Authenticated Key Agreement Scheme with Privacy-Protection in the Three-party Setting

Jeng-Ping Lin[1] and Jih-Ming Fu[2]
(Corresponding author: Jeng-Ping Lin)
The Department Commerce Technology & Management, Chihlee Institute of Technology[1]
No. 313, Sec. 1, Wunhua Rd., Banciao District, New Taipei City, 22050 Taiwan, R.O.C.
The Department of Computer Science & Information Engineering, Cheng Shiu University[2]
No.840, Chengcing Rd., Niaosong District, Kaohsiung City, 83347 Taiwan, R.O.C.
(Email: jplin@mail.chihlee.edu.tw)
(Received December 05, 2010; revised and accepted March 26, 2011)

Abstract

Going along with the rapid development of web technologies, in some applications on demands, partners or staffs may make a great quantity of web transactions or personal communications anytime and anywhere. However, the partners could be distributed over different network domains. They require a communal trusted third party to help them establish a shared session key for future communications. In addition, from the privacy or security point of view, the partners hope that their transactions patterns or movements are not recorded from any external eavesdropper. However, this privacy issue has never been addressed in previous literature. In this paper, we first propose a three-party key agreement scheme to construct a secure transaction mechanism with privacy protection. In our scheme, the major merits include: (1) prevention of some known attacks; (2) satisfaction of the perfect forward secrecy; (3) security against the session state reveal; (4) privacy protection; (5) no sensitive verifier table; and (6) low communication and computation cost.

Keywords: authentication, cryptography, PKI infrastructure, privacy preservation, three-party key agreement

1 Introduction

In some applications, partners or staffs have many opportunities to obtain their preferred services from web sites by using their mobile devices through Internet. However, both of the service requester and the service provider are distributed over different network domains [12], a secure mechanism has to make sure that the communication is securely finished and is secure against any unauthorized user from eavesdropping the delivery contents [2, 10, 19, 21, 25]. To use a traditional two-party key agreement scheme to help two participants who do not believe each other in Internet to establish a session key, the communication and computation cost is heavy [15, 18]. Many previous schemes were proposed to construct secure and efficient three-party key agreement schemes [9, 13, 24]. However, those schemes require a password table to identify the registered users and the burden of the communication and computation cost is still heavy. A service requester could take along the laptop, PDA, or tablet type devices anytime and anywhere. When the service requester is ready to communicate from some web sites, the requester may use those devices to launch some transactions through the resource-restricted environments. The efficiency then becomes a crucial criterion.

For some purposes, people attach great importance to the privacy issue for protecting the reveal of some sensitive information [22]. Those sensitive information could be a service requester's movement, individual's social circle, shopping patterns and individual preferences etc [7, 11]. Besides, from a company competition point of view, it is an important issue to protect the identity. The privacy preserving of service requesters can be briefly defined the follows kinds [26]:

1) External privacy: The identity information of the service requester is hidden from the external attackers;

2) Internal privacy I: The identity information of the service requester is hidden from the external attackers and the service provider;

3) Internal privacy II: The identity information of the service requester is hidden from the external attackers, the service provider and the trusted third party;

In this paper, we address the internal privacy I of the service requester and propose a robust authenticated key agreement scheme to satisfy the following security requirements which are not addressed or cannot be satisfied in previous literature:

S1. Validity: In the absence of an active adversary who can send cheated messages to the communicated parties, two communicated parties accept the same session key.

S2. Man-in-the-middle attack: In this attack, the attacker actively eavesdrop the connection between the victims and relays the messages to them. Since the received messages are true, the victims will not perceive the attack. This attack can be easily implemented in many circumstances such as wireless.

S3. Preventing stolen-verifier attack: In most password-based applications, the server stores a verifier (clear text of passwords or hashed password) to identify remote users. This attack means that an adversary steals the verifier from the server and can masquerade as a legal user to pass the verification of the server [17]. A secure authentication scheme should prevent it.

S4. Preventing unknown key-sharing: On an authenticated key agreement scheme, the attacker makes the following event comes true that an entity A believes to share a session key with another entity B. However, B mistakenly believes that the session key is instead shared with another entity E [6].

S5. Perfect forward secrecy: For a key agreement scheme, the property means that the compromise of long-term key material should not damage the secrecy of any session keys that were previously established from the long-term material, where the key material indicates the user's secret key [20].

S6. Security against session state reveal: The definition is formally addressed in [14]. The security consideration is addressed that the used ephemeral keys in the sessions are more easily leaked to an outsider or are not more well protected than the long-term secret key. Even if the used ephemeral keys are known by an adversary, the adversary cannot learn the corresponding session key.

S7. Privacy protection: No one can identify the identity of the service requester except a partner of a fresh session. More, all of the requests cannot be separated from the intercepted messages. It implies that the linkability of the requests from being sent by a user cannot be traced.

S8. Session key indistinguishability: For all probabilistic, any passive adversary who is ill and observes the exchanged messages cannot derive the session key in a polynomial time.

In the next section, we first define the security of the proposed scheme. In Section 3, we propose an efficient three-party key agreement scheme. In Section 4, we analyze the security of the proposed scheme. In Section 5, we analyze the efficiency among our proposed scheme and the related schemes. Finally, we conclude this paper in Section 6.

2 Notations of Security

We first define some hard mathematical problems in the elliptic curve cryptosystem.

2.1 Hard Problems

Let the form of an elliptic curve be $y^2 = x^3 + ax + b$ over a prime finite field F_q, where a, $b \in F_q$ and $4a^3+27b^2 \neq 0 \bmod q$. Let G_1 be an additive cyclic group of a prime order q, and G_2 be a multiplicative cyclic group with the same order, where $G_1 * G_1 \longrightarrow G_2$. Let Q be a generator of G_1.

1) Definition 1. Elliptic Curve Discrete Logarithm Problem (ECDLP). Given two elements Q and $Q_1 = s * Q$, it is computationally infeasible to derive s, where $s \in Z_q^*$.

2) Definition 2. Elliptic Curve Computational Diffie-Hellman Problem (ECCDHP). Given three points Q, $Q_1 = s * Q$ and $Q_2 = t * Q$, it is computationally infeasible to calculate the point $s * t * Q$, where $s, t \in Z_q^*$.

3) Definition 3. Elliptic Curve Factorization Problem (ECFP). Given two points Q and $Q_1 = s * Q + t * Q$, it is computationally infeasible to discriminate two points $s * Q$ and $t * Q$, where $s, t \in Z_q^*$.

Until now, it is commonly believed that there are no efficient algorithms to solve the above problems in a polynomial-time with a non-negligible probability [12, 16].

2.2 Security Definitions

The concrete security of a three party-based scheme is built up both the property of the session key indistinguishability [21, 22]. In the proposed scheme, the participants include service requester C_A and a service provider $C_B \in \overline{C} = \{C_1, ..., C_{N_C}\}$ and a trusted server S. Each service requesters C_A and service providers C_B hold secret keys, and the server S maintains a long-term private key. We also assume that an adversary $\mathcal{AD}$ who controls all the communications that take place by C_A^i, C_B^j and S is a probabilistic machine, where C_A^i is the ith instance of the service requester C_A and C_B^j is the jth instance of the service provider C_B. $\mathcal{AD}$ can interact with all the participants (C_A, C_B, S) through the following oracle queries.

- Execute(C_A^i, C_B^j), Execute(C_A^i, S), Execute(C_B^j, S): We use this query to model passive attacks where an attacker can eavesdrop all the communications between the instances (C_A^i, C_B^j) and (C_A^i, S), and (C_B^j, S) respectively.

- SendClient(C_A^i, m): We use this query to model an active attack that the attacker sends a message m to a participant C_A at the ith instance. Then query outputs the result that C_A^i would generate upon receiving the message m.

- SendServer(m): We use this query to model an active attack that the attacker sends a message m to the server S. Then query outputs the result that S would generate upon receiving the message m.

- Reveal(C_A^i): We use this query to model an active attack against the known-key attack at the ith instance C_A. The query says that if the instance does not accept the session key, the output is $\perp$; otherwise, the output is the real session key.

- Corrupt(C_A): We use this query to allow that an attacker $\mathcal{AD}$ can corrupt the complete internal state of an entity C_A.

- Test(C_A^i): If an attacker $\mathcal{AD}$ queries this oracle and no session key for $C_A^i \in \overline{C}$ is accepted, this oracle outputs $\perp$; otherwise, the oracle flips a coin b. If $b = 1$, it returns the real session key; if $b = 0$; it returns a random key which has the same key with the real session key.

The security definition of the proposed scheme depends on the partnership and freshness of oracles, where the partnership of the oracles is defined using the session identifier *sid* and the partnership is defined to restrict the adversary's Reveal and Corrupt queries. If the partnership is not accepted by the oracles, the adversary is trying to guess the session key.

- Partnership: We say that two oracles C_A^i and C_B^j are partners, if and only if both of the oracles have accepted the same session key with the same session identifier and they have agreed on the same set of exchanging messages. Besides C_A^i and C_B^j, no other oracles have accepted with the same session identifier.

- Freshness: We say that two oracles C_A^i and C_B^j are fresh if and only if the oracle C_A^i has accepted another partner oracle C_B^j, the oracle C_B^j has accepted another partner oracle C_A^i, and all the oracles C_A^i and C_B^j have not been sent a Reveal query a Corrupt query.

- Session key security: We use the standard semantic security notation to model this property [3]. The security of the session key is defined as that the adversary who wants to discriminate the real key from a random one in the game G is indistinguishable, where the game played between the adversary $\mathcal{AD}$ and a collections of U_x^i oracles. The players $U_x \in \overline{C}$ and S and instances i and $j \in \{1, ..., N_I\}$. $\mathcal{AD}$ runs the game G with the following stages.

  Stage 1: $\mathcal{AD}$ is allowed to send the queries (Execute, SendClient, SendServer, Reveal and Corrupt) in the game.

  Stage 2: During the game G, at some point, $\mathcal{AD}$ can choose a fresh session and send a Test query to one of the fresh oracles C_A^i and C_B^j for the testing. Depending on the unbiased coin b, $\mathcal{AD}$ is given either the actual session key K or a random one from the session key distribution.

  Stage 3: $\mathcal{AD}$ can continue to send the queries to the oracles Execute, SendClient, SendServer, Reveal and Corrupt for its choice. However, $\mathcal{AD}$ is restricted to send the Reveal and Corrupt queries to the oracles for its test session.

  Stage 4: Eventually, $\mathcal{AD}$ winds up the game simulation and decides to output its guess bit b'.

The success of $\mathcal{AD}$ from breaking the scheme in the game is measured in terms of the advantage of $\mathcal{AD}$ from distinguishing whether the received value is the real key or a random one. Let $Adv_P^{G,\mathcal{AD}}(k)$ be the advantage of $\mathcal{AD}$ and the advantage function be defined as follows: $Adv_P^{G,\mathcal{AD}}(k) = |\Pr[b' \text{ - } b] \text{ - } \frac{1}{2}|$, where k is a security parameter.

3 Our Scheme

The intention of our scheme is to propose a secure and efficient three-party key agreement scheme with privacy protection of service requesters by using the elliptic curve cryptosystem. Before we introduce the scheme, we first notify some used parameters as follows.

* $\|$ denotes the concatenation operator of any two strings.
* H_1: $\{0, 1\}^* \longrightarrow G_1$ be a cryptographic hash function which maps a string to a point of the additive cyclic group G_1.
* Let H_2 be a secure one-way hash function which maps a string to a 160-bits string such as SHA-1 [5]. We assume that $H_2()$ is secure against the brute force attack by an adversary.
* We use (P_x, P_y) to denote the x and y coordinates of a point P.

3.1 The Proposed Scheme

The Initialization Phase

1) The used parameters of the elliptic curve cryptosystem are defined in Subsection 2.

2) Users A and B must register to S for generating their private keys and the corresponding public keys. Users A and B send their identities to S for registration.

3) S calculate $Q_{ID_A} = H_1(ID_A)$ and $Q_{ID_B} = H_1(ID_B)$ to check whether the identities have been registered

before. If the registered identities are fresh, S records Q_{ID_A} and Q_{ID_B} into a ID table.

4) Then S helps A and B to generate their private keys and the corresponding public keys. Let d_A and U_A be the private key and the public key of the user A, where $U_A = d_A * Q$. Let d_B and U_B be the private key and the public key of the user B, where $U_B = d_B * Q$. Let d_S and U_S be the private key and the public key of the server S, where $U_S = d_S * Q$.

The Authenticated Key Agreement Phase

When a requester A gets ready to obtain the service from a web site B, they have to exchange some secret information through a communal trusted server S for agreeing a common session key. The rounds are introduced as follows.

Round 1.

1) A generates a session identifier sid and selects a random integer $r_A \in Z_q^*$. A calculates Q_{ID_A}, Q_{ID_B}, (R_{A1}, K_{AS1}, K_{AS2}, R_{A2}, and V_A), where $Q_{ID_A} \longleftarrow \mathrm{H}_1(ID_A) = (Q_{ID_Ax}, Q_{ID_Ay})$, $Q_{ID_B} \longleftarrow \mathrm{H}_1(ID_B) = (Q_{ID_Bx}, Q_{ID_By})$, $R_{A1} \longleftarrow r_A * Q = (R_{A1x}, R_{A1y})$, $K_{AS1} \longleftarrow d_A * U_S = (K_{AS1x}, K_{AS1y})$, $K_{AS2} \longleftarrow r_A * U_S = (K_{AS2x}, K_{AS2y})$, $R_{A2} \longleftarrow Q_{ID_A} + K_{AS2}$ and $V_A \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS1x} \| R_{A1x})$.
2) A sends the request (sid, R_{A1}, R_{A2}, V_A) to B for asking to share a session key.

Round 2.

1) Upon receiving the request, B selects a random integer $r_B \in Z_q^*$ and calculates (Q_{ID_B}, R_{B1}, K_{BS1}, K_{BS2}, and V_B), where $Q_{ID_B} \longleftarrow \mathrm{H}_1(ID_B) = (Q_{ID_Bx}, Q_{ID_By})$, $R_{B1} = r_B * Q = (R_{B1x}, R_{B1y})$, $K_{BS1} = d_B * U_S = (K_{BS1x}, K_{BS1y})$, $K_{BS2} = r_B * U_S = (K_{BS2x}, K_{BS2y})$ and $V_B \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS1x} \| R_{A1x} \| R_{B1x})$.
2) B sends the response (sid, R_{B1}) to A for notifying the request is accepted and (sid, R_{A1}, R_{A2}, V_A, ID_B, R_{B1}, V_B) to S for verifying the validity of A, simultaneously.

Round 3.

1) Upon receiving the message (sid, R_{A1}, R_{A2}, V_A, ID_B, R_{B1}, V_B), S calculates $K_{AS2} \longleftarrow d_S * R_A = (K_{AS2x}, K_{AS2y})$ to derive the point $Q_{ID_A} \longleftarrow R_{A2}$ - $K_{AS2} = (Q_{ID_Ax}, Q_{ID_Ay})$. If the point exists in the ID table, S calculates $K_{AS1} \longleftarrow d_S * U_A = (K_{AS1x}, K_{AS1y})$ and $Q_{ID_B} \longleftarrow \mathrm{H}_1(ID_B) = (Q_{ID_Bx}, Q_{ID_By})$ to check whether V_A is equal to $\mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS1x} \| R_{A1x})$. If it is true, S believes that A is a valid user; otherwise, S terminates the communication and sends an authentication-failed message to B.
2) S calculates $K_{BS1} \longleftarrow d_S * U_B = (K_{BS1x}, K_{BS1y})$ and checks whether V_B is equal to $\mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS1x} \| R_{A1x} \| R_{B1x})$. If it is true, S believes that B is also a privileged user and A an B are the communicated parties; otherwise, S terminates the communication and sends an authentication-failed message to A.
3) S selects a random integer $r_S \in Z_q^*$ and calculates $K_{AS3} \longleftarrow r_S * U_B = (K_{AS3x}, K_{AS3y})$, $R_{SA} \longleftarrow K_{AS3} + K_{AS2}$ and $V_{SA} \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS2x} \| R_{A1x} \| R_{B1x} \| K_{AS3x})$. Similarly, S calculates $K_{BS2} \longleftarrow d_S * R_B = (K_{BS2x}, K_{BS2y})$, $K_{BS3} \longleftarrow r_S * U_A = (K_{BS3x}, K_{BS3x})$, $R_{SB} \longleftarrow K_{BS3} + K_{BS2}$ and $V_{SB} \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS2x} \| R_{A1x} \| R_{B1x} \| K_{BS3x})$.
4) S sends the response (sid, R_{SB}, V_{SB}) to B and (sid, R_{SA}, V_{SA}) to A, simultaneously.
5) Upon receiving the response (sid, R_{SA}, V_{SA}) from S, A calculates $K_{AS3} \longleftarrow R_{SA}$ - $K_{AS2} = (K_{AS3x}, K_{AS3y})$ to check whether V_{SA} is equal to $\mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS2x} \| R_{A1x} \| R_{B1x} \| K_{AS3x})$. If it holds, A calculates the point $K_{AB} \longleftarrow d_A * K_{AS3} + r_A * R_{B1} = (K_{ABx}, K_{ABy})$ and the session key $SK_{AB} \longleftarrow \mathrm{H}_2(sid \| K_{ABx} \| R_{A1x} \| R_{B1x})$ and believes that B is confirmed by S and both of A and B hold the same session key SK_{AB}.
6) Upon receiving the response (sid, R_{SB}, V_{SB}) from S, B calculates $K_{BS3} \longleftarrow R_{SB}$ - $K_{BS2} = (K_{BS3x}, K_{BS3y})$ to check whether V_{SB} is equal to $\mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS2x} \| R_{A1x} \| R_{B1x} \| K_{BS3x})$. If it holds, B calculates the point $K_{AB} \longleftarrow d_B * K_{BS3} + r_B * R_{A1} = (K_{ABx}, K_{ABy})$ and the session key $SK_{AB} \longleftarrow \mathrm{H}_2(sid \| K_{ABx} \| R_{A1x} \| R_{B1x})$ and believes that A is confirmed by S and both of A and B hold the same session key SK_{AB}.

Validity. Following the scheme, both of the service requester A and the service requester B can obtain the same session key: $K_{AB} = d_B * K_{BS3} + r_B * R_{A1} = d_B * r_S * U_A + r_B * r_A * Q = d_A * r_S * U_B + r_B * r_A * Q = d_A * K_{AS3} + r_A * R_{B1}$.

4 Security Analysis

We analyze that some well-known security threats cannot work on our proposed scheme.

Man-in-the-middle attack:

Case 1. $\mathcal{A}$ can observe the network activities of B. Since the trust of B believing the received message depends on the response of the server S, $\mathcal{A}$ has to forge V_A for passing the verification of S. By our Definition 2, it is

hard for $\mathcal{A}$ to derive the secret point $d_A * d_S * Q$ by using the points U_A and U_S.

Case 2. $\mathcal{A}$ can observe the network activities of S. Since the trust of S believing the received message depends on the verification of V_A and V_B, $\mathcal{A}$ has to forge V_A and V_B for passing the verifications. The reason is the same as Case 1. By our Definition 2, this way is infeasible.

Case 3. $\mathcal{A}$ wants to forge the responses of S. Since the trust of A and B believing the responses are sent from S depends on the verification of V_{SA} and V_{SB}. The goal of $\mathcal{A}$ is to forge V_{SA} and V_{SB} for passing the verifications. By our Definition 2, it is computationally infeasible for $\mathcal{A}$ to calculate the points $r_A * d_S * Q$ and $r_B * d_S * Q$. It implies that $\mathcal{A}$ cannot forge the points (V_{SA}, V_{SB}). The man-in-the-middle attack does not work in our scheme.

The perfect forward secrecy. Assume that $\mathcal{A}$ obtains the private keys (d_A, d_B, d_S) with the communicated messages (sid, R_{A1}, R_{A2}, V_A, R_{B1}, R_{SA}, V_{SA}, R_{SB}, V_{SB}). $\mathcal{A}$ can derive the points K_{AS3} and K_{BS3}. However, by our Definition 2, it is still computationally infeasible for $\mathcal{A}$ to calculate the points $r_A * R_{B1}$ and $K_{AB} \longleftarrow d_A * K_{AS3} + r_A * R_{B1} = (K_{ABx}, K_{ABy})$. It implies that $\mathcal{A}$ cannot derive the previous session keys $SK_{AB} \longleftarrow \mathrm{H}_2(sid \parallel K_{ABx} \parallel R_{A1x} \parallel R_{B1x})$ without the knowledge of the ephemeral keys r_A and r_B.

The session state reveal attack. Assume that $\mathcal{A}$ can learn the internal state to obtain the ephemeral keys r_A and r_B with the corresponding exchanged messages. $\mathcal{A}$ can derive the point K_{AS3} by using r_A, but $\mathcal{A}$ cannot construct the secret point ($d_A * K_{AS3}$) by using U_A and K_{AS3} due to our Definition 2. It implies that $\mathcal{A}$ does not gain any advantage to calculate $SK_{AB} \longleftarrow \mathrm{H}_2(sid \parallel K_{ABx} \parallel R_{A1x} \parallel R_{B1x})$ without the knowledge of the private key d_A.

Using the same way, $\mathcal{A}$ also cannot gain any advantage from the points (R_{B1}, R_{SB}). Therefore, our scheme is secure against the session state reveal attack.

Privacy Protection. The goal of $\mathcal{A}$ is to correctly identify the identity of the service requester. **Case 1.** By our Definition 2, when $\mathcal{A}$ wiretaps the communicated message (R_{A1}, R_{A2}), it is computationally infeasible for $\mathcal{A}$ to calculate the point ($K_{AS2} = r_A * U_S$) using the points (R_{A1}, U_S). Also, by our Definition 3, it is also hard for $\mathcal{A}$ to derive the point Q_{ID_A} from R_{A2} without the knowledge of the point K_{AS2}. **Case 2.** $\mathcal{A}$ may guess an identity Q_{ID_A}' and use the guessed value to derive K_{AS2}'. The guessed value cannot be verified on the message V_A. The privacy of user A is preserved in our scheme.

Stolen-verifier attack. In our proposed scheme, each user registers to S and S records a ID table. S does not need to maintain any secrets for identifying the users. Therefore, our scheme is secure against the stolen-verifier attack.

Unknown-key sharing attack. Although the service requester does not know the identity of the service provider. However, the service provider still can confirm the validity of the received messages by the help of the trusted server and believe the communicated party is correct. No outsider $\mathcal{O}$ or any valid malicious user $\mathcal{C}$ can launch the attack successfully.

Case 1. Any outsider $\mathcal{O}$ wants to cheat B from believing the communicated party is $\mathcal{O}$ by using A's messages. At the same time, A still believe the communicated party is B. By our Definition 2, it is computationally infeasible to calculate the point ($d_A * U_S$) and to fit the point into the message authenticated code $V_A \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS1x} \| R_{A1x})$.

Case 2. Any valid malicious user $\mathcal{C}$ wants to cheat B from believing the communicated party is $\mathcal{C}$ by using A's messages. At the same time, A still believes the communicated party is B. The reason is the same as Case 1. It is computationally infeasible to calculate the point ($d_A * U_S$).

Case 3. Any outsider $\mathcal{O}$ or valid malicious user $\mathcal{C}$ wants to imitate the server from sending the messages (R_{SA}, V_{SA}, R_{SB}, V_{SB}) to achieve the purpose of the attack. By our Definition 2, it is still computationally infeasible since it is hard for $\mathcal{O}$ or $\mathcal{C}$ to calculate the points ($r_A * U_S$, $r_B * U_S$) by using the points (R_{A1}, R_{B1}, U_S).

5 Comparisons

In this section, we compare the satisfaction of the security requirements and the performance among our scheme and the previous schemes.

5.1 Security Consideration

As mentioned in Introduction, we compare those admired security criteria with the previous schemes [24, 25] and show the result in Table 1.

5.2 Performance Consideration

5.2.1 Analysis of the Communication Cost

We assume that the output block size of a symmetric cryptosystem is 128bits such as AES [1], the output size of a secure one-way hash function is 160bits such as SHA-1 [5], and the length of the identity, the timestamp and the identifiers is 32-bit and the length of an ephemeral key is 128bits. We also suppose that the elliptic curve cryptosystem is over 163-bits finite field which has the same security level with 1024-bits public key cryptosystems. Therefore, a point in the elliptic curve cyrptosystem is $163 * 2 = 326$bits.

In the meantime, we observe the message steps and communication rounds to evaluate the communication cost and show the compared result in Table 2.

Table 1: Comparisons of satisfaction of the security criteria

	Our scheme	Yang-Chang [23]*1	Chen *et al.* [8]*2
S2	Yes	Yes	Yes
S3	Yes	Yes	Yes
S4	Yes	No	Yes
S5	Yes	Yes	Yes
S6	Yes	No	No
S7	Yes	No	No
Security Proof	Provably Security	Heuristics	Heuristics

"Yes" denotes the satisfaction of the criteria; "No" denotes the unsatisfaction of the criteria; ∗1: the scheme is suffered from the impersonation attack. See Appendix B. ∗2: the timestamp is required in the scheme. It implies that the time-synchronization exists in the scheme.

Table 2: Comparisons of the communication cost

	Communication Cost		
	Round	Step	Sent Size*1
Our scheme	3	5	844bits
Yang and Chang [23]	3	6	1190bits
Chen *et al.* [8]	3	7	1442bits

∗1: We only compare the sent message size of the user A since A is at the resource-restricted environment.

1) Message Step means that one entity has sent data to the communicated party.

2) Communication Round means that if the sent data are independent between message steps, one or more message steps can be integrated into the same communication round due to the data can be performed in parallel. The burden of the communication cost can be reduced.

5.2.2 Analysis of the Computation Cost

We assume that T_{EXP} is the time of one modular exponential operation over a large prime number; T_H is the time of one hash function operation; T_{SYM} is the time of one symmetric en/decrypted operation; T_{MUL} is the time for one modular multiplication; T_{EC_M} is the time for the multiplication operation over an elliptic curve; and $T_{EC_{ADD}}$ is the time for the addition operation over an elliptic curve. We also assume that a point P_1 subtracts another point P_2 is similar to P_1 adds the negative value of P_2, $(P_1 - P_2)$.

As introduced in [20], we learn a relationship as follows: $1T_{EXP} \simeq 240T_{MUL}$, $1T_{EXP} \simeq 600T_H$, $1T_{EC_{ADD}} \simeq 5T_{MUL}$ and $1T_{EC_M} \simeq 29T_{MUL}$. We then analyze the computation cost of the authenticated key agreement phase.

Table 3: Comparisons of the computation cost

	Computation Cost		
	A	B	S
Our scheme	$5T_{EC_M} + 5T_H + 3T_{EC_{ADD}}$	$5T_{EC_M} + 4T_H + 2T_{EC_{ADD}}$	$6T_{EC_M} + 5T_H + 3T_{EC_{ADD}}$
Yang and Chang [23]	$5T_{EC_M} + 2T_{SYM}$	$5T_{EC_M} + 2T_{SYM}$	$2T_{EC_M} + 4T_{SYM}$
Chen *et al.* [8]	$2T_{EC_M} + 4T_H$	$2T_{EC_M} + 4T_H$	$6T_H + 2T_{Sub} + 2T_{MUL}$*2

∗2: The modular size is 160bits.

Client A (1.) In Round 1, A calculates Q_{ID_A}, Q_{ID_B}, $R_{A1} \longleftarrow r_A * Q$, $K_{AS1} \longleftarrow d_A * U_S$, $K_{AS2} \longleftarrow r_A * U_S$, $R_{A2} \longleftarrow Q_{ID_A} + K_{AS2}$, and $V_A \longleftarrow \mathrm{H}_2(sid \parallel Q_{ID_{Ax}} \parallel Q_{ID_{Bx}} \parallel K_{AS1x} \parallel R_{A1x})$. The computation cost is three T_H plus three T_{EC_M} plus one $T_{EC_{ADD}}$. (2.) In Round 3, A recovers the point $K_{AS3} \longleftarrow R_{SA} - K_{AS2}$, $\mathrm{H}_2(sid \parallel Q_{ID_{Ax}} \parallel Q_{ID_{Bx}} \parallel K_{AS2x} \parallel R_{A1x} \parallel R_{B1x} \parallel K_{AS3x})$, $K_{AB} \longleftarrow d_A * K_{AS3} + r_A * R_{B1}$, and $SK_{AB} \longleftarrow \mathrm{H}_2(sid \| K_{ABx} \| R_{A1x} \| R_{B1x})$. The computation cost is two T_H plus two T_{EC_M} plus two $T_{EC_{ADD}}$.

Client B (1.) In Round 2, B calculates Q_{ID_B}, $R_{B1} \longleftarrow r_A * Q$, $K_{BS1} \longleftarrow d_B * U_S$, $K_{BS2} \longleftarrow r_B * U_S$, and $V_B \longleftarrow \mathrm{H}_2(sid \parallel Q_{ID_{Bx}} \parallel K_{BS1x} \parallel R_{A1x} \parallel R_{B1x})$. The computation cost is two T_H plus three T_{EC_M}. (2.) In Round 3, B recovers the point $K_{BS3} \longleftarrow R_{SB} - K_{BS2}$, $\mathrm{H}_2(sid \parallel Q_{ID_{Bx}} \parallel K_{BS2x} \parallel R_{A1x} \parallel R_{B1x} \parallel K_{BS3x})$, $K_{AB} \longleftarrow d_B * K_{BS3} + r_B * R_{A1}$, and $SK_{AB} \longleftarrow \mathrm{H}_2(sid \| K_{ABx} \| R_{A1x} \| R_{B1x})$. The computation cost is two T_H plus two T_{EC_M} plus two $T_{EC_{ADD}}$.

Server S (1.) In Round 3, S calculates $K_{AS2} \longleftarrow d_S * R_A$, $Q_{ID_A} \longleftarrow R_{A2} - K_{AS2}$, $K_{AS1} \longleftarrow d_S * U_A$, Q_{ID_B}, $\mathrm{H}_2(sid \parallel Q_{ID_{Ax}} \parallel Q_{ID_{Bx}} \parallel K_{AS1x} \parallel R_{A1x})$, $K_{BS1} \longleftarrow d_S * U_B$, and $\mathrm{H}_2(sid \parallel Q_{ID_{Bx}} \parallel K_{BS1x} \parallel R_{A1x} \parallel R_{B1x})$. The computation cost is three T_H plus three T_{EC_M} plus one $T_{EC_{ADD}}$. (2.) If all of the conditions passed, S calculates $K_{AS3} \longleftarrow r_S * U_B$, $R_{SA} \longleftarrow K_{AS30} + K_{AS2}$, $V_{SA} \longleftarrow \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \parallel K_{AS2x} \parallel R_{A1x} \parallel R_{B1x} \parallel K_{AS3x})$, $K_{BS2} \longleftarrow d_S * R_B$, $K_{BS3} \longleftarrow r_S * U_A$, $R_{SB} \longleftarrow K_{BS3} + K_{BS2}$ and $V_{SB} \longleftarrow \mathrm{H}_2(sid \parallel Q_{ID_Bx} \parallel K_{BS2x} \parallel R_{A1x} \parallel R_{B1x} \parallel K_{BS3x})$. The computation cost is two T_H plus three T_{EC_M} plus two $T_{EC_{ADD}}$.

We show the results in Table 3 and we can see that the communication cost and the sent message size of user A in our scheme is lower than other schemes. Besides, the computation cost of user A is efficient and is similar to other schemes.

6 Conclusion

We have shown that Yang-Chang's efficient three-party mechanism [23] is not secure against the impersonation and the unknown key sharing attacks in Appendix B. By Tables 1, 2 and 3, our scheme not only satisfies more admired security criteria with privacy protection (Internal privacy I), but also keeps the efficiency.

References

[1] "Advanced Encryption Standard," *http://csrc.nist.gov/encryption/aes/*.

[2] R. Abdellatif, H. K. Aslan, and S. H. Elramly, "New real time multicast authentication protocol," *International Journal of Network Security*, vol. 12, no. 1, pp. 13–20, 2011.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated and key exchange secure against dictionary attacks," in *Advances in Cryptology - EUROCRYPT 2000, LNCS 1807*, pp. 139–155, 2005.

[4] M. Bellare and P. Rogaway, "Provably secure session key distribution the three party case," in *Proceedings of the 27th ACM Annual Symposium on the Theory of Computing*, pp. 57–66, 1995.

[5] E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby, and C. Lemuet, "Collisions in SHA-0 and reduced SHA-1," in *Advances in Cryptology, EUROCRYPT'05*, pp. 36–57, 2008.

[6] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol," in *Public Key Cryptography (PKC'99) Proceedings, LNCS 1560*, pp. 154–170, 1999.

[7] C. H. Chen, G. Horng, and C. H. Hsu, "A novel private information retrieval scheme with fair privacy in the user side and the server side," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 3, pp. 801–810, 2009.

[8] T. H. Chen, W. B. Lee, and H. B. Chen, "A roundand computation-efficient three-party authenticated key exchange protocol," *Journal of Systems & Software*, vol. 81, no. 9, pp. 1581–1590, 2008.

[9] H. Y. Chien and T. C. Wu, "Provably secure password-based three-party key exchange with optimal message steps," *The Computer Journal*, vol. 52, no. 6, pp. 646–655, 2009.

[10] E. El-Emam, M. Koutb, H. Kelash, and O. S. Faragallah, "An authentication protocol based on Kerberos 5," *International Journal of Network Security*, vol. 12, no. 3, pp. 159–170, 2011.

[11] W. S. Juang, S. T. Chen, and H. T. Liaw, "Robust and efficient password authenticated key agreement using smart cards," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 6, pp. 251–2556, 2008.

[12] W. S. Juang, C. L. Lei, H. T. Liaw, and W. K. Nien, "Robust and efficient three-party user authentication and key agreement using bilinear pairings," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 2, pp. 763–772, 2010.

[13] H. S. Kim and J. Y. Choi, "Enhanced password-based simple three-party key exchange protocol," *Computers & Electrical Engineering*, vol. 35, no. 1, pp. 107–114, 2009.

[14] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in *Proc. CRYPTO'05, LNCS 3621*, pp. 546–566, 2005.

[15] M. Kumar, "A new secure remote user authentication scheme with smart cards," *International Journal of Network Security*, vol. 11, no. 2, pp. 88–93, 2010.

[16] F. Li, X. Xin, and Y. Hu, "Identity-based broadcast signcryption," *Computer Standards & Interfaces*, vol. 30, no. 1-2, pp. 89–94, 2008.

[17] C. L. Lin and T. Hwang, "A password authentication scheme with secure password updating," *Computers & Security*, vol. 22, no. 1, pp. 68–72, 2003.

[18] J. Liu and J. Li, "A better improvement on the integrated Diffie-Hellman-DSA key agreement protocol," *International Journal of Network Security*, vol. 11, no. 2, pp. 114–117, 2010.

[19] C. Ma and J. Ao, "Certificateless group oriented signature secure against key replacement attack," *International Journal of Network Security*, vol. 12, no. 1, pp. 1–6, 2011.

[20] B. Schneier, *Applied cryptography.* 2nd ed. John Wiley & Sons Inc., 1996.

[21] S. K. Sood, "An improved and secure smart card based dynamic identity authentication protocol," *International Journal of Network Security*, vol. 13, no. 3, pp. 208–215, 2011.

[22] S. F. Tzeng, C. C. Lee, and T. C. Lin, "A novel key management scheme for dynamic access control in a hierarchy," *International Journal of Network Security*, vol. 12, no. 3, pp. 178–180, 2011.

[23] J. H. Yang and C. C. Chang, "An efficient three-party authenticated key exchange protocol using elliptic curve cryptography for mobile-commerce environments," *Journal of Systems & Software*, vol. 82, no. 9, pp. 1497–1502, 2009.

[24] Z. Yong, J. Ma, and S. Moon, "An improvement on a three-party password-based key exchange protocol using weil pairing," *Computers & Electrical Engineering*, vol. 11, no. 1, pp. 17–22, 2010.

[25] M. Zhang, B. Yang, Y. Zhong, P. Li, and T. Takagi, "Cryptanalysis and fixed of short signature scheme without random oracle from bilinear parings," *International Journal of Network Security*, vol. 12, no. 3, pp. 130–136, 2011.

[26] H. Zhu, X. Lin, M. Shi, P. H. Ho, and X. Shen, "PPAB: A privacy-preserving authentication and billing architecture for metropolitan area sharing networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 5, pp. 2529–2543, 2009.

A Security Proof

We prove that our scheme provides the session key indistinguishability property in the random oracle model under the ECCDHP assumption.

Proof. We use a contradiction way to prove it. We assume that an adversary $\mathcal{AD}$ can gain a non-negligible advantage to distinguish the test key in the game and $\mathcal{AD}$ can construct a breaker $\mathcal{AD}''$ to solve the ECCDHP problem, where the advantage of $\mathcal{AD}$ from differentiating the real session key from a random key as follows: $Adv_P^{G,\mathcal{AD}}(k) = \mid Pr[b' - b]$ - $\frac{1}{2} \mid$.

We suppose that an oracle C_A has accepted the session key of the form $\mathrm{H}_2(sid \parallel K_{ABx} \parallel R_{A1x} \parallel R_{B1x})$ with another fresh and partnership oracle C_B. We say that $\mathcal{AD}$ is successful if $\mathcal{AD}$ picks an oracle C_A or C_B to ask a Test query and can output the bit guess correctly. Thus, we have $\Pr[\mathcal{AD}$ succeeds$] = \frac{1}{2} + \eta(k)$, where $\eta(k)$ is non-negligible.

Let Q_H be the event that $\mathrm{H}_2()$ has been queried on $(sid \parallel K_{ABx} \parallel R_{A1x} \parallel R_{B1x})$ by $\mathcal{AD}$ or some oracles. Then $\Pr[\mathcal{AD}$ succeeds$] = \Pr[\mathcal{AD}$ succeeds $\mid Q_H] * \Pr[Q_H]$ + $\Pr[\mathcal{AD}$ succeeds $\mid \overline{Q}_H] * \Pr[\overline{Q}_H]$. Since $\mathrm{H}_1()$ and $\mathrm{H}_2()$ are random oracles and C_A and C_B are fresh oracles, it implies $\Pr[\mathcal{AD}$ succeeds $\mid \overline{Q}_H] = \frac{1}{2}$. Hence, $\frac{1}{2} + \eta(k) \leq \frac{1}{2}$ + $\Pr[Q_H]$. We then have $\Pr[Q_H] \geq \eta(k)$.

The adversary $\mathcal{AD}$ selects a fresh oracle C_A which has accepted a session key. Then the probability of $\mathrm{H}_2()$ being queried on $(sid \parallel K_{ABx} \parallel R_{A1x} \parallel R_{B1x})$ by $\mathcal{AD}$ or some oracles other than C_A and C_B is non-negligible. As mentioned before, we have assumed that $\mathcal{AD}$ constructs a breaker $\mathcal{AD}''$ which can solve the ECCDHP with non-negligible probability. The task of $\mathcal{AD}''$ is that: Given $Q_1 = s * Q$ and $Q_2 = t * Q$, $\mathcal{AD}''$ outputs $s * t * Q$, where s and t are chosen randomly.

$\mathcal{AD}''$ executes the following process:

(1.) Randomly select C_A and C_B from $\overline{C} = \{C_1, C_2, ..., C_{N_C}\}$ and instances u and v from $\{1, 2, ..., N_I\}$, where N_C, N_I and N_{H_2} denote the number of service requesters and service providers, the instances per entity and the number of distinct queries to H_2. Note that all these parameters are polynomial on the security parameter.

(2.) Determine two oracles C_A^u and C_B^v who are partnership.

(3.) Guess that $\mathcal{AD}$ will choose one of C_A^u and C_B^v who have accepted the session to ask its Test query after $\mathcal{AD}$ decides to terminate the game.

Given the challenge $(X^* = r_A * Q, Y^* = r_B * Q)$ to $\mathcal{AD}''$, $\mathcal{AD}''$ sets the public parameters as $(\mathrm{H}_1, \mathrm{H}_2, q, Q)$ and a ID table. $\mathcal{AD}''$ also maintains the lists L_{H_1} and L_{H_2} for the random oracles H_1 and H_2 queries, L_{Send} for the communicated transcripts, and L_{Key} for the corresponding keys of each session. $\mathcal{AD}''$ selects the secret keys d_X and the corresponding public keys U_X for each C_A and $C_B \in \{C_1, C_2, ..., C_{N_C}\}$ at random.

During the game, $\mathcal{AD}$ will ask some queries to $\mathcal{AD}''$. The answers are given as follows:

(1.) Hash query: $\mathcal{AD}''$ randomly responses H_1 and H_2 queries which are like real random oracles do, and records all the inputs and the corresponding outputs in L_{H_1} and L_{H_2}, respectively.

(2.) Corrupt(C) query: If C is one of C_A and C_B, $\mathcal{AD}''$ gives up; otherwise, $\mathcal{AD}''$ answers all the internal state of C to $\mathcal{AD}$.

(3.) SendClient(C_X^i, m) query:

1) If $(C_X = C_A)$ && $(i = u)$ && (m = "start" for some $C_Y \in \overline{C}$ && $C_X \neq C_Y$), then $\mathcal{AD}''$ generates $\{sid, X^* = r_A * Q, K_{AS1} = d_A * U_S, K_{AS2} = r_A * U_S, R_{A2} = Q_{ID_A} + K_{AS2}, V_A\}$, where $V_A = \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS1x} \| X_x^*)$ and ? denotes the corresponding exponent of X^* and is unknown. Finally, $\mathcal{AD}''$ responds (sid, X^*, R_{A2}, V_A) as the oracle output and records the responsive transcript and the random exponent $(?, X^*)$ into the L_{Send} list, V_X into the L_{H_2} list and (K_{AS1}, K_{AS2}) into the L_{Key} list.

2) If $(C_X = C_B)$ && $(j = v)$ && (m has the form of (sid, Y^*, R_{Y2}, V_Y) for $C_Y \in \overline{C}$ && $C_Y \neq C_X$, then $\mathcal{AD}''$ responds the scheme says $\{sid, Y^*, R_{Y2}, V_Y, ID_X, X^*, V_X\}$, where $X^* = r_B * Q$, $K_{BS1} = d_B * U_S$, $K_{BS2} = r_B * U_S$ and $V_B = \mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS1x} \| Y_x^* \| X_x^*)$. Finally, $\mathcal{AD}''$ records the responsive transcript and the random exponents $(?, X^*)$ into the L_{Send} list, V_B into the L_{H_2} list and (K_{BS1}, K_{BS2}) into the L_{Key} list, where ? denotes the corresponding exponent of X^* and is unknown.

3) If $(C_X \in \overline{C})$ && (m has the form of ("start" from $C_Y \in \overline{C}$ && $C_Y \neq C_X$)), then $\mathcal{AD}''$ generates $\{sid, X^* = r_A * Q, K_{AS1} = d_A * U_S, K_{AS2} = r_A * U_S, R_{A2} = Q_{ID_A} + K_{AS2}, V_A\}$, where r_A is chosen by $\mathcal{AD}''$ at random and $V_A = \mathrm{H}_2(sid \| Q_{ID_Ax} \| Q_{ID_Bx} \| K_{AS1x} \| X_x^*)$. Finally, $\mathcal{AD}''$ responds (sid, X^*, R_{A2}, V_A) as the oracle output and records the responsive transcript and the random exponent (r_A, X^*) into the L_{Send} list, V_X into the L_{H_2} list and (K_{AS1}, K_{AS2}) into the L_{Key} list.

4) If $(C_X \in \overline{C})$ && (m has the form of $(C_Y, sid, Y^*, R_{Y2}, V_Y)$ for some $C_Y \in \overline{C}$), then $\mathcal{AD}''$ responds $\{sid, Y^*, R_{Y2}, V_Y, ID_X, X^*, V_X\}$, where $X^* = r_B * Q$, $K_{BS1} = d_B * U_S$, $K_{BS2} = r_B * U_S$, $V_B = \mathrm{H}_2(sid \| Q_{ID_Bx} \| K_{BS1x} \| Y_x^* \| X_x^*)$ and r_B is chosen by $\mathcal{AD}''$ at random. Finally, $\mathcal{AD}''$ records the responsive transcript and the random exponents (r_B, X^*) into the L_{Send} list, V_B into the L_{H_2} list and (K_{BS1}, K_{BS2}) into the L_{Key} list.

5) If $(C_X \in \overline{C})$ && (m has the form of (sid, R_{SX}, V_{SX}) for for some $C_Y \in \overline{C}$), then $\mathcal{AD}''$ consults its L_{Send}, L_{H_2} and L_{Key} lists by using sid to find a matched entry. If the matched entry can be found, $\mathcal{AD}''$ extracts the local values from L_{H_1}, L_{H_2} and L_{Send} lists and uses them to retrieve $K_{XS3} = R_{SX} - K_{XS2}$. $\mathcal{AD}''$ verifies the validity of V_{SX}. If the verification succeeds, $\mathcal{AD}''$ calculates $K_{XY} = d_X * K_{XS3} + r_X * R_{Y1}$

and $SK_{XY} = \mathrm{H}_2(sid \| K_{XYx} \| R_{X1x} \| R_{Y1x})$. $\mathcal{AD}''$ stores them into L_{Send} and L_{Key} and L_{H_2} lists. If the verification does not hold, $\mathcal{AD}''$ gives up; $\mathcal{AD}''$ records corresponding data into its L_{Send} list.

6) $\mathcal{AD}''$ responses with error messages for all the other cases.

(4.) SendServer(m) query:

1) If (C_X and $C_Y \in \overline{C}$) && (m has the form of ("start", sid, X^*, R_{X2}, V_X, ID_Y, Y^*, V_Y), then $\mathcal{AD}''$ uses the private key d_S to recover the received data. If V_X and V_Y can pass the verification, $\mathcal{AD}''$ selects an integer r'_S at random, and responds with the transcript $\{sid, R_{SX}, V_X\}$ and $\{sid, R_{SY}, V_Y\}$. Finally, $\mathcal{AD}''$ records all the transcripts and the randomly secret exponent r'_S in its L_{Send} list and L_{H_2} list respectively.

2) $\mathcal{AD}''$ responses with error messages for all the other cases.

Reveal(C_X^i) query: After receiving the query, $\mathcal{AD}''$ consults the records in the list of L_{Key} and reveals all the internal state and the session keys.

$\mathcal{AD}$ then answers its guess and requires $\mathcal{AD}''$ searching its L_{H_1} and L_{H_2} lists for the entry, where the entry has the input of the form (sid, K_{ABx}, R_{X1x}, R_{Y1x}) for some SK_{XY}. Finally, $\mathcal{AD}''$ outputs SK_{AB} as the Diffie-Hellman key of C_X and C_Y. There are the two possible results for the above experiment:

1) $\mathcal{AD}''$ gives up if $\mathcal{AD}$ does not make its queries where C_A^u or C_B^v has accepted their session.

2) If $\mathcal{AD}$ does make its queries, then C_A^u or C_B^v will accept their session and hold the key formed $\mathrm{H}_2(sid, K_{ABx}, R_{X1x}, R_{Y1x})$. It is the fact that the key $K_{AB} = (d_A * r_S * d_B * Q + r_A * r_B * Q)$ is unknown to $\mathcal{AD}''$, $\mathcal{AD}''$ cannot calculate this key actually.

$\mathcal{AD}''$ will search its L_{H_1} and L_{H_2} lists for the entry and certainly wins its experiment if Case 2 does happen really. Hence, the probability of $\mathcal{AD}''$ outputting the correct value on $(d_A * r_S * d_B * Q + r_A * r_B * Q)$ is: $\frac{Pr[Q_H]}{N_C^2 * N_I^2 * N_{H_2}} \geq \frac{\eta(k)}{N_C^2 * N_I^2 * N_{H_2}}$, where the probability is non-negligible and the result contradicts our ECCDHP assumption. Hence, we can conclude that $\eta(k)$ must be negligible and is the advantage of $Adv_P^{G,\mathcal{AD}}(k)$. The theorem is proven.

B Yang and Chang's scheme [23]

Recently, Yang and Chang proposed a novel three-party key agreement scheme and claimed that their scheme provides low computation cost and light communication load for constrained-resource environment. We then briefly review their scheme and show that the unknown key sharing and the impersonation attacks can work on their scheme.

B.1 The Scheme

The Initialization Phase

The trusted server S initializes the used parameters as follows: (1.) Define an elliptic curve equation $E_q(a, b)$: $y^2 \equiv x^3 + ax + b \bmod q$ with the order n over a finite field F_q, where q is a large odd prime and is larger than 2^{160} and $a, b \in F_q$; (2.) Select $E_K()$ and $D_K()$ as the encryption/decryption algorithms in a secure symmetric cryptosystem such as AES [26], where K is the symmetric key; (3.) Select a public point Q from the equation $E_q(a, b)$, where the order of Q is n; (4.) Publish the parameters ($E_q(a, b)$, $E_K()$, $D_K()$, Q); (5.) The registration phase is the same as our scheme. See step 4 of the initialization phase in our scheme.

The Authenticated Key Agreement Phase

When users A and B want to negotiate a shared session key, they need the help of the server S by performing the following steps.

Round 1.

(1.) A selects a random integer $r_A \in Z_q^*$ and calculates $R_A = r_A * U_A = r_A * d_A * Q$ and $\hat{R}_A = r_A * U_S = r_A * d_S * Q$.

(2.) A calculates $K_A = d_A * \hat{R}_A = d_A * r_A * d_S * Q = (K_{Ax}, K_{Ay})$.

(3.) A selects a random integer $w_A \in Z_q^*$ and calculates $W_A = w_A * Q$ and $C_A = E_{K_{Ax}}(R_A, W_A)$.

(4.) A sends the request (ID_A, *Request*) to B for negotiating a session key to protect the future communications and (ID_A, ID_B, C_A, R_A) to S, simultaneously.

Round 2.

(1.) Upon receiving the request, B selects a random integer $r_B \in Z_q^*$ and calculates $R_B = r_B * U_B = r_B * d_B * Q$ and $\hat{R}_B = r_B * U_S = r_B * d_S * Q$.

(2.) B calculates $K_B = d_B * \hat{R}_B = d_B * r_B * d_S * Q = (K_{Bx}, K_{By})$.

(3.) B selects a random integer $w_B \in Z_q^*$ and calculates $W_B = w_B * Q$ and $C_B = E_{K_{Bx}}(R_B, W_B)$.

(4.) B sends the response (ID_B, *Response*) to A for notifying that the request is accepted and (ID_B, ID_A, C_B, R_B) to S, simultaneously.

Round 3.

(1.) Upon receiving (ID_A, ID_B, C_A, R_A) and (ID_B, ID_A, C_B, R_B), S calculates $K_A = d_S * R_A = (K_{Ax}, K_{Ay})$ and $K_B = d_S * R_B = (K_{Bx}, K_{By})$ by using the private key d_S.

(2.) S then calculates $D_{K_{Ax}}(C_A)$ and $D_{K_{Bx}}(C_B)$.

(3.) S checks whether the decrypted R_A is the same as the received R_A. If it is true, S believes that A is a valid user; otherwise, S terminates the communication and sends an authentication-failed message to B. Using the same way, S checks whether the decrypted R_B is the same as the received R_B. If it is true, S believes that B is a valid user; otherwise, S terminates the communication and sends an authentication-failed message to A.

(4.) S calculates $C_{SA} = E_{K_{Ax}}(R_A, W_B)$ and $C_{SB} = E_{K_{Bx}}(R_B, W_A)$. S sends C_{SA} and C_{SB} to A and B, simultaneously.

(5.) Upon receiving C_{SA}, A calculates $D_{K_{Ax}}(C_{SA})$ to obtain R_A and W_B. A checks whether the decrypted R_A is the same as before. If it is true, A believes that B is confirmed by S and B obtains the same session key $SK = w_A * W_B$; otherwise, the transaction is rejected by A.

(6.) Upon receiving C_{SB}, B calculates $D_{K_{Bx}}(C_{SB})$ to obtain R_B and W_A. B checks whether the decrypted R_B is the same as before. If it is true, B believes that A is confirmed by S and A obtains the same session key $SK = w_B * W_A$; otherwise, the transaction is rejected by B.

B.2 Weakness

We show that the Yang-Chang's scheme is not secure against the unknown key sharing and the impersonation attacks as follows.

Unknown Key Sharing Attack: In the following, we demonstrate that key integrity checking cannot make sure the session key is used between the communicated parties. The described attack is commonly referred to as the unknown key share attack in the literature [4, 6].

Round 1.

(1.) We assume that an adversary $\mathcal{C}$ can observe the network activities of one of the communicated parties. Note that $\mathcal{C}$ is also a valid user.

(2.)In Round 1, A sends the request (ID_A, *Request*) to B for asking to share a session key and (ID_A, ID_B, C_A, R_A) to S, simultaneously.

(3.) Now, $\mathcal{C}$ intercepts them and modifies them as (ID_C, *Request*) and (ID_C, ID_B, C_A, R_A), where we use ID_C to denote the identity of $\mathcal{C}$.

(4.) $\mathcal{C}$ forwards the modified messages to B and S, simultaneously.

Round 2.

(1.) Upon receiving the request, B sends the response (ID_B, *Response*) to A for notifying that the request is accepted and (ID_B, ID_C, C_B, R_B) to S, simultaneously.

(2.) Now, A believes that B receives the request and B believes that the requester is $\mathcal{C}$.

Round 3.

(1.) Upon receiving (ID_C, ID_B, C_A, R_A) and (ID_B, ID_C, C_B, R_B), S calculates K_A and K_B and decrypts C_A and C_B, respectively.

(2.) Without loss of generality, S will believe the received messages are sent from $\mathcal{C}$ and B since both of the decrypted R_A and R_B are the same as the received R_A and R_B.

(3.) S calculates C_{SA} and C_{SB} and sends C_{SA} and C_{SB} to $\mathcal{C}$ and B, simultaneously.

(4.) $\mathcal{C}$ then forwards the response C_{SA} to A. Upon receiving C_{SA}, A calculates $D_{K_{Ax}}(C_{SA})$ to obtain R_A and W_B. A will believe that B is confirmed by S and B obtains the same session key $SK = w_A * W_B$; otherwise, the transaction is rejected by A.

(5.) Upon receiving C_{SB}, B calculates $D_{K_{Bx}}(C_{SB})$ to obtain R_B and W_A. B checks whether the decrypted R_B is the same as before. If it is true, B believes that $\mathcal{C}$ is confirmed by S and $\mathcal{C}$ obtains the same session key $SK = w_B * W_A$; otherwise, the transaction is rejected by B.

(6.) Without loss of generality, A and B believe that the communicated parties are confirmed by S and the calculated session key is true. It is the fact that A believes the communicated party is B and B believes the communicated party is $\mathcal{C}$. The attack is successful.

The Impersonation Attack: The goal of a secure authenticated key agreement scheme is to prevent any unauthorized user to obtain the communicated contents and to prevent that an adversary cheats a privileged user to establish a shared session key. We demonstrate our attack as follows.

Round 1.

(1.) An adversary $\mathcal{C}$ does not register to S and wants to impersonate a valid user A. $\mathcal{C}$ selects a random integer $r_C \in Z_q^*$ and calculates $R_C = r_C * Q$.

(2.) $\mathcal{C}$ calculates $K_C = r_C * U_S = r_C * d_S * Q = (K_{Cx}, K_{Cy})$, where K_{Cx} and K_{Cy} are the x and y coordinates of the point K_C over $E_q(a, b)$.

(3.) $\mathcal{C}$ selects a random integer $w_C \in Z_q^*$ and calculates $W_C = w_C * Q$ and $C_C = E_{K_{Cx}}(R_C, W_C)$. $\mathcal{C}$ sends the request (ID_A, *Request*) to B for asking to share a session key and (ID_A, ID_B, C_C, R_C) to S, simultaneously.

Round 2.

(1.) Upon receiving the request, B calculates R_B, $\hat{R}_B$, K_B, W_B and $C_B = E_{K_{Bx}}(R_B, W_B)$.

(2.) B sends the response (ID_B, *Response*) to $\mathcal{C}$ for notifying the request is accepted and (ID_B, ID_A, C_B, R_B) to S, simultaneously.

Round 3.

(1.) Upon receiving (ID_A, ID_B, C_C, R_C) and (ID_B, ID_A, C_B, R_B), S calculates $K_C = d_S * R_C = d_S * r_C * Q = (K_{Cx}, K_{Cy})$ and $K_B = d_S * R_B = (K_{Bx}, K_{By})$ by using the private key d_S.

(2.) S then calculates $D_{K_{Cx}}(C_C)$ and $D_{K_{Bx}}(C_B)$, respectively.

(3.) S checks whether the decrypted R_C is the same as the received R_C. Without loss of generality, S will believes that the communicated party is a valid user A; Using the same way S checks whether the decrypted R_B is the same as the received R_B. If it is true, S believes that B is also a valid user.

(4.) S calculates $C_{SC} = E_{K_{Cx}}(R_C, W_B)$ and $C_{SB} = E_{K_{Bx}}(R_B, W_C)$. S sends C_{SC} and C_{SB} to $\mathcal{C}$ and B, simultaneously.

(5.) Upon receiving C_{SC}, $\mathcal{C}$ calculates $D_{K_{Cx}}(C_{SC})$ to obtain R_C and W_B. $\mathcal{C}$ checks whether the decrypted R_C is the same as before. If it is true, $\mathcal{C}$ believes that B is confirmed by S and B obtains the same session key $SK = w_C * W_B = w_C * w_B * Q$; otherwise, the transaction is rejected by $\mathcal{C}$.

(6.) Upon receiving C_{SB}, B calculates $D_{K_{Bx}}(C_{SB})$ to obtain R_B and W_C. B checks whether the decrypted R_B is the same as before. If it is true, B believes that $\mathcal{C}$ is confirmed by S, the communicated party is A and $\mathcal{C}$ obtains the same session key $SK = w_B * w_C * Q$; otherwise,

the transaction is rejected by B.

(7.) Without loss of generality, both of B and S believe the communicated party is A and B establish a session key with $\mathcal{C}$ through the help of S. The attack is successful.

Jeng-Ping Lin received the Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1999. He is currently an Assistant Professor with the Department of Commerce Technology and Management, Chihlee Institute of Technology, New Taipei City, Taiwan. His current research interests include cryptography and fault-tolerant computing.

Jih-Ming Fu received the Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 2001. He is currently an Assistant Professor with the Department of Computer Science and Information Engineering, Cheng Shiu University, Kaohsiung, Taiwan. His current research interests include cryptography, embedded operating system and hardware-software codesign.