Cryptanalysis on Improved One-round Lin-Li's Tripartite Key Agreement Protocol

Meng-Hui Lim[1], Sanggon Lee[2], Hoonjae Lee[2]

[1]Department of Ubiquitous IT, Graduate school of Design & IT,
Dongseo University, Busan 617-716, Korea
meng17121983@yahoo.com
[2]Department of Information & Communication,
Dongseo University, Busan 617-716, Korea
{nok60,hjlee}@dongseo.ac.kr

Abstract

A tripartite authenticated key agreement protocol is designed for three entities to communicate securely over an open network particularly with a shared key. Recently, we have improved a one-round tripartite authenticated key agreement protocol proposed by Lin-Li due to its vulnerability to the forging attack in our previous report. However, we have later discovered that both the original Lin-Li's scheme and our previous enhanced protocol are vulnerable to the insider replay attack. Moreover, we have also realized that both protocols have falsely claimed the forward secrecy attribute. In this paper, we will revise our improvements and again secure this protocol against these cryptanalytic attacks while recovering the precious perfect forward secrecy property.

1. Introduction

A *key agreement protocol* is defined as a mechanism in which a shared secret key, often known as *session key*, is derived by two or more protocol entities as a function of information contributed by each of these parties such that no single entity can predetermine the resulting value. This secret key, usually established over a public network, can then be used to create a confidential or integrity-protected communication channel among the entities. In general, a key agreement protocol is called *authenticated* if the protocol is able to ensure that the session key is known only to the intended entities in a protocol run. Without authentication, a key agreement protocol would turn out to be insecure as an adversary can easily intrude the scheme by using the man-in-the-middle attack as well as other cryptographic attacks.

The situation where three or more parties share a secret key is known as *conference keying*. The three-party (or tripartite) case is of most practical importance because it is the most common size for electronic conferences. Not only that, a tripartite key agreement protocol can be used to provide a range of services particularly in the communication of two parties. For instance, a third party can be added to chair or referee a conversation for ad hoc auditing, data recovery or escrow purposes [1].

Over the years, a variety of key agreement protocols have been proposed. However, most of them have been proven to be insecure [4, 5, 7, 10, 11, 14, 16] due to their failure in fulfilling all the desirable security attributes of a key agreement protocol defined by Wilson and Menezes [17, 18]. We define these security attributes as follows:

Known session key security. A protocol is considered to be *known session key secure* if it remains achieving its goal in the face of an adversary who has learned some previous session keys.

(Perfect) forward secrecy. A protocol enjoys *forward secrecy* if the secrecy of the previous session keys is not affected when the long term private keys of one or more entities are compromised. *Perfect forward secrecy* refers to the scenario when the long term private keys of all the participating entities are compromised.

Key-Compromise Impersonation Resilience. Suppose that A's long term private key is disclosed. Obviously an adversary who knows this value can now impersonate A since it is precisely the value which identifies A. We say that a protocol is *key-compromise impersonation resilient* if this loss will not enable an adversary to masquerade as other legitimate entities to A as well or obtain other entities' secret key.

Unknown Key-Share Resilience. In an unknown key-share attack, an adversary convinces a group of entities that they share a key with the adversary whereas in fact, the key is shared between the group and another party. This situation can be exploited in a number of ways by the adversary when the key is subsequently used to provide encryption of integrity.

Key Control Resilience. It should not be possible for any of the participants (or an adversary) to compel the session key to a preselected value or predict the value of the session key.

Joux [8] has initiated the development of one-round pairing-based tripartite Diffie-Hellman key agreement protocol in 2000. However, Shim [16] has pointed out that Joux's protocol does not provide authentication and therefore, it cannot resist the man-in-the-middle attack. Shim has further proposed an improved scheme which employs the public key infrastructure to overcome the security flaw in Joux's protocol. Unfortunately in 2005, Lin-Li [11] has identified the weaknesses of Shim's improved scheme and subsequently demonstrated its vulnerability to the insider impersonation attack and the key-compromise impersonation attack. In addition, Lin-Li has proposed their enhanced scheme by introducing an extra verification process in order to authenticate the communicating parties. They claimed that their enhanced scheme is secure and efficient. However, we have proven them wrong by showing a forging attack in [10] which renders their enhanced scheme totally insecure. On top of that, we have proposed an improved scheme in [10] mainly to fix the flaw that we have identified. However, recently, we have spotted another demerit in both original Lin-Li's scheme as well as our previous improved scheme, which allows a malicious adversary to carry out an insider replay attack successfully on both schemes. Furthermore, we have also discovered that both protocols in fact do not fulfill the forward secrecy property, which has been falsely claimed in both papers [10, 11]. Hence, we aim to address them thoroughly by proposing our latest improvements in this paper. Not only that, we also revise our previous improvement to optimize the efficiency mainly in the message verification processes by the message recipients.

We organize the structure of this paper as follows. In the next section, we will illustrate some basic properties of modified Weil pairings and some Diffie-Hellman assumptions. In Section 3, we will review our previous improved one-round pairing-based tripartite authenticated key agreement protocol. Then, we will present our attack and the weakness of the protocols in Section 4 and subsequently demonstrate our enhancements as well as the associated discussions in Section 5. Last but not least, we will conclude this paper in Section 6.

2. Preliminaries

Let p be a prime number such that $p = 2 \pmod 3$ and $p = 6q - 1$ for some prime $q > 3$. Let $E[q]$ be a supersingular curve defined by $y^2 = x^3 + 1$ over $\mathbf{F}_p$. Let $P \in E/\mathbf{F}_p$ be a generator of the group of points with order $q = (p + 1)/6$. Let μ_q be a subgroup of $\mathbf{F}_{p^2}^*$ that contains all elements of order q. The Weil pairing on the curve $E/\ \mathbf{F}_{p^2}*$ is a mapping e: $G_q \text{ x } G_q \rightarrow \mu_q$. Hence, we define the modified Weil pairing to be $\hat{e}$: $G_q \text{ x } G_q \rightarrow \mu_q$, $\hat{e}(P,Q) = e(P, \psi(Q))$, where $\psi(x, y) = (\zeta x, y)$, $1 \neq \zeta \in \mathbf{F}_{p^2}^*$ is a solution of $x^3 - 1 = 0 \pmod p$ and G_q is the group of points with order q. The modified Weil pairing then satisfies the following properties:

a) **Bilinear:** $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} = \hat{e}(abP, Q)$ for any P, $Q \in E[q]$ and $a, b \in Z_q*$.
b) **Alternative:** $\hat{e}(P, Q) = \hat{e}(Q, P)^{-1}$.
c) **Non-degenerate**: There exists a point $P \in G_q$ where $\hat{e}(P, P) \neq 1$.
d) **Polynomial-time computable**: There is an efficient algorithm to compute $\hat{e}(P,Q)$ for any P, $Q \in G_q$.

A bilinear map which satisfies all these properties above is known as *admissible bilinear*. It is noted that the modified Weil pairing associated with the supersingular elliptic curves or abelian varieties, can create such bilinear maps.

Now, we describe some hard cryptographic problems:
Bilinear Diffie-Hellman Problem (BDHP). Let $\mathbf{G}_1$, $\mathbf{G}_2$ be two groups of prime order q ($\mathbf{G}_1$ is an additive group and $\mathbf{G}_2$ is a multiplicative group). Let P be a generator of $\mathbf{G}_1$. Given a quadruple (P, aP, bP, cP) with $a, b, c \in Z_q^*$, compute $\hat{e}(P, P)^{abc} \in \mathbf{G}_2$. An algorithm α is deemed to have an advantage ε in solving the BDHP in $(\mathbf{G}_1, \mathbf{G}_2, \hat{e})$ based on the random choices of a, b, c in Z_q^* and the internal random operation of α if $\Pr[\alpha((P, aP, bP, cP)) = \hat{e}(P, P)^{abc}] \geq \varepsilon$.
Discrete Logarithm Problem (DLP). Given two groups of elements P and Q, such that $Q = nP$. Find the integer n whenever such an integer exists.

Throughout this paper, we assume that BDHP is a hard computational problem such that there is no polynomial time algorithm to solve BDHP and DLP with non-negligible probability.

3. Revisit of Previous Improved Lin-Li's Scheme

In this section, we review our previous improved scheme as described in [10]. Generally, this improved tripartite key agreement protocol consists of 4 phases, namely the *setup* phase, the *message exchange* phase, the *message verification* phase and the *key generation* phase. We now illustrate the scheme as follows:

Setup. Suppose that three protocol principals A, B and C wish to communicate with each other and agree on a common session key at the end of the protocol execution. The public domain parameters $(p, q, E, P, \hat{e}, H)$ are made common to all entities, where $H: \{0, 1\}^* \rightarrow Z_q^*$ is a predefined collision-free one-way hash function. Assume that the static public keys are exchanged via certificates. $Cert_A$ denotes A's public-key certificate, containing his static public key $Y_A = aP$ which uniquely identifies A and a certification authority CA's signature over this information. Similarly, $Cert_B$ and $Cert_C$ denote the certificate of B and C respectively, with $Y_B = bP$ and $Y_C = cP$ as their static public key, where b and c are B and C's corresponding static private key.

Message Exchange. Suppose that x, y and z are the ephemeral private keys chosen by A, B and C respectively. In a communication run, A computes

$$R_A = xP, \tag{1}$$
$$T_A = x \cdot (aP), \tag{2}$$
$$m_A = H(ax), \tag{3}$$
$$s_A = (ax)^{-1}(m_A + a) \bmod q. \tag{4}$$

B computes

$$R_B = yP, \tag{5}$$
$$T_B = y \cdot (bP), \tag{6}$$
$$m_B = H(by), \tag{7}$$
$$s_B = (by)^{-1}(m_B + b) \bmod q. \tag{8}$$

C computes

$$R_C = zP, \tag{9}$$
$$T_C = z \cdot (cP), \tag{10}$$
$$m_C = H(cz), \tag{11}$$
$$s_C = (cz)^{-1}(m_C + c) \bmod q. \tag{12}$$

Then, A, B and C perform the message broadcast as follows:

$$A \rightarrow B, C: \{R_A, T_A, m_A, s_A, Cert_A\},$$
$$B \rightarrow A, C: \{R_B, T_B, m_B, s_B, Cert_B\},$$
$$C \rightarrow A, B: \{R_C, T_C, m_C, s_C, Cert_C\}.$$

Message Verification. Upon receiving their respective peers' message, A, B and C carry out the message verification as follows:

$$\hat{e}(T_A, P) = \hat{e}(R_A, Y_A) \tag{13}$$
$$t_A = s_A^{-1} \bmod q \tag{14}$$
$$u_A = (t_A m_A) \bmod q \tag{15}$$
$$u_A\cdot P + t_A\cdot Y_A \overset{?}{=} T_A \tag{16}$$
$$\hat{e}(T_B, P) = \hat{e}(R_B, Y_B) \tag{17}$$
$$t_B = s_B^{-1} \bmod q \tag{18}$$
$$u_B = (t_B m_B) \bmod q \tag{19}$$
$$u_B\cdot P + t_B\cdot Y_B \overset{?}{=} T_B \tag{20}$$
$$\hat{e}(T_C, P) = \hat{e}(R_C, Y_C) \tag{21}$$
$$t_C = s_C^{-1} \bmod q \tag{22}$$
$$u_C = (t_C m_C) \bmod q \tag{23}$$
$$u_C\cdot P + t_C\cdot Y_C \overset{?}{=} T_C \tag{24}$$

- A authenticates T_B and T_C by verifying whether Eqs. (17) and (21) hold correspondingly. Then, A computes Eqs. (18), (19), (22) and (23), and verifies whether Eqs. (20) and (24) hold simultaneously. If any of these does not hold, A terminates the protocol run immediately.
- B authenticates T_A and T_C by verifying whether Eqs. (13) and (21) hold correspondingly. Then, B computes Eqs. (14), (15), (22) and (23), and verifies whether Eqs. (16) and (24) hold simultaneously. If any of these does not hold, B terminates the protocol run immediately.
- C authenticates T_A and T_B by verifying whether Eqs. (13) and (17) hold correspondingly. Then, C computes Eqs. (14), (15), (18) and (19), and verifies whether Eqs. (16) and (20) hold simultaneously. If any of these does not hold, C terminates the protocol run immediately.

Key Generation. If all the verifications are successful, for the session key $K_A = K_B = K_C$, A computes K_A by using Eq. (25), B computes K_B by using Eq. (26) and C computes K_C by using Eq. (27).

$$K_A = \hat{e}\,(Y_B + T_B, Y_C + T_C)^{a + ax} = \hat{e}\,(P, P)^{(a + ax)\,(b + by)\,(c + cz)} \tag{25}$$
$$K_B = \hat{e}\,(Y_A + T_A, Y_C + T_C)^{b + by} = \hat{e}\,(P, P)^{(a + ax)\,(b + by)\,(c + cz)} \tag{26}$$
$$K_C = \hat{e}\,(Y_A + T_A, Y_B + T_B)^{c + cz} = \hat{e}\,(P, P)^{(a + ax)\,(b + by)\,(c + cz)} \tag{27}$$

4. Further Demerits of Previous Improved Lin-Li's Scheme

4.1 The Insider Replay Attack

In both the original Lin-Li's scheme and our previous improved scheme, we discover that these schemes are in fact susceptible to another cryptanalytic attack, namely the insider replay attack. Note that an insider adversary can easily impersonate a protocol principal by replaying his message from a previous session. Suppose that B is an insider adversary who has eavesdropped and obtained $\{R_A', T_A', m_A', s_A', Cert_A\}$ from a previous session involving A, B and C. B now wishes to cheat another protocol participant D (with d/Y_D as his static private/public key pair) by means of initiating a protocol run with D while at the same time, B plays another role as A. We assume that A does not know anything about this protocol run and we define B_A as B masquerading as A. The insider replay attack can be carried out as follows:

$$B_A \rightarrow B, D\text{: } \{R_A', T_A', m_A', s_A', Cert_A\},$$
$$B \rightarrow B_A, D\text{: } \{R_B, T_B, m_B, s_B, Cert_B\},$$
$$D \rightarrow B_A, B\text{: } \{R_D, T_D, m_D, s_D, Cert_D\},$$

where k is an ephemeral private key chosen by D, $R_D = kP$, $T_D = k\cdot(dP)$, $m_D = H(dk)$ and $s_d = (dk)^{-1}(m_d + d) \bmod q$. On receipt of B and B_A's message, D verifies their message as per the protocol specification. Apparently, D would authenticate B's message successfully since B generates his message legally. However, note that B_A is also able to convince D in accepting A's old message since D would find Eqs. (13) and (16) hold and mistakenly believe that A is active in this session. Eventually, D computes the session key as

$$K_D = \hat{e}\,(Y_A' + T_A', Y_B + T_B)^{d + dk} = \hat{e}\,(P, P)^{(a + ax')\,(b + by)\,(d + dk)}$$

while B/B_A computes the session key as

$$K_B = K_{BA} = \hat{e}\,(Y_A' + T_A', Y_D + T_D)^{b + by} = \hat{e}\,(P, P)^{(a + ax')\,(b + by)\,(d + dk)}.$$

With this, B successfully cheats D by replaying A's message from a previous session and subsequently impersonating A in a particular session. As a result, B and D agree upon a session key for the tripartite key agreement scheme without A's presence. Since Lin-Li do not take the old communication reuse issue into their protocol design consideration, an insider adversary can easily bring any "virtual entity" into the communication scene by means of retransmitting the virtual entity's old message. The consequence of this attack can be disastrous if the impersonated party is a referee, a key escrow or an auditor.

4.2 The Absence of Forward Secrecy Property

Besides the replay attack, we also found out that the conjectured perfect forward secrecy of both the original Lin-Li's scheme and our previous improved scheme are falsely claimed. Consider that A, B and C have completed a session and computed a common session key successfully in prior. Assume that the adversary, eavesdropping on the channel, has obtained all the messages exchanged in that session. Suppose that A's private key, a has been compromised later by some means. With the eavesdropped m_A, the adversary can now derive A's ephemeral key, x by computing

$$\begin{aligned}(a(m_A + a)^{-1}s_A)^{-1} &= (a(m_A + a)^{-1}(ax)^{-1}(m_A + a))^{-1}\\ &= x \bmod q\end{aligned}$$

and reconstruct the session key by using Eq. (25). Similarly, the compromise of the other party's private key would render the previous established sessions insecure. Hence, we conclude that both the protocols do not satisfy the forward secrecy property (not even the partial forward secrecy).

5. Improvements on Lin-Li's Scheme and Related Discussions

In this section, we aim to address the insider replay attack by proposing an enhanced protocol which recovers the perfect forward secrecy property and preserves all other desired security attributes.

5.1 Protocol Description

Initially, we let p_A, p_B and $p_C \in Z_q^*$ to be the timestamp generated by A, B and C respectively. With this, we define our improved protocol as follows:

Setup. This enhanced protocol has the same initial setting as the previous one.

Message Exchange. A computes T_A and m_A by using Eqs. (2) and (3) respectively. Then A calculates

$$W_A = xH(x)\cdot(aP), \tag{28}$$

$$n_A = H(T_A, W_A, p_A), \tag{29}$$

$$s_A = (axH(x))^{-1}(m_A + an_A) \bmod q, \tag{30}$$

B computes T_B and m_B by using Eqs. (6) and (7) respectively. Then B calculates

$$W_B = yH(y)\cdot(bP), \tag{31}$$

$$n_B = H(T_B, W_B, p_B), \tag{32}$$

$$s_B = (byH(y))^{-1}(m_B + bn_B) \bmod q. \tag{33}$$

C computes T_C and m_C by using Eqs. (10) and (11) respectively. Then C calculates

$$W_C = zH(z)\cdot(cP), \tag{34}$$

$$n_C = H(T_C, W_C, p_C), \tag{35}$$

$$s_C = (czH(z))^{-1}(m_C + cn_C) \bmod q. \tag{36}$$

Then, A, B and C perform the message broadcast as follows:

$$A \rightarrow B, C\text{: } \{W_A, T_A, m_A, s_A, p_A, Cert_A\},$$
$$B \rightarrow A, C\text{: } \{W_B, T_B, m_B, s_B, p_B, Cert_B\},$$
$$C \rightarrow A, B\text{: } \{W_C, T_C, m_C, s_C, p_C, Cert_C\}.$$

Message Verification. Upon receiving their respective peers' message, A, B and C carry out the message verification as follows:

$$u_A\cdot P + t_A\cdot\ n_A\cdot Y_A \overset{?}{=} W_A \tag{37}$$

$$u_B\cdot P + t_B\cdot\ n_B\cdot Y_B \overset{?}{=} W_B \tag{38}$$

$$u_C\cdot P + t_C\cdot\ n_C\cdot Y_C \overset{?}{=} W_C \tag{39}$$

- A checks whether p_B and p_C lie within the acceptable time interval. Then, A computes n_B and n_C by using Eqs. (32) and (35). Subsequently, A computes t_B, u_B, t_C and u_C by using Eqs. (18), (19), (22) and (23) respectively, and verifies whether Eqs. (38) and (39) hold simultaneously. If any of these does not hold, A terminates the protocol run immediately.
- B checks whether p_A and p_C lie within the acceptable time interval. B then computes n_A and n_C by using Eqs. (29) and (35). Subsequently, B computes t_A, u_A, t_C and u_C by using Eqs. (14), (15), (22) and (23) respectively, and verifies whether Eqs. (37) and (39) hold simultaneously. If any of these does not hold, B terminates the protocol run immediately.
- C checks whether p_A and p_B lie within the acceptable time interval. Then C computes n_A and n_B by using Eqs. (29) and (32). Subsequently, C computes t_A, u_A, t_B and u_B by using Eqs. (14), (15), (18) and (19) respectively, and verifies whether Eqs. (37) and (38) hold simultaneously. If any of these does not hold, C terminates the protocol run immediately.

Key Generation. If all the verifications are successful, A computes K_A by using Eq. (40), B computes K_B by using Eq. (41) and C computes K_C by using Eq. (42).

$$K_A = H(Y_A, Y_B, Y_C, \hat{e}(Y_B + T_B, Y_C + T_C)^{a+ax}, W_A, T_A, m_A, s_A, p_A, W_B, T_B, m_B, s_B, p_B, W_C, T_C, m_C, s_C, p_C) \tag{40}$$

$$K_B = H(Y_A, Y_B, Y_C, \hat{e}(Y_A + T_A, Y_C + T_C)^{b+by}, W_A, T_A, m_A, s_A, p_A, W_B, T_B, m_B, s_B, p_B, W_C, T_C, m_C, s_C, p_C) \tag{41}$$

$$K_C = H(Y_A, Y_B, Y_C, \hat{e}(Y_A + T_A, Y_B + T_B)^{c+cz}, W_A, T_A, m_A, s_A, p_A, W_B, T_B, m_B, s_B, p_B, W_C, T_C, m_C, s_C, p_C) \tag{42}$$

5.2 Discussions

Now, we highlight the enhancements that we have made and we then analyze this enhanced protocol to ensure that the soundness of the protocol remains and the existing flaws have been eliminated.

For combating the insider replay attack, countermeasures can be applied in two ways:

(1) Use timestamp to guarantee freshness of the protocol, providing that time synchronization is feasible. In this method, each protocol participants adds the current time to his message before broadcasting it, and upon receiving it, the message will be checked by the message recipients whether the timestamp lies within an acceptable window of their current time.

(2) Employ nonces (random challenges) where each protocol participant generates a random number and broadcasts it to the other protocol participants. This nonce is then returned together with the message after processing with some cryptographic function.

Since method (2) requires an additional round (which results in a total of 2 rounds) of execution within a protocol run, we modified the protocol by using method (1) to secure against the insider replay attack while maintaining one round of execution as in the original protocol.

Lemma 1. Our improved protocol is able to withstand the insider replay attack, provided that time synchronization is feasible.

Proof. By incorporating timestamp into our enhanced protocol, the insider replay attack as shown in the previous section would be infeasible since the timestamp would expired if it is reused in other subsequent sessions. Note that we also bind the timestamp value to the signature (s_A, s_B and s_C) of the corresponding party to ensure that any modification to the timestamp will be detected by the message recipients as they verify Eqs. (37), (38) or (39).

Perfect forward secrecy can always be provided by a key agreement scheme at a higher computational cost. As describe in the previous subsection, we have introduced an extra parameter (W_A, W_B and W_C) to be computed (where the hashed value of each parameter, n_A, n_B, n_C is respectively bound to s_A, s_B and s_C) and sent along with the other messages by each communicating entity. Furthermore, we have also modified the signature s_A, s_B and s_C, and the verification processes in Eqs. (37), (38) and (39) such that the comparisons are done against W_A, W_B and W_C rather than T_A, T_B and T_C respectively. These steps are in fact necessary to conceal the particular ephemeral key if the private keys of all A, B and C have been compromised.

Lemma 2. Our improved protocol offers perfect forward secrecy if the hash function is strictly one-way and collision-free.
Proof. By considering the similar scenario in Section 4.2, the adversary is now restricted from deriving the respective ephemeral key after the particular entity's private key has been exposed. Apparently if the adversary compute

$$\begin{aligned}(a(m_A + a)^{-1}s_A)^{-1} &= (a(m_A + a)^{-1}(axH(x))^{-1}(m_A + a))^{-1}\\ &= xH(x) \bmod q,\end{aligned}$$

she would be unable to gain any information about the ephemeral key x from $xH(x)$ since $H(x)$ is conjectured to be perfectly one-way and collision-free. Also, note that the shared secret $\hat{e}\,(P, P)^{(a + ax)\,(b + by)\,(c + cz)}$ is derived by using peers' information Y_i and T_i, instead of W_i, for $i \in \{A, B, C\}$. Hence, since $xH(x)$ is not involved in the construction of session key, we speculate that the exposure of all entities' private keys would not affect the other established sessions.

Besides that, we have revised our previous improvement in proposing this enhanced scheme in order to optimize the computational efficiency. Note that each verification procedure in Eqs. (13), (17) and (21) requires 2 expensive pairing computations chiefly to restrict T_A, T_B or T_C to be computed in the specified form (enhancement made corresponding to the forging attack presented in [10]). On top of that, an additional elliptic curve scalar point multiplication (R_A, R_B, and R_C) need to be computed by each protocol participant to realize such verification. Hence, in our enhanced protocol, we omit these redundant computation operations and bind an additional hashed value (n_A, n_B and n_C) to the signature s_A, s_B and s_C respectively to achieve the same purpose.

Lemma 3. Our improved protocol is able to resist the forging attack presented in [10], despite omitting the previous proposed verification steps.
Proof. In this enhanced protocol, we do not restrict T_A, T_B and T_C to be in their specified form. Instead, we bind these values in the signatures s_A, s_B and s_C respectively, whereby the message recipients can verify them efficiently in Eqs. (37), (38) and (39). As a result, forging attack presented in [10] is impossible as the adversary is not able to forge any signature by assigning a random value to it before determining the value of T_A, T_B or T_C. Eventually, verification step in Eqs. (37), (38) or (39) would eventually fail if the parameters are not computed according to their designated way. Not only that, our current improved scheme turns out to be highly efficient compared to the previous one. Note that our current improvements only require each protocol participant to perform an additional scalar point multiplications and an extra hash function to provide all desired security features while resisting all known attacks compared to the insecure original Lin-Li's scheme while our previous improved scheme [10] would require each protocol participant to perform four extra expensive pairing operations and one extra scalar point multiplication to partially secure the original scheme.

Similar to Choo et al.'s strategy [6], rather than having only the shared secret as the session key, we construct the session key by computing the hash value of the protocol participants' identity, the shared secret and the session identifiers basically to avoid many other cryptanalytic attacks, such as the unknown key share attack, the key replicating attack [9] and the triangle attack [2]. Note that our modification in this enhanced protocol does not affect the other desired security features of the original protocol. Please refer to [10] and [11] for the full details of these attributes.

6. Conclusion

In a nutshell, we have proven the vulnerability of both the original Lin-Li's scheme and our previous improved protocol to the insider replay attack. We have also pointed out that they do not satisfy the valuable forward secrecy property. Based on these deficiencies, we have suggested several essential modifications to the original Lin-Li's scheme so as to resist all known cryptanalytic attacks and to optimize the efficiency especially in the message verification phase. To justify these improvements, we have further provided detailed discussions to illustrate the significance of our improvements in resisting the known attacks and providing the perfect forward secrecy feature, without affecting the other desired security properties of Lin-Li's protocol.

References

1. S.S. Al-Riyami and K.G. Paterson, Tripartite Authenticated Key Agreement Protocols from Pairings, Cryptology ePrint Archive: Report, (035)(2002).
2. M. Burmester, On the Risk of Opening Distributed Keys, Crypto'94, LNCS, vol.839, 1994, pp. 308-317.
3. Z.H. Cheng, L. Vasiu, and R. Comley, Pairing-based One-round Tripartite Key Agreement Protocols, Cryptology ePrint Archive, Report (079)(2004).
4. H.Y. Chien, Comments: Insider Attack on Cheng et al.'s Pairing-based Tripartite Key Agreement Protocols, Cryptology ePrint Archive: Report, (013)(2005).
5. H.Y. Chien and R.Y. Lin, An Improved Tripartite Authenticated Key Agreement Protocol Based on Weil Pairing, Int. J. Appl. Sci. Eng., 2005. 3 , 1, pp. 13-18.

6. K.-K. R. Choo, C. Boyd and Y. Hitchcock, On Session Key Construction in Provably-Secure Key Establishment Protocols, Mycrypt 2005, LNCS, vol. 3715, 2005, pp. 116-131.
7. J.S. Chou, C.H. Lin, C.H. Chiu, Weakness of Shim's New ID-based Tripartite Multiple-key Agreement Protocol, Cryptology ePrint Archive: Report, (457)(2005).
8. A. Joux, A One-round Protocol for Tripartite Diffie-Hellman, Proceedings of the 4th International Algorithmic Number Theory Symposium (ANTS-IV), LNCS 1838, 2000, pp.385-394.
9. H. Krawczyk, HMQV: A High-Performance Secure Diffie-Hellman Protocol, Crypto'05, LNCS, vol. 3621, 2005, pp. 546-566.
10. M.-H. Lim, S.G. Lee, Y.H. Park, H.J. Lee, An Enhanced One-Round Pairing-based Tripartite Authenticated Key Agreement Protocol, ICCSA2007, LNCS, vol. 4706, 2007, pp. 503-513.
11. C.H. Lin, H.H. Li, Secure One-Round Tripartite Authenticated Key Agreement Protocol from Weil Pairing, Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA 2005), pp.135-138.
12. D. Nalla, ID-based Tripartite Key Agreement with Signatures, Cryptology ePrint Archive: Report, (144)(2003).
13. D. Nalla and K.C. Reddy, ID-based tripartite Authenticated Key Agreement Protocols from pairings, Cryptology ePrint Archive: Report, (004)(2003).
14. K. Shim, Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols, Cryptology ePrint Archive: Report, (122)(2003).
15. K. Shim, Efficient ID-based Authenticated Key Agreement Protocol based on Weil Pairing, Electronics Letters, Vol. 39, no. 8, April, 2003, pp.653-654.
16. K. Shim, Efficient One-round Tripartite Authenticated Key Agreement Protocol from Weil Pairing, Electronics Letters, Vol. 39, no. 2, January, 2003, pp.208-209.
17. S.B. Wilson, and A. Menezes, Authenticated Diffie-Hellman key agreement protocols, Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC '98), LNCS, 1998, vol.1556, pp. 339-361.
18. S.B. Wilson, D. Johnson and A. Menezes, Key Agreement Protocols and their Security Analysis, Proceedings of the 6th IMA International Conference on Cryptography and Coding, LNCS, vol. 1355, pp. 30-45, 1998.
19. F.G. Zhang, S.L. Liu, K.J. Kim, ID-based One Round Authenticated Tripartite Key Agreement Protocol with Pairings, Cryptology ePrint Archive: Report, (122)(2002).