symmetry

MDPI

Article

Ephemeral-Secret-Leakage Secure ID-Based Three-Party Authenticated Key Agreement Protocol for Mobile Distributed Computing Environments

Chao-Liang Liu [1], Wang-Jui Tsai [2], Ting-Yi Chang [2,*] and Ta-Ming Liu [2]

[1] Department of Applied Informatics and Multimedia, Asia University, Lioufeng Rd., Wufeng, Taichung 413, Taiwan; jliu@asia.edu.tw

[2] Department of Industrial Education and Technology, National Changhua University of Education, Changhua City 500, Taiwan; tychang@cc.ncue.edu.tw (W.-J.T.); aliu0605l@gmail.com (T.-M.L.)

* Correspondence: tychange@cc.ncue.edu.tw; Tel.: +886-4723-2105 (ext. 7381)

Received: 25 January 2018; Accepted: 13 March 2018; Published: 28 March 2018

Abstract: A three-party Authenticated Key Agreement (AKA) protocol in the distributed computing environment is a client that requests services from an application server through an authentication server. The authentication server is responsible for authenticating the participating entities and helping them to construct a common session key. Adopting the Key Transfer Authentication Protocol (KTAP) in such an environment, the authentication server is able to monitor the communication messages to prevent and trace network crime. However, the session key in the KTAP setting is created only by the authentication server and is vulnerable to the resilience of key control. On the other hand, with the rapid growth of network technologies, mobile devices are widely used by people to access servers in the Internet. Many AKA protocols for mobile devices have been proposed, however, most protocols are vulnerable to Ephemeral Secret Leakage (ESL) attacks which compromise the private keys of clients and the session key by an adversary from eavesdropped messages. This paper proposes a novel ESL-secure ID-based three-party AKA protocol for mobile distributed computing environments based on ESL-secure ID-based Authenticated Key Exchange (ID-AKE) protocol. The proposed protocol solves the key control problem in KTAP while retaining the advantages of preventing and tracing network crime in KTAP and also resists ESL attacks. The AVISPA tool simulation results confirm the correctness of the protocol security analysis. Furthermore, we present a parallel version of the proposed ESL-secure ID-based three-party AKA protocol that is communication-efficient.

Keywords: Ephemeral-Secret-Leakage; distributed computing; three-party authenticated key agreement protocol; mobile device; bilinear pairing; Automated Validation of Internet Security Protocols and Applications (AVISPA) tool

1. Introduction

With the rapid growth of network technologies, portable mobile devices (e.g., mobile phone, notebook and tablet) are widely used by people to access remote servers on the Internet. Due to the limited computing capability and power energy of mobile devices, many Authenticated Key Agreement (AKA) protocols are based on the traditional public key cryptography system [1–3] and the ID-based AKA (ID-AKA) protocols [4–6] for mobile devices. Imbalanced computation is used to shift the computational burden to a powerful server using an online/offline computation technique to further reduce the mobile device computational load. If an AKA protocol adopted an online/offline computation technique, the ephemeral secrets are generally generated by an external source that may be controlled by an adversary. The ephemeral secrets are also involved in the offline pre-computation

and stored in the insecure memory of mobile devices. If ephemeral secrets are compromised, an adversary can reveal the private keys of clients and the session key would turn out to be known from the eavesdropped messages. This phenomenon is called Ephemeral Secret Leakage (ESL) attacks [7]. To solve this security vulnerability, Tseng et al. [7] proposed the first ESL-secure ID-based Authenticated Key Exchange (ID-AKE) protocol for mobile client-server environments.

In 1976, Diffie and Hellman [8] proposed the first Public Key Cryptography (PKC) concept, and proposed the first key agreement protocol that allows two participants to construct a common session key over a public network. Unfortunately, Diffie and Hellman's protocol does not authenticate the communication participants and is vulnerable to the man-in-the-middle attacks. Different approaches have been proposed by cryptographic researchers to defeat the weakness in terms of improving protocol security and efficiency. An AKA protocol should provide implicit key authentication for participants. Each participant is ensured that no other participants or adversaries can learn or determine the value of a common session key in a protocol run. The common session key is used to ensure information integrity, confidentiality and availability between participants using symmetric encryptions.

In traditional public key cryptography systems, the users have to access and verify other user's certificate before using its public key through the Certificate Authority (CA). The CA requires high computational cost and storage efforts to manage the certificates and also increases the computational cost for the client side [9]. Shamir [10] proposed the first ID-based cryptosystem to simplify the certificate management procedures. In Shamir's ID-based cryptosystem setting, the user's public key is an easy calculated function of the user's known identity information such as identity numbers, E-mail address and so forth. The corresponding private key can be calculated and issued to the user via a secure channel by a trusted party referred to as Private Key Generator (PKG). In this way, the users do not need to verify other user's certificates before using the public key and thus can substantially reduce the computational burden of the CA. However, Shamir's ID-based cryptosystem is not easy to be realized in practice due to lack of efficient encryption and decryption algorithms and thus restricts its development. Boneh and Franklin [11] proposed the first secure and efficient practical ID-based encryption scheme based on the Weil pairing defined on elliptic curves. Subsequently, the ID-based cryptographic schemes based on bilinear pairings have received much attention from cryptographic researchers and a large number of ID-based cryptographic systems using bilinear pairings have been published in the literatures [12–14].

Joux [15] proposed the first three-party key agreement protocol based on bilinear pairings. Joux's key agreement protocol is the first key agreement protocol based on bilinear pairings and also the first one-round three-party key agreement protocol. However, like the basic Diffie-Hellman key agreement protocol, Joux's protocol without authentication is also insecure against the man-in-the-middle attacks. To solve this problem, Al-Riyami and Paterson [16] proposed several three-party AKA protocols that use bilinear pairings. Al-Riyami and Paterson's protocols overcome the security flaw in Joux's protocol. However, Al-Riyami and Paterson's protocols still need certificates issued by the CA to ensure the authenticity and a user also needs to verify the certificates before using other users' public keys. Afterwards, many ID-based three-party AKA protocols using bilinear pairings have rapidly emerged and been well-studied as well [17–19].

Up to now, most of the literature on ID-based three-party AKA protocols using bilinear pairings focused on the environment in which PKG computes the public key and corresponding private key from a user's identity information, and issuing the public/private key pair to the user via a secure channel. The participants can authenticate each other using the public/private key pair and construct a common session key. In 2004, Yeh et al. [20] pointed out that another common communication environment exists, referred to as the distributed computing environment, discussed in Kerberos [21]. In this open distributed computing environment, if the users would like to access services on application servers distributed throughout the network, a centralized authentication server is provided rather than building an elaborate authentication protocol at each server. The authentication

server is responsible for participating entities' authentication and helps the user and the application server construct a common session key. In the centralized authentication server model an inspection mechanism is provided to prevent and trace network crime, permitting users to access the services provided by the application servers legally. The Key Transfer Authentication Protocol (KTAP) is adopted in this system [22]. In a KTAP setting the session key is created by the authentication server and secretly transmitted to the user and the application server. The authentication server can therefore monitor the transferred messages [23–26]. However, the disadvantage of KTAP is that the session key is only created by the authentication server and the client and the application servers are unable to participate in constructing the common session key. KTAP is therefore vulnerable to key control resilience attacks [27].

With the rapid growth of network technologies, portable mobile devices (e.g., mobile phone, notebook and tablet) are widely used by people to access remote servers on the Internet. Due to the limited computing capability and power energy of mobile devices, many AKA protocols are based on the traditional public key cryptography system [1–3] and the ID-based AKA (ID-AKA) protocols [4–6] for mobile devices. Imbalanced computation is used to shift the computational burden to a powerful server using an online/offline computation technique to further reduce the mobile device computational load. If an AKA protocol adopted an online/offline computation technique, the ephemeral secrets are generally generated by an external source that may be controlled by an adversary. The ephemeral secrets are also involved in the offline pre-computation and stored in the insecure memory of mobile devices. If ephemeral secrets are compromised, an adversary can reveal the private keys of clients and the session key would turn out to be known from the eavesdropped messages. This phenomenon is called Ephemeral Secret Leakage (ESL) attacks. To solve this security vulnerability, Tseng et al. [7] proposed the first ESL-secure ID-based Authenticated Key Exchange (ID-AKE) protocol for mobile client-server environments.

Most KTAPs [20,23–25] are password-based authentication protocols. In these protocols the users need to share their own passwords with the authentication server and employ the authentication server public keys to ensure the identities of the participants. These protocols also use symmetric cryptosystems to encrypt the transferred messages. For password-based authentication protocol security a strong password should consist of letters (uppercase letters, lowercase letters), numbers and special punctuations to resist various attacks, such as password guessing attacks and dictionary attacks, etc. However, most mobile devices do not employ standard QWERTY keyboards for users to conveniently enter strong passwords. Instead, these mobile devices often use numeric passwords for user authentication, which is called Personal Identification Number based (PIN-based) authentication. The PIN-based authentication provides a small password space size and thus is vulnerable to various attacks [28]. Otherwise, using the authentication server public keys and the symmetric cryptosystems for the user authentication requires expensive computation, which is not applicable to mobile devices with limited computing capability.

This paper improved Tseng et al.'s scheme [7], to propose an ESL-secure ID-based three-party AKA protocol which is more suitable for mobile distributed computing environments. The proposed protocol adopts imbalanced computation to shift the computational burden to the powerful server and an online/offline computation technique to further reduce the computational cost required for mobile devices. The offline pre-computation is executed prior to protocol execution to achieve better performance. The proposed protocol keeps all of the merits of KTAP regarding security including the authentication server is able to monitor the communication messages to prevent and trace network crime and solves the session key problem, which is only created by the authentication server. All participants can contribute information to derive the common session key. In the security analysis, the proposed protocol resists ESL attacks and also satisfies the security attributes required for AKA protocols: known-key security, partial forward secrecy, key-compromise impersonation resilience, unknown key-share resilience and key control resilience [29]. Furthermore, the proposed protocol is validated by Automated Validation of Internet Security Protocols and Applications (AVISPA) [30]

formal validation tool to show its security against various active and passive attacks. In addition, a parallel version of the proposed protocol is proposed to enhance the protocol run performance.

This paper is organized as follows. Section 2 gives a brief review of the basic bilinear pairing concept, the related mathematical assumptions and the security attributes required for AKA protocols and notions used in the proposed protocol. The proposed ESL-secure ID-based three-party AKA protocol is presented in Section 3. In Section 4 the proposed protocol security and performance analyses are conducted. Conclusions are given in Section 5.

2. Preliminaries

In this Section the basic bilinear pairings concept, the related mathematical assumptions, the security attributes required for AKA protocols, and the notations used in the proposed protocol are briefly introduced.

2.1. Bilinear Pairings

Let P denote a generator of G_1, where G_1 is an additive cyclic group of large prime order q and let G_2 be a multiplicative group of the same large prime order q. G_1 is a subgroup of the group of points on an elliptic curve E defined over a finite field. G_2 is a subgroup of the group of the multiplicative cyclic group defined over a finite field. A bilinear pairing is defined as a map: $\hat{e}: G_1 \times G_1 \rightarrow G_2$. The map $\hat{e}$ is called an admissible bilinear map if it satisfies the following properties.

1. Bilinearity: Let $P, Q, R \in G_1$, we have:

   (1) $\hat{e}(P+Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$.
   (2) $\hat{e}(P, Q+R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$.
   (3) $\hat{e}(aP, bP) = \hat{e}(bP, aP) = \hat{e}(P, P)^{ab}$.

2. Non-degeneracy: There exist $P \in G_1$ such that $\hat{e}(P, P) \neq 1$.
3. Computable: For $P, Q \in G_1$, there exists an efficient algorithm to compute $\hat{e}(P, Q)$.

2.2. Computational Problems

The security of the proposed protocol is based on the following two computational problems. There is no polynomial time algorithm to solve these computational problems with non-negligible probability:

1. *Discrete Logarithm Problem* (DLP): Give $P, Q \in G_1$; find an integer a such that $Q = aP$ whenever such integer exists.
2. *Computational Diffie-Hellman Problem* (CDHP): Given $P, aP, bP \in G_1$ for unknown $a, b \in Z_q^*$, the CDHP is to compute the value $abP \in G_1$.

2.3. Security Attributes

Here, A, B and S are going to agree upon a common session key and communicate to each other securely. An AKA protocol should provide implicit key authentication for A, B and S, so there are additional security attributes defined for AKA protocols.

- **Known-Key Security.** A unique session key should be constructed in each round of an AKA protocol. An adversary cannot derive other previous session keys if knowledge of the previous session keys has been compromised. The main purpose of known-key security is to ensure that the compromising of one session key will not compromise other or future session keys.
- **Forward Secrecy.** If the long-term private keys of one or more of the participants are compromised, the secrecy of previously established session keys will not be obtained by an adversary. The main purpose of forward secrecy is to provide complete protection for the previous transferred messages.

If all long-term private keys of the participants are compromised without compromising previous established session keys, that means an AKA protocol still provides protection for the previously transferred messages. We say that the AKA protocol offers perfect forward secrecy.

- **Key-Compromise Impersonation Resilience.** Suppose that A's private key has been revealed to an adversary. The adversary only can impersonate A to cheat S and B. It is desired that the compromise of A's private key does not allow the adversary to impersonate S or B to cheat A.
- **Unknown Key-Share Resilience.** After the session key has been established, A believes the session key is shared with S and B, while S and B mistakenly believe that the session key is instead shared with an adversary. Therefore, a desirable AKA protocol should be resistant to unknown key-share attacks. None of the participants can force A to establish a session key with a participant that he does not know but A believes he is sharing the session key with the participants that he knows.
- **Key Control Resilience.** The session key should be determined jointly by all participants (e.g., A, S and B). None of the participants can control the session key construction procedure alone. The main purpose of key control resilience is to ensure session key construction fairness and security. It should not be possible for any participants or adversaries to predict or predetermine the session key value.

Constructing a desirable AKA protocol must conform to these desirable security attributes: known-key security, forward secrecy, key-compromise impersonation resilience, unknown key-share resilience, key control resilience. Thus, an AKA protocol is able to resist various active and passive attacks.

2.4. Notations

The system parameters, notations and functions used in the whole proposed protocol are defined as follows:

- G_1: an additive cyclic group.
- G_2: a multiplicative cyclic group.
- $\hat{e}$: a bilinear map, $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
- P: a generator of the group G_1.
- s: the private key of the authentication server, $s \in Z_q^*$.
- P_{pub}: the public key of the authentication server, $P_{pub} = s \cdot P$.
- ID_S: the identity of the authentication server.
- ID_A: the identity of the client.
- ID_B: the identity of the application server.
- $DID_{A/B}$: the private key of ID_A/the private key of ID_B.
- $f_1(), f_2(), f_3(), f_4(), f_5(), f_6()$: six one-way hash functions, $f_1, f_2, f_3, f_4, f_5, f_6: \{0, 1\}^* \rightarrow \{0, 1\}^n$, where n is a fixed length and $2^n < q$.
- $H_1(), H_2()$: two map-to-point hash functions, $H_1, H_2: \{0, 1\}^* \rightarrow G_1$.

3. The Proposed Protocol

We present the proposed ESL-secure ID-based three-party AKA protocol for mobile distributed computing environments in this section. The proposed protocol consists of three phases: the system setup phase, the key extract phase and the mutual authentication and key agreement phases. We also present a parallel version of the proposed protocol to enhance the protocol run performance.

3.1. System Setup Phase

The proposed protocol system consists of an authentication server S, an application server B and a mobile client A. The client A refers to a user with handheld device and communicates with the

authentication server S and the application server B through open channels such as wireless networks. The application server B provides services or applications to the client A. The authentication server S is responsible for computing private keys according to the application server B and the client A' identities and distribute the private keys to them via a secure channel. The authentication server S is also responsible to generate the systems parameters.

In the system setup phase the authentication server S first generates two cyclic groups G_1 and G_2 of a large prime order q, an admissible bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$, and a random generator P of G_1, where G_1 and G_2 are additive and multiplicative cyclic groups of large prime order q, respectively. The authentication server S then performs the following tasks:

1. Randomly select a system private key $s \in Z_q^*$.
2. Compute the system public key $P_{pub} = s \cdot P$.
3. Choose two map-to-point hash functions $H_1, H_2: \{0,1\}^* \rightarrow G_1$.
4. Choose six one-way hash functions $f_1, f_2, f_3, f_4, f_5, f_6: \{0,1\}^* \rightarrow \{0,1\}^n$, where n is a fixed length and $2^n < q$.
5. Publish public parameters and functions $Params = < G_1, G_2, q, P, \hat{e}, P_{Pub}, H_1, H_2, f_1, f_2, f_3, f_4, f_5, f_6 >$.

3.2. Key Extract Phase

In the key extract phase client A and the application server B separately submit their identities ID_A and ID_B to the authentication server S and receive their corresponding private keys DID_A and DID_B, respectively. The key extract phase is depicted in Figure 1. To the client A as an example, the detailed procedures are presented as follows:

1. The client A submits its identity ID_A to the authentication server S.
2. Upon receiving the client A with identity ID_A, the authentication server S chooses an ephemeral secret value $l_A \in Z_q^*$, and compute $QID_{A1} = l_A \cdot P$, $h_A = f_1(ID_A, QID_{A1})$, $DID_{A1} = l_A + h_A \cdot s$, $QID_{A2} = H_1(ID_A)$, $QID_S = H_1(ID_S)$ and $DID_{A2} = s \cdot QID_{A2}$.
3. Set $DID_A = (DID_{A1}, DID_{A2}, QID_{A1}, QID_S)$ as the client A's private key and send it to the client A via a secure channel.

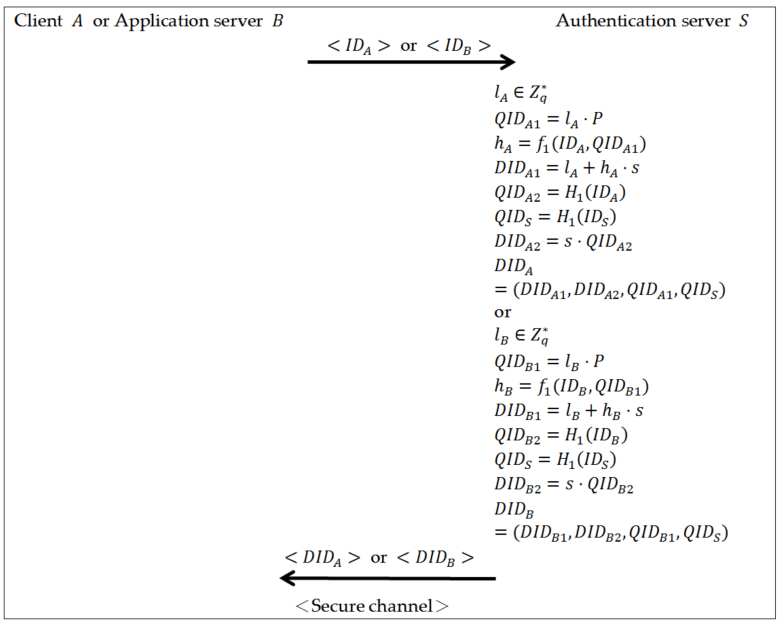

Figure 1. The key extract phase.

The authentication server S also set $DID_B = (DID_{B1}, DID_{B2}, QID_{B1}, QID_S)$ as the application server B's private key in the same procedures.

3.3. Mutual Authentication and Key Agreement Phase

Suppose that the client A would like to communicate with the authentication server S and request services or applications from the application server B. As depicted in Figure 2, the detailed interactions between the three participants are presented as below:

Step 1. $A \rightarrow S: < ID_A, ID_B, U_{A1}, U_{A2}, V_A, QID_{A1} >$

The client A with identity ID_A performs the following off-line computations in advance:

(1) Random select an ephemeral secret $r_A \in Z_q^*$.
(2) Compute $U_{A1} = r_A \cdot P$ and $U_{A2} = r_A \cdot QID_{A2}$.
(3) Compute $W_A = H_2(U_{A1}, U_{A2})$ and $V_A = (r_A + DID_{A1}) \cdot W_A + DID_{A2}$.
(4) Send $< ID_A, ID_B, U_{A1}, U_{A2}, V_A, QID_{A1} >$ to the authentication server S.

Step 2. $S \rightarrow B: < ID_A, U_S, U_{SB1}, U_{SB2}, V_{SB} >$

Upon receiving $< ID_A, ID_B, U_{A1}, U_{A2}, V_A, QID_{A1} >$, the authentication server S authenticates the client A by performing the following tasks:

(1) Compute $W_A = H_2(U_{A1}, U_{A2}), h_A = f_1(ID_A, QID_{A1})$ and $QID_{A2} = H_1(ID_A)$.
(2) Check whether $\hat{e}(P, V_A) = \hat{e}(U_{A1} + QID_{A1},\ W_A) \cdot \hat{e}\left(P_{pub},\ h_A \cdot W_A + QID_{A2}\right)$ or not. If the equation holds, then the authentication server S accepts the request. Otherwise, the authentication server S terminates it.
(3) Random select an ephemeral secret $r_S \in Z_q^*$.
(4) Compute $U_S = r_S \cdot P, U_{SB1} = r_S \cdot U_{A1}$ and $U_{SB2} = r_S \cdot U_{A2}$.
(5) Compute $DID_S = s \cdot QID_S, h_{SB} = f_2(ID_A, ID_B, U_S, U_{SB1}, U_{SB2})$ and $V_{SB} = r_S \cdot P_{pub} + h_{SB} \cdot DID_S$.
(6) Send $< ID_A, U_S, U_{SB1}, U_{SB2}, V_{SB} >$ to the application server B.

Step 3. $B \rightarrow S: < ID_B, U_{B1}, U_{B2}, V_B, QID_{B1} >$

Upon receiving $< ID_A, U_S, U_{SB1}, U_{SB2}, V_{SB} >$, the application server B authenticates the authentication server S by performing the following tasks:

(1) Compute $h_{SB} = f_2(ID_A, ID_B, U_S, U_{SB1}, U_{SB2})$.
(2) Check whether $\hat{e}(V_{SB}, P) = \hat{e}\left(U_S + h_{SB} \cdot QID_S,\ P_{pub}\right)$ or not. If the equation holds, then the application server B accepts the request. Otherwise, the application server B terminates it.
(3) Random select an ephemeral secret $r_B \in Z_q^*$.
(4) Compute $U_{B1} = r_B \cdot P$ and $U_{B2} = r_B \cdot QID_{B2}$.
(5) Compute $W_B = H_2(U_{B1}, U_{B2})$ and $V_B = (r_B + DID_{B1}) \cdot W_B + DID_{B2}$.
(6) Send $< ID_B, U_{B1}, U_{B2}, V_B, QID_{B1} >$ to the authentication server S.

Step 4. $S \rightarrow A: < N_A, Auth_{SA}, ID_B, U_S, U_{SA1}, U_{SA2} >, < N_B, Auth_{SB} >$

Upon receiving $< ID_B, U_{B1}, U_{B2}, V_B, QID_{B1} >$, the authentication server S authenticates the application server B by performing the following tasks:

(1) Compute $W_B = H_2(U_{B1}, U_{B2}), h_B = f_1(ID_B, QID_{B1})$ and $QID_{B2} = H_1(ID_B)$.

(2) Check whether $\hat{e}(P, V_B) = \hat{e}(U_{B1} + QID_{B1}, W_B) \cdot \hat{e}\left(P_{pub}, h_B \cdot W_B + QID_{B2}\right)$ or not. If the equation holds, then the authentication server S accepts the application server B. Otherwise, the authentication server S terminates it.
(3) Compute $U_{SA1} = r_S \cdot U_{B1}$ and $U_{SA2} = r_S \cdot U_{B2}$.
(4) Acquire two nonce N_A, N_B.
(5) Compute $K_{SA} = s \cdot U_{A2}$ and $Auth_{SA} = f_3(ID_A, ID_B, U_{A1}, U_{A2}, V_A, N_A, K_{SA}, U_S, U_{SA1}, U_{SA2})$.
(6) Compute $K_{SB} = s \cdot U_{B2}$ and $Auth_{SB} = f_3(ID_A, ID_B, U_{B1}, U_{B2}, V_B, N_B, K_{SB}, U_S, U_{SB1}, U_{SB2})$.
(7) Send $< N_A, Auth_{SA}, ID_B, U_S, U_{SA1}, U_{SA2} >$ and $< N_B, Auth_{SB} >$ to the client A.

Step 5. $A \rightarrow B: < N_B, Auth_{SB}, Auth_{AB} >$

Upon receiving $< N_A, Auth_{SA}, ID_B, U_S, U_{SA1}, U_{SA2} >$ and $< N_B, Auth_{SB} >$, the client A authenticates the authentication server S by performing the following tasks:

(1) Compute $K_{AS} = r_A \cdot DID_{A2}$.
(2) Check whether $Auth_{SA} = f_3(ID_A, ID_B, U_{A1}, U_{A2}, V_A, N_A, K_{AS}, U_S, U_{SA1}, U_{SA2})$ or not. If the equation holds, then the client A accepts the authentication server S. Otherwise, the client A terminates it.
(3) Compute $K_{AB} = \hat{e}\left(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\right)$ and $Auth_{AB} = f_4(ID_A, ID_B, U_S, K_{AB})$.
(4) Send $< N_B, Auth_{SB}, Auth_{AB} >$ to the application server B.

Step 6. $B \rightarrow A: < Auth_{BA} >$

Upon receiving $< N_B, Auth_{SB}, Auth_{AB} >$, the application server B authenticates the client A and the authentication server S by performing the following tasks:

(1) Compute $K_{BS} = r_B \cdot DID_{B2}$.
(2) Check whether $Auth_{SB} = f_3(ID_A, ID_B, U_{B1}, U_{B2}, V_B, N_B, K_{BS}, U_S, U_{SB1}, U_{SB2})$ or not. If the equation holds, then the application server B accepts the authentication server S. Otherwise, the application server B terminates it.
(3) Compute $K_{BA} = \hat{e}\left(U_{SB1} + U_{SB2}, r_B \cdot P_{pub} + K_{BS}\right)$.
(4) Check whether $Auth_{AB} = f_4(ID_A, ID_B, U_S, K_{BA})$ or not. If the equation holds, then the application server B can be sure that the client A has the ability to compute the session key. Otherwise, the application server B notifies the authentication server S that the authentication has been failed and terminates it.
(5) Compute $Auth_{BA} = f_5(ID_A, ID_B, U_S, K_{BA}, Auth_{AB})$.
(6) Compute the session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{BA}, Auth_{AB}, Auth_{BA})$.
(7) Send $< Auth_{BA} >$ to the client A.

On the other side, upon receiving $< Auth_{BA} >$, the client A checks whether $Auth_{BA} = f_5(ID_A, ID_B, U_S, K_{AB}, Auth_{AB})$ or not. If the equation holds, the client A can be sure that the application server B has the ability to compute the session key, then the client A computes the session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$. Otherwise, the client A notifies the authentication server S that the authentication has been failed and terminates it. After transferring the messages in the above six steps, the client A and the application server B can authenticate each other via the authentication server S and agree upon the session key. The authentication server S also can compute $K_{AB} = \hat{e}(s \cdot U_{A1} + K_{SA}, U_{SA1} + U_{SA2})$ or $K_{BA} = \hat{e}(U_{SB1} + U_{SB2}, s \cdot U_{B1} + K_{SB})$ to compute the session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$ which is shared with the client A and the application server B. The session key is created by the three participants and not created by the authentication server S alone. Therefore, the proposed protocol can solve the problem in the resilience of key control of the KTAP while the authentication server S still can monitor the communication messages to prevent and trace network crime.

In Steps 2 and 4, the correctness of the equations that the authentication server S uses to authenticate the client A and the application server B can be proved by the bilinear pairings feature operation. To the client A as an example, the correctness of $\hat{e}(P, V_A) = \hat{e}(U_{A1} + QID_{A1}, W_A) \cdot \hat{e}\left(P_{pub}, h_A \cdot W_A + QID_{A2}\right)$ in Step 2 is presented as follows:

$$\begin{aligned}
\hat{e}(P, V_A) &= \hat{e}(P, (r_A + DID_{A1}) \cdot W_A + DID_{A2}) \\
&= \hat{e}(P, (r_A + l_A + h_A \cdot s) \cdot W_A + s \cdot QID_{A2}) \\
&= \hat{e}(P, (r_A + l_A) \cdot W_A + h_A \cdot s \cdot W_A + s \cdot QID_{A2}) \\
&= \hat{e}(P, (r_A + l_A) \cdot W_A + s \cdot (h_A \cdot W_A + QID_{A2})) \\
&= \hat{e}(P, (r_A + l_A) \cdot W_A) \cdot \hat{e}(P, s \cdot (h_A \cdot W_A + QID_{A2})) \\
&= \hat{e}((r_A + l_A) \cdot P, W_A) \cdot \hat{e}(s \cdot P, h_A \cdot W_A + QID_{A2}) \\
&= \hat{e}(U_{A1} + QID_{A1}, W_A) \cdot \hat{e}\left(P_{pub}, h_A \cdot W_A + QID_{A2}\right)
\end{aligned}$$

In Step 3, the correctness of the equation that the application server B uses to verify the message sent from the authentication server S. The correctness of $\hat{e}(V_{SB}, P) = \hat{e}\left(U_S + h_{SB} \cdot QID_S, P_{pub}\right)$ is presented as follows:

$$\begin{aligned}
\hat{e}(V_{SB}, P) &= \hat{e}\left(r_S \cdot P_{pub} + h_{SB} \cdot DID_S, P\right) \\
&= \hat{e}(r_S \cdot s \cdot P + h_{SB} \cdot s \cdot QID_S, P) \\
&= \hat{e}(r_S \cdot P + h_{SB} \cdot QID_S, s \cdot P) \\
&= \hat{e}\left(U_S + h_{SB} \cdot QID_S, P_{pub}\right)
\end{aligned}$$

In Steps 5 and 6, the equations $Auth_{SA}$ and $Auth_{SB}$ that the client A and the application server B use to authenticate the authentication server S, respectively. If $K_{SA} = K_{AS}$ and $K_{SB} = K_{BS}$, we say that $Auth_{SA}$ and $Auth_{SB}$ are valid, the equations hold because of $K_{SA} = s \cdot U_{A2} = s \cdot r_A \cdot QID_{A2} = r_A \cdot DID_{A2} = K_{AS}$ and $K_{SB} = s \cdot U_{B2} = s \cdot r_B \cdot QID_{B2} = r_B \cdot DID_{B2} = K_{BS}$.

In Step 6, if $K_{AB} = K_{BA}$, we say that $Auth_{AB}$ and $Auth_{BA}$ are valid. The client A, the application server B and the authentication server S have established a common session key SK_{AB}. The correctness of $K_{AB} = K_{BA}$ is presented as follows:

$$\begin{aligned}
K_{AB} &= \hat{e}\left(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\right) \\
&= \hat{e}(r_A \cdot s \cdot P + r_A \cdot s \cdot QID_{A2}, r_S \cdot r_B \cdot P + r_S \cdot r_B \cdot QID_{B2}) \\
&= \hat{e}(r_A \cdot P + r_A \cdot QID_{A2}, r_B \cdot P + r_B \cdot QID_{B2})^{s \cdot r_S} \\
&= \hat{e}(r_S \cdot r_A \cdot P + r_S \cdot r_A \cdot QID_{A2}, s \cdot r_B \cdot P + s \cdot r_B \cdot QID_{B2}) \\
&= \hat{e}(r_S \cdot U_{A1} + r_S \cdot U_{A2}, r_B \cdot s \cdot P + r_B \cdot DID_{B2}) \\
&= \hat{e}\left(U_{SB1} + U_{SB2}, r_B \cdot P_{pub} + K_{BS}\right) \\
&= K_{BA}
\end{aligned}$$

3.4. The Parallel Version

Chen et al. [31] pointed out that to enhance the performance and reduce latency of an AKA protocol, the communication steps in the protocol should be as parallel as possible. To enhance the proposed protocol performance, a parallel version of the protocol is presented. We reordered the steps in the proposed protocol as shown in Figure 3. It can be obviously seen that the steps are reordered but the message exchange contents are the same as those in the preceding protocol. In the parallel version of the proposed protocol, the protocol can be executed in four rounds.

3.5. Preventing and Tracing Network Crime

In a distributed computing environment preventing and tracing network crime often consists of mechanisms to prevent, detect and deter security violations that involve the transmission of messages. Authentication, Authorization, Accounting (AAA) services are the commonly used mechanisms to protect the security of networks in distributed computing environments [32–34]. The authentication service is the first inspection mechanism that prevents and traces network crime. The authorization service is based on the security primitives of the authentication service. The authentication service and the authorization service are usually performed together to ensure that a client requesting access to the services is in fact the client to whom entry is authorized. It is able to prevent services access by a client that is not authorized and trace which services were accessed by which client. After authentication and authorization, a common session key is constructed using symmetric encryptions. These are most effective techniques to protect transferred messages from being intercepted, disclosed and forged by an adversary. The AAA services also provide accounting services that are used to collect resources usage information for billing. In the generic AAA services, the authentication, authorization and accounting services are provided by an authentication server referred to as AAA server.

The proposed protocol is an ID-based three-party AKA protocol that provides mutual authentication and establishes a common session key between three participants. The authentication server in the proposed protocol is responsible for the authentication of participating entities and restricts access to authorized clients. Both clients and the application servers must send their identities to the authentication server and retrieve the private keys computed by the authentication server in the key extract phase. A client or an adversary without the private key is unable to request and access the services provided by an application server. An application server without the private key is unable to be accepted by the authentication server and the clients. The authentication server needs to authenticate the clients and the application servers but also needs to be authenticated by the clients and the application servers. The proposed protocol is able to prevent unauthorized access and detect, deter an adversary who constructs a forged identity as the real authentication server or application server to acquire useful information from the clients.

The proposed protocol allows three participants to establish a common session key to protect the transferred messages and also ensures the authenticity. The common session key of the proposed protocol is derived from the authentication server, the client and the application server's private keys and their ephemeral secrets in the mutual authentication and key agreement phase. The three participants are able to ensure that no other participants or adversaries can learn or determine the value of the common session key and the authentication server could use the common session key to monitor the transferred messages. The common session key of the proposed protocol is able to protect and guarantee the authenticity of the transferred messages and also trace which client has accessed the services provided by which application server. None of the clients are able to deny that the client has requested and accessed the services provided by the application server.

In the proposed protocol the authentication server provides authentication service, authorization service and uses the common session key to prevent and trace network crime. However, the authentication server does not provide accounting services. The most important aspect in the AAA services is the authentication service and the purpose of the accounting service is to collect information on resource usage for billing, auditing and trend analysis [35–38]. Hence, the proposed protocol is suitable for various communication systems in the distributed computing environments in terms of protecting the security of network, preventing and tracing network crime.

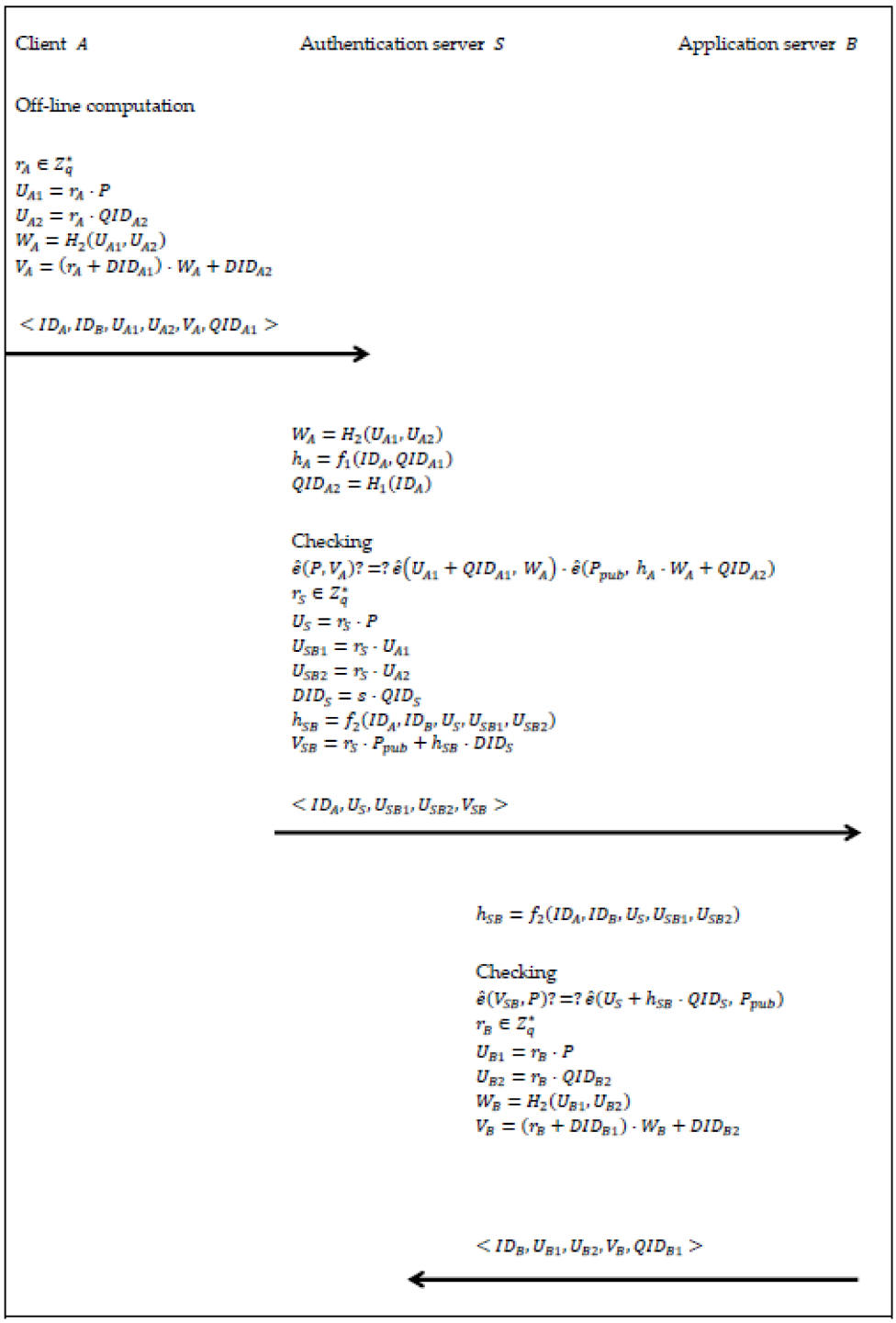

Figure 2. *Cont.*

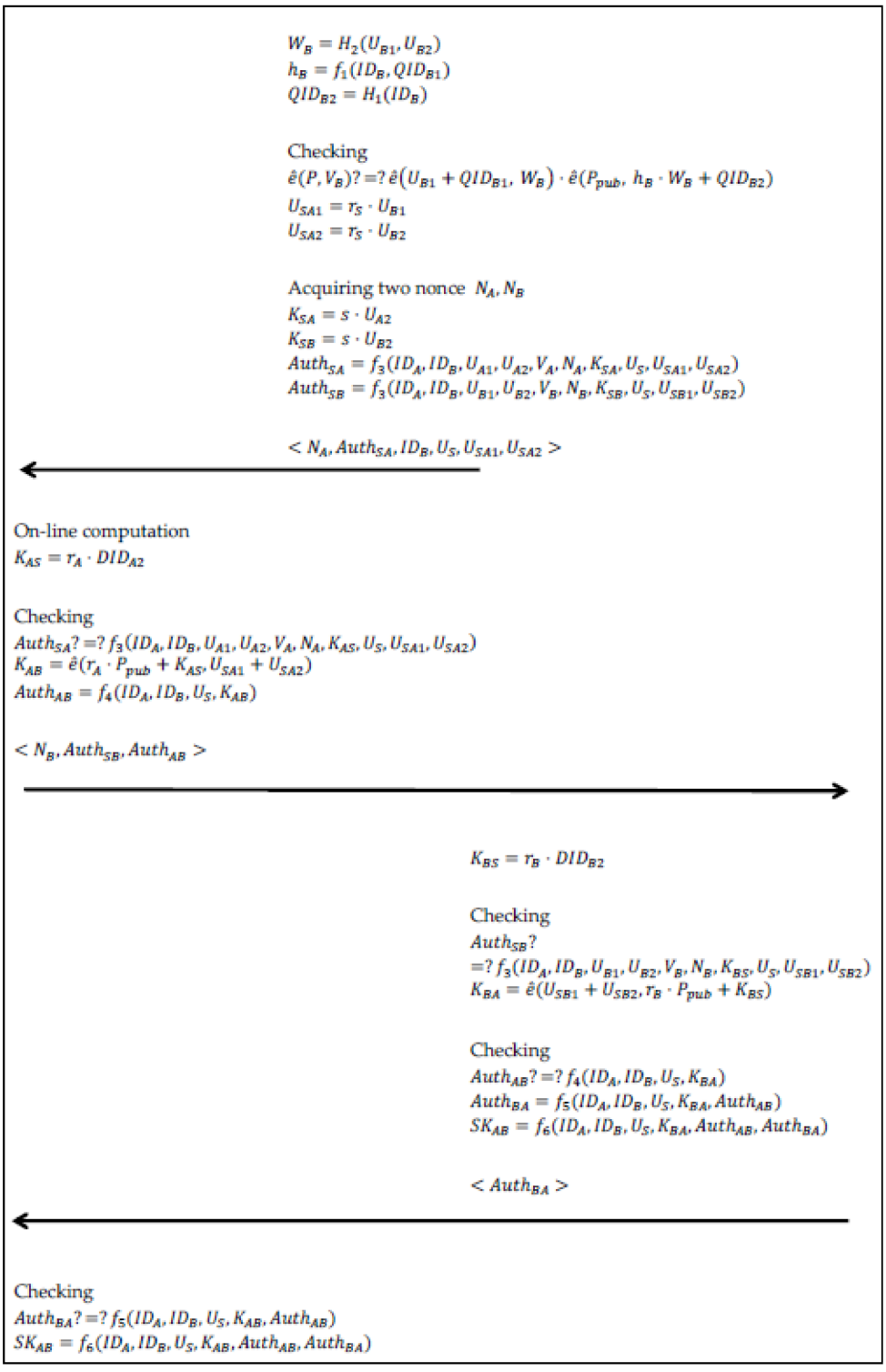

Figure 2. The mutual authentication and key agreement phase.

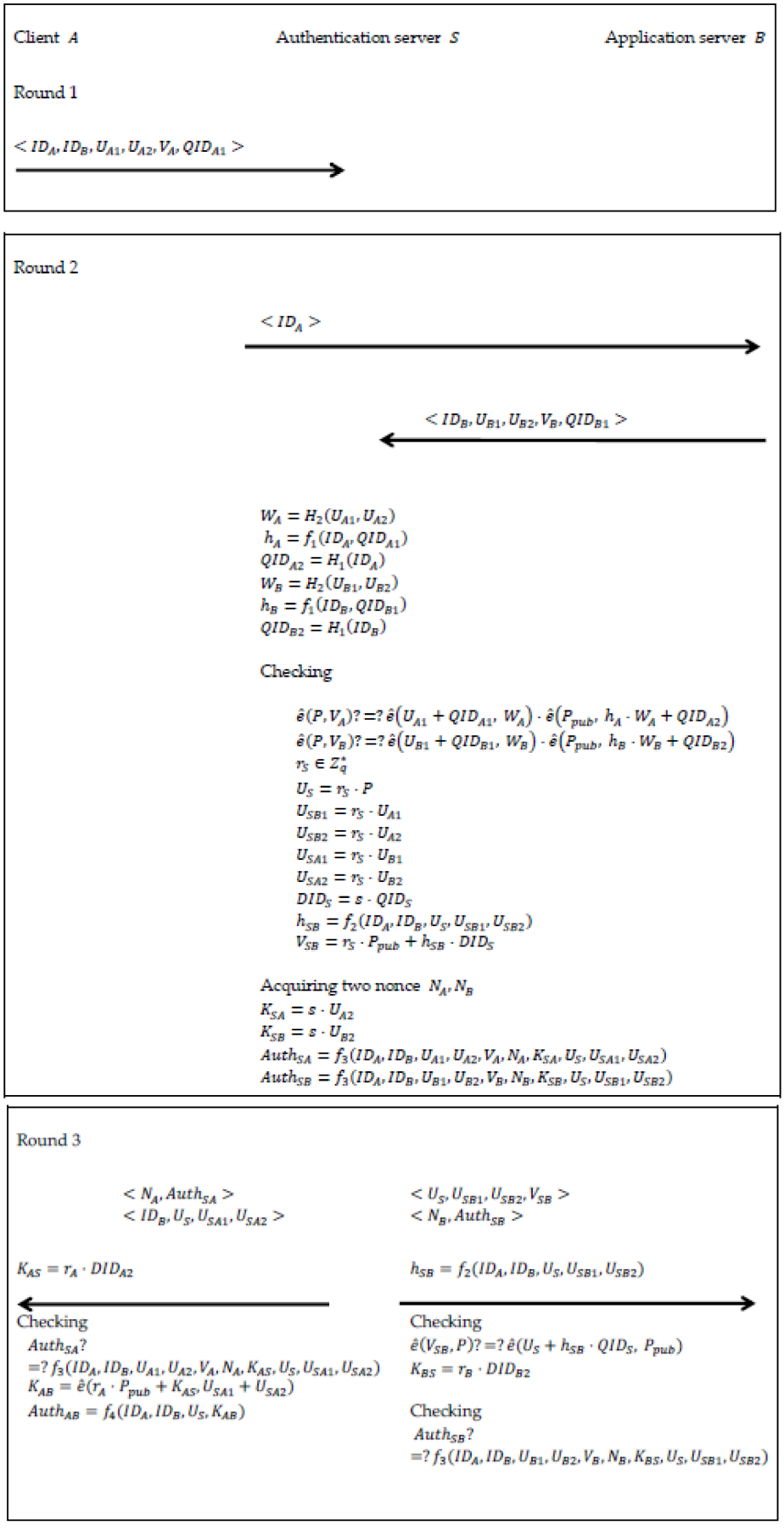

Figure 3. *Cont.*

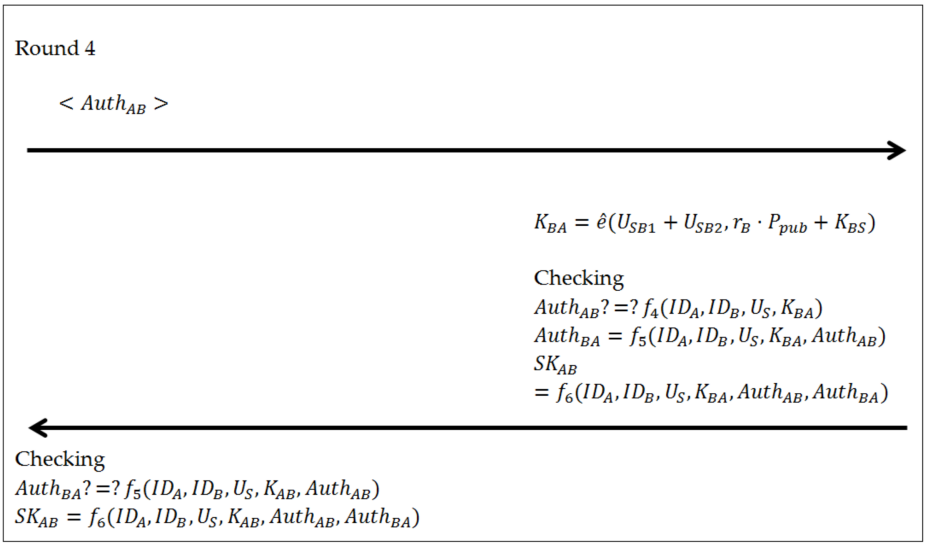

Figure 3. The parallel version of Ephemeral Secret Leakage (ESL)-secure ID-based three-party Authenticated Key Agreement (AKA) protocol.

4. Security and Performance Analysis

In this Section we discuss the security analysis of the proposed protocol. The simulation results were performed using the AVISPA tool. The methodology used in our performance analysis and the metrics utilized to evaluate the results of the proposed protocol.

4.1. Security Analysis

In this Section we present the security analysis of the proposed protocol to show the security attributes required for AKA protocols are satisfied and the simulation results by the AVISPA tool.

- **Mutual authentication and Ephemeral-Secret-Leakage Resistance.** The proposed protocol adopted Tseng et al.'s ESL-secure ID-AKE protocol [7] to achieve the client-to-server authentication. The authentication server S authenticates the client A and the application server B by checking whether $\hat{e}(P, V_A) = \hat{e}(U_{A1} + QID_{A1}, W_A) \cdot \hat{e}\left(P_{pub}, h_A \cdot W_A + QID_{A2}\right)$ and $\hat{e}(P, V_B) = \hat{e}(U_{B1} + QID_{B1}, W_B) \cdot \hat{e}\left(P_{pub}, h_B \cdot W_B + QID_{B2}\right)$ or not. To the client A as an example, the message $< ID_A, ID_B, U_{A1}, U_{A2}, V_A, QID_{A1} >$, where $W_A = H_2(U_{A1}, U_{A2})$, $h_A = f_1(ID_A, QID_{A1})$ and $QID_{A2} = H_1(ID_A)$ can be viewed as a signature on a message U_{A2} in Tseng et al.'s ESL-secure ID-AKE protocol. According to the security analysis by Tseng et al., if an adversary could have obtained the ephemeral secret r_A using ESL attacks, the adversary still needs to solve the computational Diffie-Hellman problem to violate the client-to-server authentication.

  The application server B authenticates the authentication server S by checking whether $\hat{e}(V_{SB}, P) = \hat{e}\left(U_S + h_{SB} \cdot QID_S, P_{pub}\right)$ or not. The message V_{SB}, where $V_{SB} = r_S \cdot P_{pub} + h_{SB} \cdot DID_S$, $h_{SB} = f_2(ID_A, ID_B, U_S, U_{SB1}, U_{SB2})$ and $DID_S = s \cdot QID_S$, can be viewed as a signature on a message $< ID_A, U_S, U_{SB1}, U_{SB2} >$. Without knowledge of the authentication server S's private key s, none of the participants or adversaries can forge the message and compute a valid signature. To forge a valid message $< ID_A, U_S, U_{SB1}, U_{SB2}, V_{SB} >$, an adversary must have obtained the authentication server S's private key s from P_{pub}, where $P_{pub} = s \cdot P$. It is a discrete logarithm problem to the adversary.

  The proposed protocol provides the server-to-client authentication that is also based on Tseng et al.'s ESL-secure ID-AKE protocol. The client A and the application server B authenticate the authentication

server S by checking whether $Auth_{SA} = f_3(ID_A, ID_B, U_{A1}, U_{A2}, V_A, N_A, K_{AS}, U_S, U_{SA1}, U_{SA2})$ and $Auth_{SB} = f_3(ID_A, ID_B, U_{B1}, U_{B2}, V_B, N_B, K_{BS}, U_S, U_{SB1}, U_{SB2})$ or not. Even though an adversary could obtain the client A's ephemeral secret r_A by the ESL attacks. Since $< Auth_{SA} >$ and $< Auth_{SB} >$ are derived from $K_{SA} = s \cdot U_{A2} = r_A \cdot DID_{A2} = K_{AS}$ and $K_{SB} = s \cdot U_{B2} = r_B \cdot DID_{B2} = K_{BS}$, respectively. To violate the server-to-client authentication, the adversary has to solve the computational Diffie-Hellman problem to obtain DID_{A2} and the discrete logarithm problem to compute the application server B's ephemeral secret r_B and the computational Diffie-Hellman problem to obtain DID_{B2} from the transferred messages. Otherwise, the adversary only can compute the authentication server S's private key s from P_{pub}, in which the adversary needs to solve the discrete logarithm problem.

The application server B can be sure that the client A has obtained the session key by checking whether $Auth_{AB} = f_4(ID_A, ID_B, U_S, K_{BA})$ or not. The client A checks whether $Auth_{BA} = f_5(ID_A, ID_B, U_S, K_{AB}, Auth_{AB})$ or not to be sure that the application server B also has obtained the session key and both of them with the authentication server S can agree upon the common session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$. For computing a valid message $< Auth_{AB} >$ or $< Auth_{BA} >$, we assume that the client A's ephemeral secret r_A has been compromised by an adversary. Since the messages $< Auth_{AB} >$ and $< Auth_{BA} >$ are derived from $K_{AB} = \hat{e}\Big(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\Big) = \hat{e}\Big(U_{SB1} + U_{SB2}, r_B \cdot P_{pub} + K_{BS}\Big) = K_{BA}$, where $K_{AS} = r_A \cdot DID_{A2}$ and $K_{BS} = r_B \cdot DID_{B2}$, the adversary must solve the computational Diffie-Hellman problem to compute DID_{A2} or the discrete logarithm problem to compute r_B and the computational Diffie-Hellman problem to compute DID_{B2} from transferred messages. Otherwise, the adversary only can compute the authentication server S's private key s from P_{pub}, in which the adversary needs to solve the discrete logarithm problem.

Without knowledge of the private key of a participant (e.g., the client A's private key DID_A). An adversary cannot impersonate the participant since the adversary is unable to forge a valid signature. The proposed protocol employs the signature to authenticate the participants' identities and one way hash function to protect the integrity of the transferred messages. Even though the ephemeral secret of the client has been compromised, each participant can be sure that none of the adversaries can impersonate other participants to violate the verification procedures and corrupt the participants' private keys and the session key.

- **Known Key Security.** Suppose that an adversary can eavesdrop on the transmitted messages to learn the previous session keys. However, the session key of the proposed protocol is unique and dependent of each participant's ephemeral secrets r_A, r_B and r_S and private keys DID_{A2}, DID_{B2} and s. Therefore, knowledge of the previous session keys does not enable the adversary to derive other session keys and does not give the adversary any information that the adversary could use to derive other session keys. Even though the client A's ephemeral secret r_A has been compromised. If an adversary would like to compute $K_{AB} = \hat{e}\Big(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\Big)$, where $K_{AS} = r_A \cdot DID_{A2}$ from the transferred messages, the adversary needs to solve the computational Diffie-Hellman problem to compute the client A's private key DID_{A2} or the discrete logarithm problem from P_{pub}, where $P_{pub} = s \cdot P$ to obtain the authentication server S's private key s. Otherwise, the adversary only can solve the discrete logarithm problem and the computational Diffie-Hellman problem to compute the application server B's corresponding ephemeral secret r_B and private key DID_{B2}. In the proposed protocol, even if one of the session key has been compromised, the security of the other or future session keys is not endangered.
- **Partial Forward Secrecy.** The proposed protocol session key is dependent on each participant's private key and corresponding ephemeral secret. In the proposed protocol, if the private key of client A or the application server B has been compromised, an adversary also needs to obtain the corresponding ephemeral secrets to compute the session key. Suppose that the adversary would like to compute the corresponding ephemeral secrets from the transferred messages, the adversary

needs to solve the discrete logarithm problem. However, if the adversary corrupts the private key s of the authentication server S, it is obvious that all of the previous session keys can be recovered from the transferred messages. Since the adversary is indeed able to compute $K_{SA} = s \cdot U_{A2}$, $K_{AB} = \hat{e}(s \cdot U_{A1} + K_{SA}, U_{SA1} + U_{SA2})$ or $K_{SB} = s \cdot U_{B2}$, $K_{BA} = \hat{e}(U_{SB1} + U_{SB2}, s \cdot U_{B1} + K_{SB})$ and $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$. The proposed protocol offers partial forward secrecy.

- **Key-Compromise Impersonation Resilience.** To discuss the key-compromise impersonation resilience property, we assume that client A's private key DID_A is compromised to an adversary who tries to impersonate the application server B to cheat the client A and the authentication server S. When the client A requests service from the application server B, the adversary only can choose $< ID_B, U_{B1}, U_{B2}, V_B, QID_{B1} >$ and $< Auth_{BA} >$ from the previous sessions and send them to the authentication server S and the client A, respectively. Since the adversary cannot derive the application server B's private key DID_{B2} and the corresponding ephemeral secret r_B from the transferred messages to compute $K_{BA} = \hat{e}\Big(U_{SB1} + U_{SB2}, r_B \cdot P_{pub} + K_{BS}\Big)$, where $K_{BS} = r_B \cdot DID_{B2}$. The adversary only can violate the verification procedures of the authentication server S. If the adversary tries to derive the ephemeral secret r_B and the private key DID_{B2} from the transferred messages to compute $Auth_{BA} = f_5(ID_A, ID_B, U_S, K_{BA}, Auth_{AB})$ and construct the session key, the adversary needs to solve the discrete logarithm problem and the computational Diffie-Hellman problem.
- **Unknown Key-Share Resilience.** To implement such an attack, the adversary is required to obtain the private key of client A or the application server B. In the proposed protocol, both of the client A and the application server B have to be authenticated by the authentication server S. Only the participant that has the private key distributed from the authentication server S could compute the valid signature and thus pass the verification procedures and compute the session key. To the client A as an example, the client A with its private key DID_A and the ephemeral secret r_A can compute $V_A = (r_A + DID_{A1}) \cdot W_A + DID_{A2}$ and $K_{AB} = \hat{e}\Big(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\Big)$ to pass the verification procedures of the authentication server S and the application server B. The ephemeral secret r_A with DID_{A2} are able to compute the session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$. Hence, the proposed protocol can withstand an unknown key-shared attack.
- **Key control Resilience.** In the proposed protocol, the session key $SK_{AB} = f_6(ID_A, ID_B, U_S, K_{AB}, Auth_{AB}, Auth_{BA})$, where $K_{AB} = \hat{e}\Big(r_A \cdot P_{pub} + K_{AS}, U_{SA1} + U_{SA2}\Big) = \hat{e}\Big(U_{SB1} + U_{SB2}, r_B \cdot P_{pub} + K_{BS}\Big) = K_{BA}$ is determined by all participants' private keys and corresponding ephemeral secrets. None of the participants can force a session key to be predetermined or predict the value and control the outcome of the session key. Hence, the proposed protocol ability to prevent the session key is created only by the authentication server S or other two participants.

4.2. Formal Analysis Using AVISPA

Besides the above analysis we also provide a formal analysis of the proposed protocol using AVISPA. AVISPA is a push-button tool which is one of the commonly used automated security validation tools for Internet security-sensitive protocols and applications [19]. AVISPA was developed based on the Dolev-Yao intruder model [35]. In this model, the intruder has full control over the network and the intruder can intercept, inject, analyze and modify messages in transit. In addition, the intruder can play the role of a legitimate participant and gain knowledge of the compromised participant, but he is not allowed to crack the underlying cryptography. The first step in using AVISPA is to implement the analyzed protocol in High-Level Protocol Specification Language (HLPSL) that is an expressive, modular, role-based formal language for security protocol description and specifying their security properties. The HLPSL presentation of the analyzed protocol is translated into a lower

level language called Intermediate Format (IF) using HLPSL2IF. IF the analyzed protocol is used as an input to different back-ends, the current version of AVISPA tool comprises four back-ends that implement a variety of automatic analysis techniques, namely, On-the-fly Model-Checker (OFMC), Constraint-Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) [30]. The four back-ends perform the analysis and output the results in precisely defined output format stating whether problems exist in the protocol or not. Thus, AVISPA is appropriate for the analysis of large-scale security protocols and applications.

To evaluate the proposed protocol security using AVISPA, we implemented the proposed protocol using the HLPSL and the role specifications of the client, the application server and the authentication server are given in Appendix A. We simulated the proposed protocol using the AVISPA web tool [36]. The proposed protocol is analyzed in the OFMC and CL-AtSe back-ends. Both of the two back-ends are helpful for the verification of the proposed protocol and detection of attacks. Note that intruder knowledge of the proposed protocol comprises the ephemeral secret r_A of the client. After executing the code in AVISPA web tool, both OFMC and CL-AtSe back-ends outputs were generated and shown in Figures 4 and 5. From the simulated results, there are no major attacks to the protocol and the simulated results also confirm the correctness of the protocol security analysis. The security goals of the proposed protocol are achieved and thus the proposed protocol confirms security against various active and passive attacks.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfile6RQKh2.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 1.14s
  visitedNodes: 89 nodes
  depth: 7 plies
```

Figure 4. Simulation result of the proposed protocol on On-the-fly Model-Checker (OFMC) model checker.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfile6RQKh2.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed    : 0 states
  Reachable   : 0 states
  Translation: 0.12 seconds
  Computation: 0.00 seconds
```

Figure 5. Simulation result of the proposed protocol on Constraint-Logic-based Attack Searcher (CL-AsSe) model checker.

4.3. Performance Analysis

In this section, the performance and the simulation results of the proposed protocol are analyzed. For convenience, the following notations are used to analyze the computational cost:

- TG_e: the execution time for a bilinear pairing operation $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
- TG_{mul}: the execution time for points in G_1 multiplication operation.
- T_{exp}: the execution time for a modular exponential operation in G_2.
- TG_H: the execution time for a map-to-point hash function in G_1.
- TG_{add}: the execution time for an addition operation of points in G_1 or a multiplication operation in G_2.
- T_H: the execution time for a one-way hash function.

The total computational cost for the client side in this proposed protocol is $TG_e + 5TG_{mul} + TG_H + 3TG_{add} + 4T_H$. However, the computation in Step 1 of the mutual authentication and key agreement phase for the client side can be pre-computed using offline computation. The total cost for on-line computation on the client side is $TG_e + 2TG_{mul} + 2TG_{add} + 4T_H$. The total computational cost for the application server side is $3TG_e + 6TG_{mul} + TG_H + 4TG_{add} + 5T_H$. On the other hand, the authentication server side performs Steps 2 and 4 to authenticate the client, the application server and constructs the session key. The total computational cost for the authentication server side is $7TG_e + 13TG_{mul} + 4TG_H + 7TG_{add} + 8T_H$.

Note that compared with TG_{add} and T_H, TG_e, TG_{mul}, T_{exp} and TG_H are more time-consuming [31]. The proposed protocol adopts an imbalanced computation technique and an online/offline computation technique to reduce the computational load for the client side. From the analysis result mentioned above, the computational burdens are major on the authentication server and the application server which are powerful servers and the offline pre-computation reduces the computational cost required for the client side. Although, bilinear pairing operations are still required for the client side, the proposed protocol captures all basic desirable security attributes including the ESL resistance and the authentication server is able to monitor the transferred messages to prevent and trace network crime. The required computation cost of the proposed protocol is also reasonable.

4.4. Software Performance

Scott et al. [37] and Oliveira et al. [38] implemented the related pairing-based operations for low-power computing devices (i.e., smartcards and sensor nodes) in 2006 and 2011, respectively. Scott et al. used the processor on the Philips HiPersmart card offers the maximum clock speed of 36 MHz for related pairing-based operations. Scott et al. pointed out that for the security level of the Ate pairing system, a popular and valid choice would be to use a supersingular curve or non-supersingular curve over a finite field $E(F_p)$, with $p = 512$ bits and a large prime order $q = 160$ bits. The simulations are performed with a simulator implemented in Android system. We used the Jpair library to execute the simulations, which is a Java implementation of bilinear pairings to implement the software and a mobile device (Android 5.0 and 2.3 GHz Intel Atom with 4 GB of RAM) for the client side. Since the Weil pairing evaluation is more time-consuming than the Tate pairing [39], simulations are provided based on the Tate pairing on a supersingular elliptic curve. In the proposed protocol security level, the parameters will be the same as Scott et al.'s experimental data mentioned above. Table 1 lists the simulations data for related pairing-based operations on the client side. The execution time results for the client side have an average of 1000 simulations. Table 2 lists the computational cost and the execution time (in seconds) on the client side, where the execution time is measured using Table 1. The proposed protocol adopts an online/offline computation technique in which the online execution time on the client side requires 0.553 s. Although, the bilinear pairing operations are still required for the client side, the proposed protocol is still efficient and suitable for mobile distributed computing environments.

Table 1. Computational cost on the client side.

	TG_e	TG_{mul}	T_{exp}	TG_H	TG_{add}	T_H
Android 5.0 and 2.3 GH_z Intel Atom with 4 GB of RAM	0.251 s	0.148 s	0.076 s	0.002 s	0.001 s	0.001 s

Table 2. Execution time on the client side.

	Computational Cost (Total)	Computational Cost (Online)	Execution Time (Total)	Execution Time (Online)
Client Side	$TG_e + 5TG_{mul} + TG_H + 3TG_{add} + 4T_H$	$TG_e + 2TG_{mul} + 2TG_{add} + 4T_H$	$\coloneqq$ 1.0 s	$\coloneqq$ 0.553 s

5. Conclusions

We proposed an ESL-secure ID-based three-party AKA protocol for mobile distributed computing environments. The proposed protocol employs Tseng et al.'s ESL-secure ID-AKE protocol [7] against ESL attacks. The proposed protocol keeps all of the merits of KTAP regarding security including the authentication server is able to monitor the communication messages to prevent and trace network crime and solves the problems of KTAP in which the session key is only created by the authentication server. Each participant can contribute information to derive the common session key. In the security analysis, the proposed protocol resists ESL attacks and also satisfies the security attributes required for AKA protocols: known-key security, partial forward secrecy, key-compromise impersonation resilience, unknown key-share resilience and key control resilience. In addition, the formal validation of the proposed protocol is performed using an automated validation tool AVISPA. The simulations results show that the proposed protocol is secure against active and passive adversaries. In the performance analysis, the proposed protocol adopted an imbalanced computation technique to shift the computational burden to the powerful servers. The software performance simulation result shows that the mobile client could perform offline pre-computation to reduce the online computation cost. Furthermore, a parallel version of the proposed protocol is proposed to enhance the protocol run performance.

In the future, we will implement the proposed in the "To Say" APP. The "To Say" APP allows users store the last words (text, image, audio, and video) in the application server and send the last words to the designated receivers in a secure channel. The authentication server is responsible for authenticate the users and helps them to construct a common session key with the receivers. When the APP detects the users are not active, the server will automatically send the last words to the designated receivers.

Acknowledgments: This study was supported by the National Science Council of Taiwan under grant MOST 106-2221-E-018-001 and MOST 106-2622-E-018-002-CC3.

Author Contributions: The authors contributed equally to this work.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix

The HLPSL code of the proposed protocol.

```
role client (A, S, B : agent,
             P, Ppub : symmetric_key,
             Union, Pred, Ebilinear : hash_func,
             Ra, Rs, Rb : text,
             SND_SA, RCV_SA, SND_BA, RCV_BA : channel(dy))
played_by A
def =
local State : nat,
              Na, Nb : text,
              Ida, Qida1, Qida2, Ua1, Ua2, Wa, Va, Authab, Ids, Us, Usa1, Usa2, Usb1, Usb2,
              Hsb, Vsb, Authsa, Authsb, Idb, Qidb1, Qidb2, Ub1, Ub2, Wb, Vb, Authba : message,
              Ss, Dida1, Dida2, Kab, Ksa, Ksb, Kba, Didb1, Didb2, Dids, Skab : symmetric_key
              const dida1, dida2, didb1, didb2, dids, ksa, ksb, kab, kba, skab : protocol_id
init State := 0
transition
1. State = 0    /\ RCV_BA (start) =|>
```

Figure A1. Role specification of the client in HLPSL.

```
   State' := 1   /\ Ra' := new()
                 /\ Ua1' := Pred (Ra, P)
                 /\ Ua2' := Pred (Ra, Qida2)
                 /\ Wa' := Pred (Ua1, Ua2)
                 /\ Va' := Union (Pred (Pred (Ra, Dida1), Wa), Dida2)
                 /\ SND_SA (A. S. Ida. Idb. Ua1. Ua2. Va. Qida1)
                 /\ secret (Dida1, dida1, {A, S})
                 /\ secret (Dida2, dida2, {A, S})
                 /\ witness (A, S, a_s_dida1, Dida1)
                 /\ witness (A, S, a_s_dida2, Dida2)
2. State = 8     /\ RCV_SA (S. A. Na'. Authsa'. Idb. Us'. Usa1'. Usa2'. Nb'. Authsb')
                 /\ Ksa' = Pred (Ss, Ua2)
                 /\ Authsa' = Union (Ida, Idb, Ua1, Ua2, Va, Na, Ksa, Us, Usa1, Usa2) =|>
   State' := 9   /\ Kab' := Ebilinear (Pred (Ra, Ppub), Pred (Ra, Dida2), Pred (Usa1, Usa2))
                 /\ Authab' := Union (Ida, Idb, Us, Kab)
                 /\ SND_BA (A. B. Nb. Authsb. Authab)
                 /\ request (S, A, s_a_ksa, Ksa)
                 /\ secret (Kab, kab, {A, B})
                 /\ witness (A, B, a_b_kab, Kab)
3. State = 12    /\ RCV_BA (B. A. Authba')
                 /\ Kba' = Ebilinear (Pred (Usb1, Usb2), Pred (Rb, Ppub), Pred (Rb, Didb2))
                 /\ Authba' = Union (Ida, Idb, Us, Kba, Authab) =|>
   State' := 13  /\ Skab' := Union (Ida, Idb, Us, Kab, Authab, Authba)
                 /\ request (B, A, b_a_kba, Kba)
                 /\ secret (Skab, skab, {A, S, B})
end role
```

Figure A2. *Cont.*

```
role authentication_server (A, S, B : agent,
                            P, Ppub : symmetric_key,
                            Union, Pred, Ebilinear : hash_func,
                            Ra, Rs, Rb : text,
                            SND_AS, RCV_AS, SND_BS, RCV_BS : channel(dy))
played_by S
def =
local State : nat,
                      Na, Nb : text,
                      Ida, Qida1, Qida2, Ua1, Ua2, Wa, Va, Authab, Ids, Us, Usa1, Usa2, Usb1, Usb2,
                      Hsb, Vsb, Authsa, Authsb, Idb, Qidb1, Qidb2, Ub1, Ub2, Wb, Vb, Authba : message,
                      Ss, Dida1, Dida2, Kab, Ksa, Ksb, Kba, Didb1, Didb2, Dids, Skab : symmetric_key
                      const dida1, dida2, didb1, didb2, dids, ksa, ksb, kab, kba, skab : protocol_id
init State := 2
transition
```

```
1. State = 2    /\ RCV_AS (A. S. Ida. Idb. Ua1'. Ua2'. Va'. Qida1) =|>
   State' := 3  /\ Rs' := new()
                /\ Us' := Pred (Rs, P)
                /\ Usb1' := Pred (Rs, Ua1)
                /\ Usb2' := Pred (Rs, Ua2)
                /\ Hsb' := Union (Ida, Idb, Us, Usb1, Usb2)
                /\ Vsb' := Union (Pred (Rs, Ppub), Pred (Hsb, Dids))
                /\ SND_BS (S. B. Ida. Us. Usb1. Usb2. Vsb)
                /\ wrequest (A, S, a_s_dida1, Dida1)
                /\ wrequest (A, S, a_s_dida2, Dida2)
                /\ secret (Dids, dids, {S, B})
                /\ witness (S, B, s_b_dids, Dids)
2. State = 6    /\ RCV_BS (B. S. Idb. Ub1'. Ub2'. Vb'. Qidb1) =|>
   State' := 7  /\ Usa1' := Pred (Rs, Ub1)
                /\ Usa2' := Pred (Rs, Ub2)
                /\ Ksa' := Pred (Ss, Ua2)
                /\ Ksb' := Pred (Ss, Ub2)
                /\ Na' := new()
                /\ Nb' := new()
                /\ Authsa' := Union (Ida, Idb, Ua1, Ua2, Va, Na, Ksa, Us, Usa1, Usa2)
                /\ Authsb' := Union (Ida, Idb, Ub1, Ub2, Vb, Nb, Ksb, Us, Usb1, Usb2)
                /\ SND_AS (S. A. Na. Authsa. Idb. Us. Usa1. Usa2. Nb. Authsb)
                /\ wrequest (B, S, b_s_didb1, Didb1)
                /\ wrequest (B, S, b_s_didb2, Didb2)
                /\ secret (Ss, ss, {S, A})
                /\ secret (Ss, ss, {S, B})
                /\ secret (Ksa, ksa, {A, S})
                /\ secret (Ksb, ksb, {B, S})
                /\ witness (S, A, s_a_ksa, Ksa)
                /\ witness (S, B, s_b_ksb, Ksb)
end role
```

Figure A2. Role specification of the authentication server in HLPSL.

```
role application_server (A, S, B : agent,
                         P, Ppub : symmetric_key,
                         Union, Pred, Ebilinear : hash_func,
                         Ra, Rs, Rb : text,
                         SND_SB, RCV_SB, SND_AB, RCV_AB : channel(dy))
played_by B
```

```
def =
local State : nat,
      Na, Nb : text,
      Ida, Qida1, Qida2, Ua1, Ua2, Wa, Va, Authab, Ids, Us, Usa1, Usa2, Usb1, Usb2,
      Hsb, Vsb, Authsa, Authsb, Idb, Qidb1, Qidb2, Ub1, Ub2, Wb, Vb, Authba : message,
      Ss, Dida1, Dida2, Kab, Ksa, Ksb, Kba, Didb1, Didb2, Dids, Skab : symmetric_key
      const dida1, dida2, didb1, didb2, dids, ksa, ksb, kab, kba, skab : protocol_id
init State := 4
transition
1. State = 4    /\ RCV_SB (S. B. Ida. Us'. Usb1'. Usb2'. Vsb') =|>
   State' := 5  /\ Rb' := new()
                /\ Ub1' := Pred (Rb, P)
                /\ Ub2' := Pred (Rb, Qidb2)
                /\ Wb' := Pred (Ub1, Ub2)
                /\ Vb' := Union (Pred (Pred (Rb, Didb1), Wb), Didb2)
                /\ SND_SB (B. S. Idb. Ub1. Ub2. Vb. Qidb1)
                /\ wrequest (S, B, s_b_dids, Dids)
                /\ secret (Didb1, didb1, {S, B})
                /\ secret (Didb2, didb2, {S, B})
                /\ witness (B, S, b_s_didb1, Didb1)
                /\ witness (B, S, b_s_didb2, Didb2)
2. State = 10   /\ RCV_AB (A. B. Nb'. Authsb'. Authab')
                /\ Ksb' = Pred (Ss, Ub2)
                /\ Authsb' = Union (Ida, Idb, Ub1, Ub2, Vb, Nb, Ksb, Us, Usb1, Usb2)
                /\ Kab' = Ebilinear (Pred (Ra, Ppub), Pred (Ra, Dida2), Pred (Usa1, Usa2))
```

Figure A3. *Cont.*

```
                /\ Authab' = Union (Ida, Idb, Us, Kab) =|>

 State' := 11   /\ Kba' := Ebilinear (Pred (Usb1, Usb2), Pred (Rb, Ppub), Pred (Rb, Didb2))

                /\ Authba' := Union (Ida, Idb, Us, Kba, Authab)

                /\ Skab' := Union (Ida, Idb, Us, Kab, Authab, Authba)

                /\ SND_AB (B. A. Authba)

                /\ request (S, B, s_b_ksb, Ksb)

                /\ request (A, B, a_b_kab, Kab)

                /\ secret (Kba, kba, {A, B})

                /\ secret (Skab, skab, {A, S, B})

                /\ witness (B, A, b_a_kba, Kba)

end role
```

Figure A3. Role specification of the application server in HLPSL.

References

1. Wen, H.A.; Lin, C.L.; Hwang, T. Provably secure authenticated key exchange protocols for low power computing clients. *Comput. Secur.* **2006**, *25*, 106–113. [CrossRef]
2. Wong, D.S.; Chan, A.H. Efficient and mutually authenticated key exchange for low power computing devices. In Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, 9–13 December 2008; pp. 272–289.
3. Jakobsson, M.; Pointcheval, D. Mutual Authentication for Low-Power Mobile Devices. *Financ. Cryptogr.* **2002**, *2339*, 178–195.
4. Choi, K.; Hwang, J.; Lee, D.; Seo, I. ID-based Authenticated Key Agreement for Low-Power Mobile Devices. *Inf. Sec. Priv.* **2005**, *3574*, 494–505.
5. Chuang, Y.H.; Tseng, Y.M. Towards generalized ID-based user authentication for mobile multi-server. *Int. J. Commun. Syst.* **2012**, *25*, 447–460. [CrossRef]
6. Wu, T.Y.; Tseng, Y.M. An efficient user authentication and key exchange protocol for mobile client–server environment. *Comput. Netw.* **2010**, *53*, 1062–1070. [CrossRef]
7. Tseng, Y.M.; Tseng, L. Ephemeral-Secret-Leakage Secure ID-Based Authenticated Key Exchange Protocol for Mobile Client-Server Environments. In Proceedings of the 24th Cryptology and Information Security Conference, Putrajaya, Malaysia, 24–26 June 2014.
8. Diffie, W.; Hellman, M.E. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
9. Tsai, C.S.; Lee, C.C.; Hwang, M.S. Password Authentication Schemes: Current Status and Key Issues. *IJINS* **2006**, *3*, 101–115.
10. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. *Adv. Cryptol.* **1985**, *5*, 47–53.
11. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In Proceedings of the 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 213–229.
12. Boneh, D.; Boyen, X. Secure Identity Based Encryption without Random Oracles. In Proceedings of the 24th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; pp. 443–459.
13. Waters, B. Efficient Identity-Based Encryption without Random Oracles. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 114–127.
14. Gentry, C. Practical Identity-Based Encryption without Random Oracles. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, 28 May–1 June 2006; pp. 445–464.

15. Joux, A. A One Round Protocol for Tripartite Diffie–Hellman. *Algorithm. Number Theory* **2000**, *1838*, 385–393.
16. Al-Riyami, S.; Paterson, K. Tripartite Authenticated Key Agreement Protocols from Pairings. In Proceedings of the 9th IMA International Conference, Cirencester, UK, 16–18 December 2003; pp. 332–359.
17. Lim, M.H.; Lee, S.; Moon, S. Cryptanalysis of Tso et al.'s ID-Based Tripartite Authenticated Key Agreement Protocol. *Inf. Syst. Secur.* **2007**, *4812*, 64–76.
18. Hölbl, M.; Welzer, T.; Brumen, B. Two proposed identity-based three-party authenticated key agreement protocols from pairings. *Comput. Secur.* **2010**, *29*, 244–252. [CrossRef]
19. Xiong, H.; Chen, Z.; Li, F. New identity-based three-party authenticated key agreement protocol with provable security. *JNCA* **2013**, *36*, 927–932. [CrossRef]
20. Yeh, H.T.; Sun, H.M. Password-based user authentication and key distribution protocols for client–server applications. *J. Syst. Softw.* **2004**, *72*, 97–103. [CrossRef]
21. Kohl, J.T.; Neuman, B.C.; Tso, T.Y. The evolution of the Kerberos authentication system. In *Distributed Open System*; IEEE Computer Society Press: Washington, DC, USA, 1991; pp. 78–94.
22. Yeh, H.T.; Sun, H.M. Password authenticated key exchange protocols among diverse network domains. *Comput. Electr. Eng.* **2005**, *31*, 175–189. [CrossRef]
23. Li, G. Optimal authentication protocols resistant to password guessing attacks. In Proceedings of the Eighth IEEE Computer Security Foundations Workshop, Kerry, Ireland, 13–15 June 1995; pp. 24–29.
24. Kwon, T.; Kang, M.; Jung, S.; Song, J. An Improvement of the Password-Based Authentication Protocol (K1P) on Security against Replay Attacks. *IEICE Trans. Commun.* **1999**, *82*, 991–997.
25. Kwon, T.; Song, J. Authenticated key exchange protocols resistant to password guessing attacks. *Commun. IEE Proc.* **1998**, *145*, 304–308. [CrossRef]
26. Chang, T.Y.; Hwang, M.S.; Yang, W.P. A communication-efficient three-party password authenticated key exchange protocol. *Inf. Sci.* **2011**, *181*, 217–226. [CrossRef]
27. Ni, L.; Chen, G.; Li, J. Escrowable identity-based authenticated key agreement protocol with strong security. *Comput. Math. Appl.* **2013**, *65*, 1339–1349. [CrossRef]
28. Chang, T.Y.; Tsai, C.J.; Lin, J.H. A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. *J. Syst. Softw.* **2012**, *85*, 1157–1165. [CrossRef]
29. Blake-Wilson, S.; Menezes, A. Authenticated Diffe-Hellman Key Agreement Protocols. In Proceedings of the Selected Areas in Cryptography, Kingston, Ontario, Canada, 17–18 August 1999; pp. 339–361.
30. AVISPA v1.1 User Manual. 2006. Available online: http://www.avispa-project.org/ (accessed on 24 January 2018).
31. Chen, T.H.; Lee, W.B.; Chen, H.B. A round- and computation-efficient three-party authenticated key exchange protocol. *J. Syst. Softw.* **2008**, *81*, 1581–1590. [CrossRef]
32. Metz, C. AAA protocols: Authentication, authorization, and accounting for the Internet. *IEEE Int. Comput.* **1999**, *3*, 75–79. [CrossRef]
33. Rensing, C.; Karsten, M.; Stiller, B. AAA: A survey and a policy-based architecture and framework. *IEEE Netw.* **2002**, *16*, 22–27. [CrossRef]
34. Decugis, S. Towards a Global AAA Framework for Internet. In Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet, Bellevue, WA, USA, 20–24 July 2009; pp. 227–230.
35. Dolev, D.; Yao, A.Y. On the Security of Public Key Protocols. *IEEE Inf. Theory Soc.* **1983**, *29*, 198–208. [CrossRef]
36. AVISPA Web tool. Automated Validation of Internet Security Protocols and Applications. Available online: http://www.avispa-project.org/web-interface (accessed on 24 January 2018).
37. Scott, M.; Costigan, N.; Abdulwahab, W. Implementing Cryptographic Pairings on Smartcards. In Proceedings of the 8th International Workshop, Yokohama, Japan, 10–13 October 2006; pp. 134–147.
38. Oliveira, L.B.; Aranha, D.F.; Gouvêa, C.P.L.; Scott, M.; Câmara, D.F.; López, J. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Comput. Commun.* **2011**, *34*, 485–493. [CrossRef]
39. Hu, L.; Dong, J.W.; Pei, D.Y. Implementation of Cryptosystem Based on Tate Pairing. *J. Comput. Sci. Technol.* **2005**, *20*, 264–269. [CrossRef]