will be given to adversary C.

Definition 1 (matching session). If the sessions $\Pi_{A,B}^s$ and $\Pi_{B,A}^t$ have the same session identifier *SID*, they will be each other's matching session. *SID* is a series connection of a session participant's messages in the order of initiator or responder.

Definition 2 (fresh session). The $\Pi_{A,B}^s$ is a fresh session executed by two honest parties ID_A and ID_B, if none of the following conditions is satisfied.

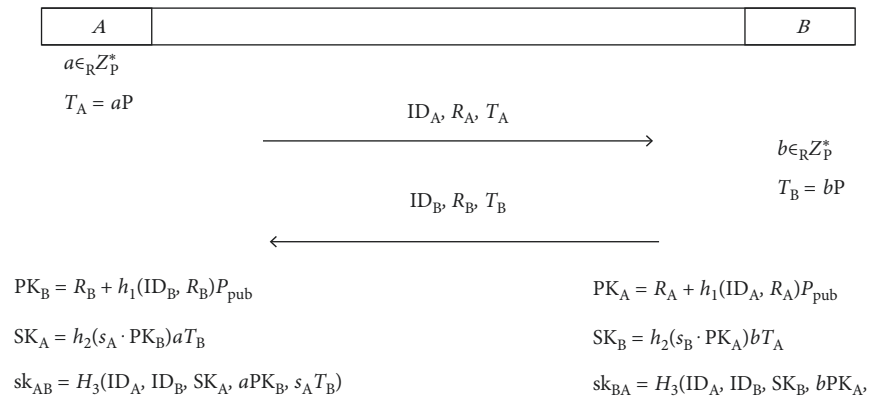

Figure 2: The key agreement phase.

(1) The session $\Pi^s_{A,B}$ has a matching session $\Pi^t_{B,A}$. Additionally, a session key of $\Pi^s_{A,B}$ or $\Pi^t_{B,A}$ is leaked to adversary C.

(2) Assume that session $\Pi^s_{A,B}$ has a matching session $\Pi^t_{B,A}$. Besides, all of ID_A's secret keys in $\Pi^s_{A,B}$ or the long-term and temporary secret keys of ID_B in $\Pi^t_{B,A}$ are revealed to adversary C.

(3) There is no matching session about $\Pi^s_{A,B}$, but an adversary knows the long-term and ephemeral private keys of ID_A in $\Pi^s_{A,B}$ or the long-term secret key of ID_B.

Definition 3 (ID-eCK security). For adversary C, **succ** denotes the event that adversary C can know b's correct value in the manner of sending a **Test** query to some fresh instances. Therefore, the advantage of adversary C is $\mathbf{adv}^{AKE}(C) = |2\mathbf{Pr}[\mathbf{succ}] - 1|$. If a 2PAKA protocol can satisfy the following conditions, this protocol is considered to be ID-eCK security.

(1) For any PPT adversary, the success probability of knowing the right b is negligible. It means that $\mathbf{adv}^{AKE}(C) = |2\mathbf{Pr}[\mathbf{succ}] - 1|$ is a negligible value.

(2) After two honest parties complete a session, they can get the same session key.

5.3. Formal Proof. Firstly, our protocol will be proved to have strong enough security in this section. Then, we will show some security attributes that our protocol has.

Theorem 1. *Assume H_1, H_2, and H_3 are three random oracles in the eCK model. According to the difficult GDH problem, no adversary can break the security of protocol Π in polynomial time.*

Proof. On the basis of ID-eCK security, we know the two properties in Definition3 are the basic conditions that a secure AKA protocol should satisfy. We can know the second condition is met in the correctness of our protocol. Next, we show that the first condition also can be met in the following content. In the eCK security model, our protocol Π's security parameter is specified as k. Meanwhile, the maximum value of truthful users activated by an adversary is $n(k)$. The symbol $q_s(k)$ is the maximum number of sessions that each party can take part in. Besides, it is also an assumed condition that the test session selected by adversary C is $\Pi^s_{A,B}$, which is established by ID_A and ID_B together. Adversary C can only use the following three methods to get the correct value of test session key.

(1) *Guessing directly attack*. Adversary C can know test session key in a guessing way.

(2) *Key replication attack*. Adversary C constructs another session that is not a matching session of the test session. But its session key is same with the test session. Hence, adversary C can make the use of a nonmatching session to acquire the test session key. Namely, it means that this protocol cannot stand up to adversary C's attack.

(3) *Forging attack*. Adversary C can query H_3 random oracle with the input $(ID_A, ID_B, T_A, T_B, SK_A, aPK_B,$ and $s_A T_B)$. Obviously, the adversary C calculates the value $(SK_A, aPK_B,$ and $s_A T_B)$ by itself.

Because k is security parameter, it indicates that the size of session key is k bit. Therefore, the guessing directly attack is successful with probability $\mathcal{O}(1/2^k)$. Besides, the role of H_3 is the same as a random oracle in the security model. If this random oracle produces no collisions, the event that a session key is jointly owned by two nonmatching sessions occurs with a negligible probability because two nonmatching sessions cannot have the same SID under the definition of matching session. Namely, key replication attack's successful probability is negligible. Consequently, we only need to consider about the successful probability of the forging attack.

Before we analyze a forging attack in detail, we firstly review the GDH problem. This mathematical assumption is that the value $(X = xP, Y = yP)$ is given, where x, $y \in_R \mathbb{Z}_p^*$ is unknown, the aim of challenger S is to get the result of $GDH(X, Y) = xyP$ by using a DDH oracle. Then, a challenger S plays the ID-eCK game with the adversary C who can break the protocol Π. During the game, S must make

responses to all kinds of queries of the adversary C. If the adversary C can make a successful forging attack with non-negligible probability, the challenger S can construct a gap Diffie–Hellman solver by using C as a subroutine. As fresh definition shown in the eCK model, we need to consider about two special cases:

(1) The session $\Pi^t_{B,A}$ is the matching session of the test session $\Pi^s_{A,B}$

(2) There is no corresponding session matching with the session $\Pi^s_{A,B}$

In the first scenario, we only consider about passive adversary who cannot change messages transmitted among all parties. Contrary to the first scenario, challenger S has an active adversary that could modify the party ID_B's long-term secret key element R_B in the second scenario. Above analysis results show that the adversary can adopt different attack strategies. So before the challenger S plays game with the adversary C, S can ensure C's test session is $\Pi^s_{A,B}$ with probability $1/n(k)^2 q_s(k)$. In addition, S must guess the attack way that the adversary could choose from the following six strategies:

(i) *Case 1*. ID_B's long-term secret key and ID_A's temporary private key are not leaked to adversary C. Meanwhile, C transmits R_B correctly.

(ii) *Case 2*. The long-term secret keys of ID_A and ID_B are not known by C. The value of R_B still is not modified by C.

(iii) *Case 3*. The temporary private keys of ID_A and ID_B are not revealed to adversary C.

(iv) *Case 4*. ID_B's temporary private key and the long-term secret key of ID_A are not acquired by adversary C.

(v) *Case 5*. Adversary C does not get the long-term secret key of ID_B and ID_A's temporary private key. But the value of R_B is changed by C.

(vi) *Case 6*. Adversary C knows nothing about long-term secret keys of ID_A and ID_B. But C alters R_B's real value.

Obviously, the six cases above cover all attack manners of different adversaries, including the passive adversary and the active adversary. On the basis of the above result, the correct test session and strategy are chosen by challenger S with the probability $1/6n(k)^2 q_s(k)$.

Case 1. ID_B's long-term secret key and ID_A's temporary private key are not leaked to adversary C. Meanwhile, C correctly transmits R_B.

(i) *Setup*. The challenger S initializes the long-term keys of all parties and KGC's public key as follows:

(1) The challenger S chooses the value $P_{pub} \in \mathbb{G}$ at random as the KGC's public key.

(2) Challenger S randomly chooses $h_B \in \mathbb{Z}_p^*$ as $H_1(ID_B, R_B)$ and calculates $R_B = Y - h_B P_{pub}$. So we can know $PK_B = R_B + h_B P_{pub} = Y$ is the long-term public key of ID_B. ID_B can get its long-term secret key's value (Δ, R_B).

(3) For other parties ID_i, the challenger S randomly selects $h_i, s_i \in \mathbb{Z}_p^*$ as $H_1(ID_i, R_i)$ and long-term private key. Similarly, the challenger can compute $R_i = s_i P - h_i P_{pub}$. Therefore, $PK_i = R_i + h_i P_{pub} = s_i P$ is ID_i's long-term public key.

After the above process, for each ID_i, the challenger passes (ID_i, R_i) to the adversary C and this new entry (ID_i, R_i, h_i) is added to H_1^{list}.

(ii) *Queries*. In order to deal with H_1, H_2, H_3, and SessionKeyReveal queries from C, the challenger first maintains the corresponding empty lists H_1^{list}, H_2^{list}, H_3^{list}, and R^{list}. Then, the challenger S responds to all queries from C as below:

(1) $H_1(ID_i, R_i)$. In the setup phase, when the long-term secret key of each party ID_i is set, S inserts the entry (ID_i, R_i, h_i) to H_1^{list}. When a query about H_1 sent by adversary C already exists in H_1^{list}, S returns the corresponding entry to C. Otherwise, S selects randomly $h_i \in \mathbb{Z}_p^*$ and adds the new entry (ID_i, R_i, h_i) to the list H_1^{list}. Then, S returns the random value h_i.

(2) $H_2(s_i PK_j)$. When C launches an H_2 query, S first searches for the relevant entry in the whole list H_2^{list}. If S finds the entry out, S transmits h_2 to C. Conversely, S computes $s_i * PK_j$ and checks whether this value is already in H_2^{list} or not. If the H_2^{list} has the corresponding value, the challenger gives h_2 to C. Otherwise, S chooses $h_2 \in \mathbb{Z}_p^*$ at random. Then, S inserts the entry $(s_i, PK_j, s_i PK_j, h_2)$ to H_2^{list} and returns h_2 to the adversary C.

(3) $H_3(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j)$. Before this query, S keeps an empty table H_3^{list} whose entries are the form of $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j, h_3)$.

(i) If the corresponding entry is found in H_3^{list}, the challenger S responds to the query with h_3.

(ii) Otherwise, S checks the whole list R^{list}. If i has a correct value B and target item is stored, S uses the DDH oracle to verify whether $DDH(H_2(s_j PK_i) T_j, T_i, SK) = DDH(h_2 T_j, T_i, SK) = 1$, $DDH(T_i, PK_j, a_i PK_j) = 1$, and $DDH(PK_i, T_j, s_i T_j) = 1$. If all of them are right, S sets $h_3 = sk_{ij}$ and stores the entry $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j, h_3)$ to H_3^{list}. If the entry is found in the list and $i \neq B$, the equation $h_3 = sk_{ij}$ is made by challenger S and the relevant item is added to H_3^{list}. But if the list H_3^{list} has not the corresponding entry or the verifications of DDH oracle are wrong, S randomly selects $h_3 \in \{0,1\}^k$ and writes these new data into the list H_3^{list}. In final, S returns the corresponding h_3 to the adversary C.

(4) *EphemeralKeyReveal*$(\Pi^s_{i,j})$. If the session $\Pi^s_{i,j}$ is $\Pi^s_{A,B}$, the challenger S aborts. Otherwise, the

temporary private key of ID_i is sent to S by adversary C.

(5) *StaticKeyReveal*(ID_i). If $i = B$, S aborts. Otherwise, the challenger provides C with (s_i, R_i).

(6) *MasterPrivateKeyReveal*(). The challenger S aborts.

(7) *EstablishParty*(ID_i). For this query, the challenger S chooses s_i, $h_i \in \mathbb{Z}_p^*$ at random. Then S assigns the value of $H_1(ID_i, R_i)$ to h_i and calculates $R_i = s_i P - h_i P_{pub}$. Finally, S sends (s_i, R_i) to C as its long-term secret key. Therefore, the adversary C can control ID_i completely, it is because the long-term secret key of ID_i is known by C.

(8) *SessionKeyReveal*($\Pi_{i,j}^s$). If the session $\Pi_{i,j}^s$ is $\Pi_{A,B}^s$ or $\Pi_{B,A}^t$, S aborts. Otherwise, S searches for the value s_{ij} in the whole list R^{list} and sends it to C.

(9) *Send*($\Pi_{i,j}^s$, M). A blank table R^{list} is held by S whose element is $(ID_i, ID_j, T_i, T_j, sk_{ij})$ for the Send query.

(i) If $\Pi_{i,j}^s$ is $\Pi_{A,B}^s$ and $M = \Delta$, adversary C receives X returned by challenger S.

(ii) If the value of i is B and $M = W$, S selects $b \in \mathbb{Z}_p^*$ at random and returns the bP to C. Besides, S verifies whether $DDH(H_2(s_j PK_i) T_j, T_i, SK) = DDH(h_2 T_j, bP, SK) = 1$, $DDH(T_i, PK_j, a_i PK_j) = 1$, and $DDH(PK_i, T_j, s_i T_j) = 1$. If SK, $a_i PK_j$, and $s_i T_j$ are correct, S sets $sk_{ij} = h_3$ and inserts the entry $(ID_i, ID_j, T_i, T_j, sk_{ij})$ to the list R^{list}. But if the list H_3^{list} does not have the entry or one of SK, $a_i PK_j$, and $s_i T_j$ is wrong, S chooses randomly $sk_{ij} \in \{0,1\}^k$ and writes new information to R^{list}.

(iii) If B is not the correct value for i, S answers this Send query according to protocol rule.

(10) *Test*($\Pi_{i,j}^s$). If the session $\Pi_{i,j}^s$ is $\Pi_{A,B}^s$, S chooses randomly $\beta \in \{0,1\}^k$ and this data is returned to adversary C. On the contrary, S is not playing this game.

(iii) *Analysis.* If a forgery attack is successfully launched by adversary C with great probability, C must have used $SK = H_2(s_A Y) DLOG(X) bP$, $a_i PK_B = DLOG(X) Y$, and $s_A T_B = s_A bP$ to query H_3 random oracle. To cope with the $GDH(X, Y)$ difficult problem, challenger S checks whether the value of an H_3 query from C is $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j)$ such that $DDH(H_2(s_j PK_i) T_j, T_i, SK) = DDH(h_2 T_j, bP, SK) = 1$, $DDH(T_i, PK_j, a_i PK_j) = 1$, and $DDH(PK_i, T_j, s_i T_j) = 1$. If this H_3 query is found, S computes $GDH(X, Y) = DLOG(X) Y = a_A PK_B$ using this query. Assume that the probability of the event that a forgery attack is made by adversary C is $\mathbf{Adv}_C^{\Pi}(k)$, so S successfully deals with the GDH problems with the advantage

$$\mathbf{Adv}_S^{GDH}(k) \geq \frac{\mathbf{Adv}_C^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{4}$$

Case 2. The long-term secret keys of ID_A and ID_B are not known by C. The value of R_B still is not modified by C.

(i) *Setup.* All parties' long-term secret keys and KGC's public key are given by challenger C as follows:

(1) $P_{pub} \in \mathbb{G}$ selected by the challenger S is assigned to KGC's public key.

(2) As for ID_A, a random value $h_A \in \mathbb{Z}_p^*$ is selected by S as the value of $H_1(ID_A, R_A)$. Then, S does the calculation on $R_A = X - h_A P_{pub}$ and ID_A's long-term secret key is given the value (Δ, R_A). Thus, X is relevant public key of ID_A.

(3) Using the same method, S selects $h_B \in \mathbb{Z}_p^*$ at random as $H_1(ID_B, R_B)$ for the party ID_B and calculates $R_B = Y - h_B P_{pub}$. So ID_B's long-term secret key can be assigned to the value (Δ, R_B). Additionally, ID_B's corresponding public key is $PK_B = R_B + h_B P_{pub} = Y$.

(4) Considering about other parties ID_i, where $i \neq A$ and $i \neq B$, challenger S randomly chooses h_i, $s_i \in \mathbb{Z}_p^*$. Similarly, S sets $h_i = H_1(ID_i, R_i)$ and computes the relevant value of $R_i = s_i P - h_i P_{pub}$. Thus, the long-term secret key of entry ID_i is (s_i, R_i). Its public key is $PK_i = R_i + h_i P_{pub}$. S sends (ID_i, R_i) to the adversary C, and this new entry (ID_i, R_i, h_i) is inserted into the table H_1^{list}.

(ii) *Queries.* To deal with the query about SessionKeyReveal and three hash queries H_1, H_2, and H_3, challenger S stores four tables R^{list} and H_1^{list}, H_2^{list}, H_3^{list}. And S uses the following ways to answer those queries asked by C.

(1) $H_3(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j)$. S has an empty list H_3^{list} in the form of $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j, h_3)$.

(i) If H_3^{list} already has the relevant entry $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j, h_3)$, S returns h_3 to the adversary C.

(ii) If not, S looks up target item in the whole table R^{list}. If the item is found and the value of i is A or B, challenger S verifies whether $DDH(h_2 T_i, T_j, SK) = DDH(H_2(DLOG(X) Y) T_i, T_j, SK) = 1$, $DDH(T_i, PK_j, a_i PK_j) = 1$, and $DDH(PK_i, T_j, s_i T_j) = 1$. If all of them are correct, S sets $h_3 = sk_{ij}$ and stores the new entry into the list H_3^{list}. If R^{list} has the goal entry (A and B are both not the right value of i), S assigns h_3 to sk_{ij}. Then, S adds the new entry to the list H_3^{list}. Otherwise, If the corresponding entry does not exist in the list R^{list} or SK, $a_i PK_j$ and $s_i T_j$ are not right, S chooses $h_3 \in \{0,1\}^k$ and inserts $(ID_i, ID_j, T_i, T_j, SK, a_i PK_j, s_i T_j, h_3)$ into the list H_3^{list}.

(2) *EphemeralKeyReveal*($\Pi_{i,j}^s$). Adversary C acquires temporary secret key of entry ID_i returned by S.

(3) *StaticKeyReveal*(ID_i). If A or B is the correct value of i, challenger S terminates this program. Otherwise, (s_i, R_i) is transmitted to C.

(4) *Send*($\Pi_{i,j}^s$, M). As before, a blank table R^{list} is held by S, the form of which is $(ID_i, ID_j, T_i, T_j, sk_{ij})$.

(i) If i is equal to A, its temporary secret key is $a \in \mathbb{Z}_p^*$ picked up by challenger S at random and the value aP is given to C. Next, S seeks the relevant item in the table H_3^{list}. If the entry exists, S checks whether $DDH(h_2T_i, T_j, \text{SK}) = DDH(H_2(\text{DLOG}(X)Y)T_i, T_j, \text{SK}) = 1$, $DDH(T_i, \text{PK}_j, a_i\text{PK}_j) = 1$, and $DDH(\text{PK}_i, T_j, s_iT_j) = 1$. If all of them are right, S sets $\text{sk}_{ij} = h_3$ and stores the new entry to R^{list}. But if this corresponding item is not found or three values verified by DDH oracle are not correct, S randomly selects $\text{sk}_{ij} \in \{0,1\}^k$ and writes new data to R^{list}.

(ii) If i is equal to B, S uses a similar way in the simulation.

(iii) For other conditions, the challenger S responds according to the protocol specification.

It is worth noting that S responds to the $H_1(\text{ID}_i, R_i)$, $H_2(s_i\text{PK}_j)$, EstablishParty(ID_i), MasterPrivateKey, Session KeyReveal($\Pi_{i,j}^s$), and Test($\Pi_{i,j}^s$) in the manner of case 1.

(iii) *Analysis.* Similarly, if the adversary C makes a successful forging attack with non-negligible probability $\mathbf{Adv}_{\text{C}}^{\Pi}$, C must make the use of $\text{SK} = H_2(\text{DLOG}(X)Y)abP$, $a_{\text{A}}\text{PK}_{\text{B}} = aY$, and $s_{\text{A}}T_{\text{B}} = bX$ to query H_3. To deal with $GDH(X,Y)$, S checks whether the content of an H_3 query from C is $\text{ID}_{\text{A}}, \text{ID}_{\text{B}}, T_{\text{A}}, T_{\text{B}}, \text{SK}, aY, bX$ such that $DDH(h_2T_i, T_j, \text{SK}) = DDH(H_2(\text{DLOG}(X)Y)T_i, T_j, \text{SK}) = 1$, $DDH(T_i, \text{PK}_j, a_i\text{PK}_j) = 1$, and $DDH(\text{PK}_i, T_j, s_iT_j) = 1$. If S can find such an H_3 query, it computes $GDH(X,Y) = \text{DLOG}(X)Y$. Therefore, GDH difficult problem can be solved by S successfully with the advantage

$$\mathbf{Adv}_{\text{S}}^{GDH}(k) \geq \frac{\mathbf{Adv}_{\text{C}}^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{5}$$

Case 3. The temporary private keys of ID_{A} and ID_{B} are not revealed to adversary C.

(i) *Setup.* S assigns the values to all parties' long-term secret keys and KGC's master keys as follows.

(1) $s \in \mathbb{Z}_p^*$ is chosen by S as KGC's master secret key and the challenger S also calculates $P_{\text{pub}} = sP$. Thus, P_{pub} is its public key. In fact, case 3 simulates MFS.

(2) For each party, S selects s_i, $h_i \in \mathbb{Z}_p^*$ at random. S sets $h_i = H_1(\text{ID}_i, R_i)$ and calculates $R_i = s_iP - h_iP_{\text{pub}}$. So ID_i's long-term secret key is (s_i, R_i). Then, S computes $\text{PK}_i = R_i + h_iP_{\text{pub}} = s_iP$. In the end, S sends (ID_i, R_i) to the adversary C. In addition, (ID_i, R_i, h_i) is inserted to the table H_1^{list}.

(ii) *Queries.* As before, S holds four blank tables H_1^{list}, H_2^{list}, H_3^{list}, and R^{list} to cope with corresponding queries. Those queries from C are responded by S in the following ways.

(1) $H_3(\text{ID}_i, \text{ID}_j, T_i, T_j, \text{SK}, a_i\text{PK}_j, s_iT_j)$. The challenger S has an empty list H_3^{list} in the form of $(\text{ID}_i, \text{ID}_j, T_i, T_j, \text{SK}, a_i\text{PK}_j, s_iT_j, h_3)$.

(i) If the list H_3^{list} already has the matching entry, S returns h_3 to C.

(ii) Otherwise, S checks the whole table R^{list}. If the item is found out, S sets $h_3 = \text{sk}_{ij}$ and puts the new entry into the list H_3^{list}. If not, $h_3 \in \{0,1\}^k$ is randomly chosen by S and the corresponding data is written into H_3^{list}.

(2) *StaticKeyReveal*(ID_i). (s_i, R_i) is revealed by S to C.

(3) *MasterPrivateKeyReveal*. The challenger S responds to this query with s.

(4) *EphemeralKeyReveal*($\Pi_{i,j}^s$). If $\Pi_{i,j}^s$ is $\Pi_{\text{A,B}}^s$ or $\Pi_{\text{B,A}}^t$, S aborts. Otherwise, S returns the ephemeral key of ID_i to C.

(5) *Send*($\Pi_{i,j}^s$, M). S maintains an empty list R^{list} in the form of $(\text{ID}_i, \text{ID}_j, T_i, T_j, \text{sk}_{ij})$.

(i) If $\Pi_{i,j}^s = \Pi_{\text{A,B}}^s$, S returns X to C.

(ii) If $\Pi_{i,j}^s = \Pi_{\text{B,A}}^t$, S returns Y to C. Then, S searches for the relevant entry in H_3^{list}. If the item is gotten by S, challenger S sets $\text{sk}_{ij} = h_3$ and the table R^{list} is added with this new entry. Conversely, $\text{sk}_{ij} \in \{0,1\}^k$ is randomly selected by S and the corresponding item is inserted into table R^{list}.

(iii) For other conditions, S responds to the C according to the protocol specification.

(iii) *Analysis.* Assume that adversary C can make a successful forging attack with non-negligible probability $\mathbf{Adv}_{\text{C}}^{\Pi}$, C must make a query to H_3 with the input $\text{SK} = h_2\text{DLOG}(X)Y$ and $a_{\text{A}}\text{PK}_{\text{B}} = s_{\text{B}}X$ and $s_{\text{A}}Y$. To solve the $GDH(X,Y)$, S checks whether the value of an H_3 query from C is $(\text{ID}_{\text{A}}, \text{ID}_{\text{B}}, T_{\text{A}}, T_{\text{B}}, h_2\text{DLOG}(X)Y, s_{\text{B}}X, s_{\text{A}}Y)$ such that $DDH(h_2X, Y, \text{SK}) = 1$, $DDH(X, \text{PK}_{\text{B}}, s_{\text{B}}X) = 1$, and $DDH(\text{PK}_{\text{A}}, Y, s_{\text{A}}Y) = 1$. If such an H_3 query is found, S can correctly calculate $GDH(X,Y) = \text{DLOG}(X)Y = h_2^{-1}\text{SK}$. Therefore, the GDH problem is solved by S with an advantage

$$\mathbf{Adv}_{\text{S}}^{GDH}(k) \geq \frac{\mathbf{Adv}_{\text{C}}^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{6}$$

Case 4. ID_{B}'s temporary private key and the long-term secret key of ID_{A} are not acquired by adversary C. For case 4, we can consider this case as case 1. Thus, S can use the similar way used in case 1 to make this simulation. Therefore, GDH problem is dealt by S successfully with great advantage

$$\mathbf{Adv}_{\text{S}}^{GDH}(k) \geq \frac{\mathbf{Adv}_{\text{C}}^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{7}$$

Case 5. Adversary C does not get long-term secret key of ID_{B} and ID_{A}'s temporary private key. But the value of R_{B} is changed by C.

Firstly, KGC's master public key is selected as $X \in \mathbb{G}$. For all participants, S randomly chooses h_i, $s_i \in \mathbb{Z}_p^*$. Next, S makes the equation $h_i = H_1(\mathrm{ID}_i, R_i)$ and gets the value of $R_i = s_i P - h_i P_{\mathrm{pub}}$. So (s_i, R_i) is ID_i's long-term secret key. Meanwhile, $\mathrm{PK}_i = R_i + h_i P_{\mathrm{pub}} = s_i P$ is its long-term public key. Then, for every ID_i, challenger S returns (ID_i, R_i) to the adversary C and adds $(\mathrm{ID}_i, R_i, h_i)$ to the table H_1^{list}. Besides, S sets Y as the ephemeral public key of ID_A. The answers to all kinds of queries from C are easy, because all participants' long-term secret keys are known by S.

Secondly, we assume that C neither queries EphemeralKeyReveal($\Pi^s_{A,B}$) nor StaticKeyReveal(ID_B). A message $(\mathrm{ID}_B, R_{B,C}, bP)$ made by adversary C is sent to the session $\Pi^s_{A,B}$. Here, the adversary C selects $b \in \mathbb{Z}_p^*$ at random and may change R_B of ID_B to $R_{B,C}$. For participant ID_B, $\widetilde{h}_B \in \mathbb{Z}_p^*$ is chosen by the challenger S and $H_1(\mathrm{ID}_B, R_{B,C}) = \widetilde{h}_B$ is also assigned by S. Finally, we make an assumption that the successful probability of a forgery attack made by C is $\mathbf{Adv}_C^{\Pi}$. So C must make a query to H_3 with the input $\mathrm{SK} = H_2(s_A \mathrm{PK}_B) bY = h_2 bY$, $a\mathrm{PK}_B = \mathrm{DLOG}(Y)(R_B + \widetilde{h}_B X)$, and $s_A T_B = s_A bP$. S checks whether C makes an H_3 query on the value $(\mathrm{ID}_A, \mathrm{ID}_B, T_A, T_B, \mathrm{SK}, \mathrm{DLOG}(Y)(R_B + \widetilde{h}_B X), s_A bP)$ such that $DDH(h_2 Y, T_B, \mathrm{SK}) = 1$, $DDH(Y, \mathrm{PK}_B, \mathrm{DLOG}(Y)\mathrm{PK}_B) = 1$, and $DDH(\mathrm{PK}_A, T_B, s_A T_B) = 1$.

On the basis of forking lemma, S restarts the game with adversary C using the same data. Similarly, $\widetilde{h}_B'$ is randomly selected and assigned to $H_1(\mathrm{ID}_B, R_{B,C})$ by S, and $\widetilde{h}_B'$ is not equal to $\widetilde{h}_B$. Assume that the probability of a forgery attack launched successfully can not be ignored. An H_3 query must be requested by C with the input $\mathrm{SK} = h_2 bY$, $a\mathrm{PK}_B = \mathrm{DLOG}(Y)(R_B + \widetilde{h}_B' X)$, and $s_A T_B = s_A bP$. Then, S does as above.

In order to cope with the GDH, challenger S does a simple calculation on $K = \mathrm{DLOG}(Y)(R_B + \widetilde{h}_B X) - \mathrm{DLOG}(Y)(R_B + \widetilde{h}_B' X) = \mathrm{DLOG}(Y)(\widetilde{h}_B - \widetilde{h}_B')X$. So the right value $GDH(X, Y) = \mathrm{DLOG}(Y)X = (\widetilde{h}_B - \widetilde{h}_B')^{-1} K$ can be acquired by S. If λ is forking lemma's utilization parameter, GDH difficult problem can be successfully dealt by S with the advantage

$$\mathbf{Adv}_S^{GDH}(k) \geq \frac{\lambda \mathbf{Adv}_C^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{8}$$

Case 6. Adversary C knows nothing about long-term secret keys of ID_A and ID_B. But C alters R_B's real value.

At first, KGC's public key is assigned by S using a random number $X \in \mathbb{G}$. Considering about ID_i, where A is not the right value of i, h_i, $s_i \in \mathbb{Z}_p^*$ is selected and $R_i = s_i P - h_i P_{\mathrm{pub}}$ is calculated by S. Then, S makes the equation $h_i = H_1(\mathrm{ID}_i, R_i)$ true and gives ID_i the long-term secret key (s_i, R_i). Thus, relevant public key of ID_i gets the value $\mathrm{PK}_i = R_i + h_i P_{\mathrm{pub}} = s_i P$. Particularly, for ID_A, $H_1(\mathrm{ID}_A, R_A)$ is assigned to $h_A \in \mathbb{Z}_p^*$ picked by S at random and $R_A = Y - h_A P_{\mathrm{pub}}$ can be worked out. Then, the long-term secret key of ID_A gets the value (Δ, R_A). So its long-term public key is $\mathrm{PK}_A = R_A + h_A P_{\mathrm{pub}} = Y$. Besides, S sends (ID_i, R_i) of all parties to the adversary C and stores $(\mathrm{ID}_i, R_i, h_i)$ to the table H_1^{list}. Particularly, the temporary secret key of ID_A is given the value $a \in \mathbb{Z}_p^*$ selected by S randomly.

Secondly, it is an assumption that C does not request queries about StaticKeyReveal(ID_A) and StaticKeyReveal(ID_B). Moreover, the simulation does not abort. A message $(\mathrm{ID}_B, R_{B,C}, bP)$ made by C is sent to the session $\Pi^s_{A,B}$. $b \in \mathbb{Z}_p^*$ is randomly selected by the adversary and C can also change R_B. Assume that S chooses $\overline{h}_B \in \mathbb{Z}_P^*$ at randomly as $H_1(\mathrm{ID}_B, R_{B,C})$. If the probability of a successful forgery attack maked by C cannot be ignored, the adversary must launch an H_3 asking with the input $s_A T_B = \mathrm{DLOG}(Y) bP$, $a_A \mathrm{PK}_B = a(R_B + \overline{h}_B X)$, and $\mathrm{SK} = H_2(\mathrm{DLOG}(Y)\mathrm{PK}_B)$ $abP = H_2(\mathrm{DLOG}(Y)(R_B + \overline{h}_B X))abP$. Next, S checks whether there is an H_3 query from the adversary C on the value $(\mathrm{ID}_A, \mathrm{ID}_B, T_A, T_B, \mathrm{SK}, \mathrm{DLOG}(Y)bP, a\mathrm{PK}_B)$ such that $DDH(h_2 T_A, T_B, \mathrm{SK}) = 1$, $DDH(T_A, \mathrm{PK}_B, a\mathrm{PK}_B) = 1$, and $DDH(Y, T_B, s_A T_B) = 1$.

Similarly, based on forking lemma, S replays the game with adversary C using the same data. S gets $\overline{h}_B' \in \mathbb{Z}_P^*$ as $H_1(\mathrm{ID}_B, R_{B,C})$. We should note that $\overline{h}_B' \neq \overline{h}_B$. As above, if the probability of a forgery attack cannot be ignored, adversary C must make an H_3 query with the input $s_A T_B = \mathrm{DLOG}(Y) bP$, $a_A \mathrm{PK}_B = a(R_B + \overline{h}_B' X)$, and $\mathrm{SK}' = H_2(\mathrm{DLOG}(Y)\mathrm{PK}_B) abP = H_2(\mathrm{DLOG}(Y)(R_B + \overline{h}_B' X)) abP$. Then, S verifies whether such an H_3 query from the adversary C exists on the value $(\mathrm{ID}_A, \mathrm{ID}_B, T_A, T_B, \mathrm{SK}, \mathrm{DLOG}(Y)bP, a\mathrm{PK}_B)$ such that $DDH(h_2 T_A, T_B, \mathrm{SK}') = 1$, $DDH(T_A, \mathrm{PK}_B, a\mathrm{PK}_B) = 1$, and $DDH(Y, T_B, s_A T_B) = 1$.

In order to deal with the GDH problem, S calculates $K = \mathrm{DLOG}(Y)(R_B + \overline{h}_B X) - \mathrm{DLOG}(Y)(R_B + \overline{h}_B' X) = \mathrm{DLOG}(Y) X(\overline{h}_B - \overline{h}_B')$. Therefore, S can compute $GDH(X, Y) = (\overline{h}_B - \overline{h}_B')^{-1} K$. If λ is forking lemma's utilization parameter, GDH difficult problem can be successfully dealt by S with an advantage

$$\mathbf{Adv}_S^{GDH}(k) \geq \frac{\lambda \mathbf{Adv}_C^{\Pi}(k)}{6n(k)^2 q_s(k)}. \tag{9}$$

All in all, because $\mathbf{Adv}_C^{\Pi}(k)$ is considered to be non-negligible, $\mathbf{Adv}_S^{GDH}(k)$ also cannot be ignored. But it is contradictory to the GDH assumption.

5.4. Other Discussions. We will show some essential security attributes that our pairing-free 2PAKA protocol holds in the following content.

(i) *Mutual authentication*. The security proof shows that a useful message cannot be successfully made by any adversary in polynomial time because authentications among users can be achieved by verifying whether those messages that they get are valid or not. Therefore, our protocol has the hidden function of mutual authentication.

(ii) *Session Key Agreement*. According to our protocol shown in the Key Agreement phase, one session key $\mathrm{sk}_{ij} = H_3(\mathrm{ID}_i, \mathrm{ID}_j, \mathrm{SK}, a_i \mathrm{PK}_j, s_i T_j) = H_3(\mathrm{ID}_i, \mathrm{ID}_j, \mathrm{SK}, b_j \mathrm{PK}_i, s_j T_i)$ can be obtained by two users after communication. Thus, our

protocol can complete the negotiation process of a session key.

(iii) *Known Session Key Security (KSKS) Resistance*. Our protocol can achieve KSKS security property because Session Key Reveal is allowed in the eCK strong security model. Namely, any other session' key could be revealed to an adversary excluding the goal session and its matching session's keys. Meanwhile, the probability of the event that C can successfully distinguish goal session key from a random number can be ignored. So C cannot use other session keys to know test session key.

(iv) *Forward Secrecy (FS)*. As it shows in our security proof, it is forbidden that both a user's long-term and temporary secret keys are known by an adversary. In other words, our protocol can get wPFS. Besides, the case 3 simulates MFS.

(v) *Key Compromise imPersonation (KCI) Resistance*. In case 1, 3, and 5, the adversary C cannot generate the goal key acquired by ID_A and any other user when C knows ID_A's the long-term secret key and even changes those messages sent to ID_A.

(vi) *Unknown Key Share (UKS) Resistance*. Because identity is the core foundation of our protocol, it indicates that one user's public key is generated depending on its ID. Obviously, the UKS attack can be resisted by our protocol.

(vii) *No Key Control (NKC) Resistance*. Because H_3 hash function does not have the same result with different input, the session key cannot be determined by one party or the adversary. Thus, our protocol can catch NKC resistance.

(viii) *BI Resistance*: If the long-term secret key of ID_A is not obtained by adversary C, C cannot successfully calculate the correct input to H_3. So C cannot get the session key generated by ID_A and any other party.

(ix) *Ephemeral Key Reveal (EKR) Resistance*: In case 1, 4, and 5, the security still can be held by our protocol while its temporary secret keys are leaked partially. In case 2 and 6, when the ephemeral keys are compromised completely, this protocol also keeps safe.

6. Performance Analysis

Within this module, our protocol's performance is analyzed from computational cost and running time. Besides, we display that our protocol is compared with other related protocols [13, 14] in terms of efficiency.

In our experiment, an additive group $\mathbb{G}$ is selected by us, where q is its order. This group has a generator P. The order q is a big prime number with 160-bits and P is a point chosen from a common elliptic curve E/F_p: $y^2 = x^3 + 1$. Here, the number of bits of prime number p is 512.

6.1. Analysis of Computational Cost. For better computational cost analysis, we firstly give out the comparison results between our protocol and some valuable protocols [13, 14] in terms of message size in the Table 1. Then, on the basis of message size, we analyze their computational cost and give the results in Figure 3.

Assume that the ID of one party is 2 bytes long. In addition, messages exchanged between two parties in our protocol are ID_i, R_i, T_i. Here, R_i and T_i belong to $\mathbb{G}$. So the messages are one ID and two points, whose total size is $(2 * 8 + 160 * 2)/8 = 42$ bytes. Similarly, the size of exchanged messages in the Bala et al.'s protocol is also 42 bytes. However, in Dang et al.'s protocol, the messages' size is 62 bytes.

Next, we present the executing time of some basic operations in Table 2.

We have achieved these basic operations in the MIRACL library [30]. The implementations were deployed in a personal computer and the platform's parameters are displayed in the following Table 3.

What deserves our attention is that our protocol is a symmetrical structure. In other words, the party ID_A and ID_B are making the computational operations at the same time. Thus, we only need to consider about the computational cost of one party. In our protocol, the computational operation only includes $T_i = a_iP$. Fortunately, this operation is only a simple scalar multiplication. But among the other two protocols being compared, we can find the computational operations are both four scalar multiplications. When precomputation is considered, there are still three scalar multiplications in the Bala et al.'s protocol and two scalar multiplications in the Dang et al.'s protocol. The result of computational cost is shown in the following figure.

Now, we know the computational operations in the three protocols. Moreover, we can find that most of these computational operations can be completed offline. We can know that a scalar multiplication needs 2.165 ms in the personal computer according to Table 2. Therefore, we can get the respective computational time of the three protocols. In general, we take the precomputation into consideration. So the computational time of Bala et al.'s protocol is $3 * 2.165$ ms $= 6.495$ ms. In the Dang et al.'s protocol, it needs $2 * 2.156$ ms $= 4.33$ ms. But in our protocol, we only need $1 * 2.165$ ms $= 2.165$ ms to achieve the required operation. Therefore, our protocol has better performance.

6.2. Analysis of Running Time. Generally, the running time of an AKA protocol is approximately made up of computational time and transmission time. Here, we can only consider the transmission time, because we already know the corresponding computational time of each protocol. As for the transmission time, we think the transmission time is mainly related to message size and hardware performance. We assume that these hardware equipments have similar performance. Hence, if the message size is longer, more time is needed to transmit it. Fortunately, we have analyzed these protocols' message size in the analysis of computational cost.

Table 1: Message size of ID-based 2PAKA protocols.

Protocol	Message content	Message size (bytes)
Bala et al.	One ID, two points	42
Dang et al.	One ID, three points	62
Our protocol	One ID, two points	42

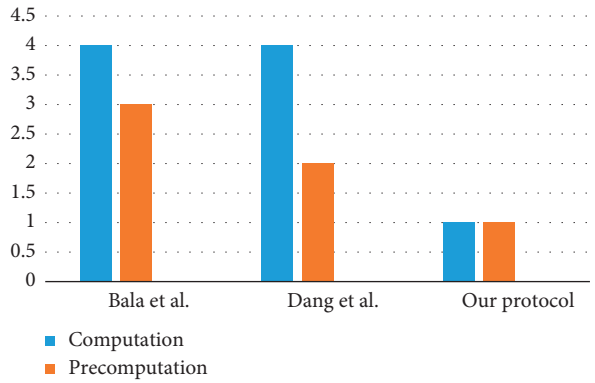

Figure 3: Comparison of computational cost.

Table 2: Executing time of basic operations (ms).

Operation	Description	Personal computer
T_{sm}	Scalar multiplication	2.165
T_{hf}	Hash function	0.007
T_{bp}	Bilinear pairing	5.427

Table 3: Simulation platform.

Device	Dell
Operating system	Windows 8
CPU	15-4460S 2.90 GHz
Memory	4 GB RAM
Program language	C

Table 1 shows that our protocol has smaller message size than Dang's protocol and the same size as Bala's protocol. Therefore, compared with other ID-based protocols, our protocol can get stronger or be at the same level of security with less running time.

In conclusion, the performance of our protocol is better than that of the other two protocols. Besides, our protocol is superior to the Dang's protocol in resisting attacks. Therefore, our protocol has better performance in VANETs environment compared with previous ID-based 2PAKA protocols.

7. Conclusion

To be able to deal with the increasing demands of VANETs (e.g., due to the increasing number of connected vehicles and devices), we constructed a new efficient 2PAKA protocol based on the identity in this paper. This protocol was designed to provide an authentication function and a session key to two users in an efficient way. Besides, we showed that our protocol has strong security in the eCK model, and it outperforms two other recently proposed 2PAKA protocols [13, 14].

Future research includes extending the protocol to achieve other desirable properties, as well as implementing an initial model of the extended protocol for evaluation in a practical application.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by the National Key Research and Development Program of China (No. 2018YFC1604000), the National Natural Science Foundation of China (Nos. 61772224, 61932016, and 61972294), and the Fund of the Jiangsu Key Laboratory of Big Data Security & Intelligent Processing (No. BDSIP1807).

References

[1] E. C. Eze, S. Zhang, and E. Liu, "Vehicular ad hoc networks (vanets): current state, challenges, potentials and way forward," in *Proceedings of the 20th International Conference on Automation and Computing, ICAC 2014*, X. Luo, Y. Cao, and Z. Tong, Eds., pp. 176–181, IEEE, Cranfield, Bedfordshire, UK, September 2014.

[2] Y. Yang, Z. Wei, Y. Zhang, H. Lu, K.-K. R. Choo, and H. Cai, "V2X security: a case study of anonymous authentication," *Pervasive and Mobile Computing*, vol. 41, pp. 259–269, 2017.

[3] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[4] N. P. Smart, "An identity based authenticated key agreement protocol based on the weil pairing," *IACR Cryptology*, vol. 2001, p. 111, 2001.

[5] M. Hölbl, T. Welzer, and B. Brumen, "An improved two-party identity-based authenticated key agreement protocol using pairings," *Journal of Computer and System Sciences*, vol. 78, no. 1, pp. 142–150, 2012.

[6] L. Ni, G. Chen, and J. Li, "Escrowable identity-based authenticated key agreement protocol with strong security," *Computers & Mathematics with Applications*, vol. 65, no. 9, pp. 1339–1349, 2013.

[7] L. Ni, G. Chen, J. Li, and Y. Hao, "Strongly secure identity-based authenticated key agreement protocols in the escrow mode," *Science China Information Sciences*, vol. 56, no. 8, pp. 1–14, 2013.

[8] S. S. Vivek, S. S. D. Selvi, L. R. Venkatesan, and C. P. Rangan, "Efficient, pairing-free, authenticated identity based key agreement in a single round," in *Proceedings of the 7th International Conference, ProvSec 2013*, W. Susilo and R. Reyhanitabar, Eds., vol. 8209 of Lecture Notes in Computer Science, pp. 38–58, Springer, Melaka, Malaysia, October 2013.

[9] S. H. Islam and G. P. Biswas, "Design of two-party authenticated key agreement protocol based on ECC and self-certified public keys," *Wireless Personal Communications*, vol. 82, no. 4, pp. 2727–2750, 2015.

[10] S. Chakraborty, S. Raghuraman, and C. P. Rangan, "A pairing-free, one round identity based authenticated key exchange protocol secure against memory-scrapers," *JoWUA*, vol. 7, no. 1, pp. 1–22, 2016.

[11] L. Ni, G. Chen, J. Li, and Y. Hao, "Strongly secure identity-based authenticated key agreement protocols without bilinear pairings," *Information Sciences*, vol. 367-368, pp. 176–193, 2016.

[12] R. W. Zhu, G. Yang, and D. S. Wong, "An efficient identity-based key exchange protocol with KGS forward secrecy for low-power devices," *Theoretical Computer Science*, vol. 378, no. 2, pp. 198–207, 2007.

[13] L. Dang, J. Xu, X. Cao et al., "Efficient identity-based authenticated key agreement protocol with provable security for vehicular ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 14, no. 4, 2018.

[14] S. Bala, G. Sharma, and A. K. Verma, "PF-ID-2PAKA: pairing free identity-based two-party authenticated key agreement protocol for wireless sensor networks," *Wireless Personal Communications*, vol. 87, no. 3, pp. 995–1012, 2016.

[15] D. He and D. Wang, "Robust biometrics-based authentication scheme for multiserver environment," *IEEE Systems Journal*, vol. 9, no. 3, pp. 816–823, 2015.

[16] D. He, N. Kumar, M. K. Khan, L. Wang, and J. Shen, "Efficient privacy-aware authentication scheme for mobile cloud computing services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1621–1631, 2018.

[17] A. Joux, "A one round protocol for tripartite diffie–hellman," in *Proceedings of the International Algorithmic Number Theory Symposium*, pp. 385–393, Springer, Leiden, The Netherlands, July 2000.

[18] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of Lecture Notes in Computer Science, pp. 213–229, Springer, Santa Barbara, CA, USA, August 2001.

[19] K. Shim, "Efficient id-based authenticated key agreement protocol based on weil pairing," *Electronics Letters*, vol. 39, no. 8, pp. 653-654, 2003.

[20] H.-M. Sun and B.-T. Hsieh, "Security analysis of shim's authenticated key agreement protocols from pairings," *IACR Cryptology*, vol. 2003, p. 113, 2003.

[21] D. Fiore and R. Gennaro, "Making the diffie-hellman protocol identity-based," in *Proceedings of the Cryptographers' Track at the RSA Conference on Topics in Cryptology, CT-RSA 2010*, J. Pieprzyk, Ed., vol. 5985 of Lecture Notes in Computer Science, pp. 165–178, Springer, San Francisco, CA, USA, March 2010.

[22] X. Cao, W. Kou, and X. Du, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges," *Information Sciences*, vol. 180, no. 15, pp. 2895–2903, 2010.

[23] H. Sun, Q. Wen, H. Zhang, and Z. Jin, "A strongly secure identity-based authenticated key agreement protocol without pairings under the GDH assumption," *Security and Communication Networks*, vol. 8, no. 17, pp. 3167–3179, 2015.

[24] L. Ni, G. Chen, J. Li, and Y. Hao, "Strongly secure identity-based authenticated key agreement protocols," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 205–217, 2011.

[25] S. H. Islam and G. P. Biswas, "A pairing-free identity-based two-party authenticated key agreement protocol for secure and efficient communication," *Journal of King Saud University—Computer and Information Sciences*, vol. 29, no. 1, pp. 63–73, 2017.

[26] D. He, N. Kumar, H. Wang, L. Wang, K.-K. R. Choo, and A. Vinel, "A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 633–645, 2018.

[27] Q. Feng, D. He, S. Zeadally, N. Kumar, and K. Liang, "Ideal lattice-based anonymous authentication protocol for mobile devices," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2775–2785, 2018.

[28] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, "BSeIn: a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, no. 1, pp. 42–52, 2018.

[29] H. Huang and Z. Cao, "An id-based authenticated key exchange protocol based on bilinear diffie-hellman problem," in *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009*, W. Li, W. Susilo, U. K. Tupakula et al., Eds., pp. 333–342, ACM, Sydney, Australia, March 2009.

[30] Shamus Software Ltd, "Miracl library," 2016, http://www.shamus.ie/index.php?page=home.