Pairing-Based One-Round Tripartite Key Agreement Protocols

Zhaohui Cheng, Luminita Vasiu and Richard Comley

School of Computing Science, Middlesex University
White Hart Lane, London N17 8HR, United Kingdom
{m.z.cheng,l.vasiu,r.comley}@mdx.ac.uk

Abstract. Since Joux published the first pairing-based one-round tripartite key agreement protocol [13], many authenticated protocols have been proposed. However most of them were soon broken or demonstrated not to achieve some desirable security attributes. In this paper we present a protocol variant based on Shim's work [20]. As the formalized model of this type of AK protocols is not mature, the security properties of the protocol are heuristically investigated by attempting a list of attacks. The attack list presented in the paper has both the importance in theory and the meaning in practice and can be used to evaluate other tripartite and group key agreement protocols.

1 Introduction

Key Agreement Protocols (KAP) are the mechanisms by which two or more parties can establish an agreed secret key over a network controlled by adversaries. Normally the established key varies on each execution (session) of the protocol. If in a protocol one party is assured that no other party aside from the specifically identified party (or parties) may gain access to the particular established secret key, then the key agreement protocol is said to provide key authentication. A key agreement protocol which provides mutual key authentication between (or among) parties is called an *Authenticated Key agreement* (AK). Although an AK provides key authentication, one party is not sure whether the other party (or parties) actually has possession of the established secret; otherwise, the protocol is said to provide key confirmation. If a key agreement protocol holds both key authentication and key confirmation, it is called an *Authenticated Key agreement with key Confirmation* (AKC) [15].

A number of security properties are generally believed to be necessary (or good) for an AK or AKC [8].

1. *Known session key security.* Each execution of the protocol should result in a unique secret session key. The compromise of one session key should not compromise the keys of other sessions (e.g., the parallel sessions, previous sessions and future sessions).
2. *Forward secrecy.* If the long-term private keys of one or more entities are compromised, the secrecy of previously established session keys should not be affected. We say that a protocol has *partial forward secrecy* if one or more but not all the entities' long-term keys can be corrupted without compromising previously established session keys, and we say that a protocol has *perfect forward secrecy* (PFS) if the long-term keys of all the entities involved may be corrupted without compromising any session key previously established by these entities.
3. *Key-compromise impersonation resilience.* The compromise of entity A's long-term private key (keys) will allow an adversary to impersonate A, but it should not enable the adversary to impersonate other entities to A.
4. *Unknown key-share resilience.* Entity A should not be able to be coerced into sharing a key with entity C when in fact A thinks that he is sharing the key with some entity B.

There are a great number of two-party or two-party with online trusted center protocols (refer to [15] [5] for surveys) available in the literature. Many group key agreements have been developed as well (a list can be found in [18] [5]). To formally evaluate a protocol's security, formalized models of AK's and

AKC's have been developed, e.g., the indistinguishability-based models [6][8] and the simulation-based models [11][19]. Some formalizing work has also been attempted for group key agreement protocols, e.g., [3][18].

Apart from the security requirements, the communication and computation cost are also the critical considerations when designing key agreement protocols. In 2000, Joux presented a new efficient one-round tripartite key agreement protocol [13] by using an old mathematical tool, i.e., pairings on elliptic curves.

Definition 1 *A (symmetric) pairing is a bilinear map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with two cyclic group $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, which has the following properties [4]:*

1. *Bilinear: For all $P, Q, R, S \in \mathbb{G}_1$, $\hat{e}(P+Q, R+S) = \hat{e}(P,R) \cdot \hat{e}(P,S) \cdot \hat{e}(Q,R) \cdot \hat{e}(Q,S)$*[1]*.*
2. *Non-degenerate: For a given point $Q \in \mathbb{G}_1$, $\hat{e}(Q,R) = 1_{\mathbb{G}_2}$ for all $R \in \mathbb{G}_1$ if and only if $Q = 1_{\mathbb{G}_1}$.*
3. *Computable: There is an efficient algorithm to compute $\hat{e}(P,Q)$ for any $P, Q \in \mathbb{G}_1$.*

By using the pairing computation and a Diffie-Hellman type scheme, the protocol[2] requires each party to transmit only a single broadcast message to establish an agreed session key among three parties. After

$$\begin{array}{l} \hline 1.\ A \rightarrow B, C : aP \\ 2.\ B \rightarrow A, C : bP \\ 3.\ C \rightarrow A, B : cP \\ \hline \end{array}$$

Fig. 1. Joux's One-round Tripartite Key Agreement

the session, A computes $K_A = (bP, cP)^a$; B computes $K_B = (aP, cP)^b$ and C computes $K_C = (aP, bP)^c$. The established session key is $K = K_A = K_B = K_C = (P,P)^{abc}$. The protocol is secure against passive adversaries based on the Bilinear Diffie-Hellman (BDH) assumption [4].

Assumption 1 Bilinear Diffie-Hellman Assumption (BDH) *[4] Let $\mathcal{G}$ be a parameter generator which with system parameters 1^k as input generates two cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q and a bilinear map $\hat{e}$. We define the advantage of an algorithm $\mathcal{A}$ in solving the problem (given $\langle P, aP, bP, cP \rangle$, to compute $\hat{e}(P,P)^{abc}$) by:*

$$\begin{aligned} Adv_{\mathcal{G},\mathcal{A}}(k) = Pr[\ & \mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc} | \\ & \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle \leftarrow \mathbb{G}(1^k), a \text{ generator } P \leftarrow \mathbb{G}_1, a, b, c \xleftarrow{R} \mathbb{Z}_q^*]. \end{aligned}$$

For any randomized polynomial time (in k) algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{G},\mathcal{A}}(k)$ is negligible.

However, like the basic Diffie-Hellman key agreement protocol, Joux's protocol also suffers from the man-in-the-middle attack because it does not authenticate the communicating parties. To address this security threat, many one-round authenticated key agreement protocols have been proposed. Basically these protocols can be divided into two broad categories, i.e., certification-based protocols including [1] [20] and identity-based protocols such as [16] [17] [21] [24]. Unfortunately most of them have been broken or shown not to achieve some good security attributes by the attacks presented in [9] [23] [21] [22]. In fact currently there is no one-round tripartite AK protocol achieving all the four security attributes.

In this paper, we are going to strengthen Shim's certification-based protocol [20] to prevent the identified attacks in the literature. Then we heuristically evaluate the protocol' security by attempting a list of attacks. The attack list has both the importance in theory, i.e., some attacks demonstrate

[1] In particular $\hat{e}(sP, tR) = \hat{e}(P,R)^{st}$ for all $P, R \in \mathbb{G}_1$ and $s, t \in \mathbb{Z}_q^*$.

[2] Note that the original protocol used the asymmetric pairing.

the incapability of current formulations, and the meaning in practice, i.e., every existing protocol can be broken or demonstrated not to achieve some security property by one or more attacks in the list. We hope that the attack list can be used as a reference to design this type of protocol in the future before the formalized model for this type of protocol becomes mature. Note that tripartite protocols are the degenerated case of group protocols, hence, sound group protocols also need to counter the listed attacks, so to achieve related security properties. The paper is organized as follows. First, we explain Shim's protocol and the attack on it. Then we present the protocol variant and a list of attacks. In Section 4, we evaluate the the influence of the attacks and propose a counter-measure. We draw a conclusion in the end.

2 Shim's Protocol

To provide implicit authentication, one method is to introduce certifications into the system. Party A with an identifer I_A, a long-term private key x_A and the public key $y_A = x_A P$ obtains a certification $Cert_A = (I_A \| y_A \| P \| S_{CA}(I_A \| y_A \| P))$ from a certification authority (CA). S_{CA} is the signature of CA and P is the system parameter. Shim presented a certification-based protocol in [20]. In the protocol each party randomly chooses an integer from $\mathbb{Z}_q^*$ and broadcasts a message consisting of its certification and the scalar result of its public key with the chosen random integer. After exchanging the messages, each party

$$\begin{array}{l}
1.\ A \rightarrow B, C : T_A = a(xP),\ Cert_A \\
2.\ B \rightarrow A, C : T_B = b(yP),\ Cert_B \\
3.\ C \rightarrow A, B : T_C = c(zP),\ Cert_C
\end{array}$$

Fig. 2. Shim's Certification-Based Protocol

computes the session key using one of the following functions. But it was shown that this protocol does not

$$\begin{array}{l}
K_A = \hat{e}(T_B, T_C)^{ax\hat{e}(Y_B,Y_C)^x} = \hat{e}(P,P)^{axbycz\hat{e}(P,P)^{xyz}} \\
K_B = \hat{e}(T_A, T_C)^{by\hat{e}(Y_A,Y_C)^y} = \hat{e}(P,P)^{axbycz\hat{e}(P,P)^{xyz}} \\
K_C = \hat{e}(T_A, T_B)^{cz\hat{e}(Y_A,Y_B)^z} = \hat{e}(P,P)^{axbycz\hat{e}(P,P)^{xyz}}
\end{array}$$

achieve the key-compromise impersonation resilience attribute [23]. If adversary E knows A's private key x, then it can randomly choose integer u and broadcast message (2) to impersonate B to A and C. After

$$\begin{array}{l}
1.\ A \rightarrow E_B, C : T_A = a(xP),\ Cert_A \\
2.\ E_B \rightarrow A, C : T_B = uP, \quad\ Cert_B \\
3.\ C \rightarrow A, E_B : T_C = c(zP),\ Cert_C
\end{array}$$

Fig. 3. Key-Compromise Impersonation Attack

the session, E can compute the session key $K_E = \hat{e}(T_A, T_C)^{u\hat{e}(Y_B,Y_C)^x} = \hat{e}(P,P)^{axucz\hat{e}(P,P)^{xyz}} = K_A = K_C$.

3 Protocol Variant

In this section, we present a variant protocol based on Shim's work. We heuristically evaluate the protocol's security by attempting a list of attacks.

As shown in the last section, Shim's certification-based protocol is vulnerable to the key-compromise impersonation attack. By introducing one more element in each message we can resolve this problem. In

$$\begin{array}{c}
\hline
1.A \rightarrow B, C : T_A^1 = aP, T_A^2 = a(xP), Cert_A \\
2.B \rightarrow A, C : T_B^1 = bP, T_B^2 = b(yP), Cert_B \\
3.C \rightarrow A, B : T_C^1 = cP, T_C^2 = c(zP), Cert_C \\
\hline
\end{array}$$

Fig. 4. Proposal 1

the new protocol, each party randomly chooses an integer from $\mathbb{Z}_q^*$ and broadcasts a message consisting of two scalars and its certification. One scalar is the result of the random integer timing the system parameter P, the other is the result of the random integer timing the party's public key. After exchanging the messages, party A verifies $\hat{e}(T_B^1, yP) = \hat{e}(T_B^2, P)$ and $\hat{e}(T_C^1, zP) = \hat{e}(T_C^2, P)$. Party B and C perform the similar operations. After the check step each party computes the session key $K = \hat{e}(P, P)^{axbycz}$ by using one of the following functions respectively.

$$\begin{array}{c}
\hline
K_A = \hat{e}(T_B^2, T_C^2)^{ax} = \hat{e}(P, P)^{axbycz} \\
K_B = \hat{e}(T_A^2, T_C^2)^{by} = \hat{e}(P, P)^{axbycz} \\
K_C = \hat{e}(T_A^2, T_B^2)^{cz} = \hat{e}(P, P)^{axbycz} \\
\hline
\end{array}$$

Security Analysis In the protocol the check step $\hat{e}(T_B^1, yP) = \hat{e}(T_B^2, P)$ guarantees that for message (2), it is hard to compute integer v and b such that $T_B^2 = vP = byP$ without knowing y. For message (1) and (3), similar guarantees are achieved. This assertion is described by Theorem 1.

Assumption 2 Discrete Logarithm (DL) Assumption. *In a cyclic group $\mathbb{G}_1$ with prime order q and a generator $P \in \mathbb{G}_1$, the problem given P and $Y = yP$ with random integer $y \in_R \mathbb{Z}_q^*$ to compute y is hard.*

Theorem 1 *Based on the DL assumption, the problem, given $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, yP\rangle$ where y is a random integer in $\mathbb{Z}_q^*$, to find integer v and b such that $\hat{e}(P, P)^v = \hat{e}(P, P)^{by}$ is hard.*

Proof: The proof is straightforward. If there exists a randomized polynomial algorithm $\mathcal{A}$ to find the integer v and b with non-negligible advantage, we can simply use this algorithm to solve the DL problem by returning $y = v/b$.

However the above theorem is only useful to evaluate some attacks presented in the following part. In fact, an adversary does not need to find both b and v such that $\hat{e}(P, P)^v = \hat{e}(P, P)^{by}$ at the same time to launch an attack. To break the protocol, an adversary simply needs to find v and bP (instead of b) satisfying equation $\hat{e}(vP, P) = \hat{e}(bP, yP)$. Fortunately it seems to be hard to solve this problem formally defined as following.

Assumption 3 *Given $\langle q, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, P, yP\rangle$ with $y \xleftarrow{R} \mathbb{Z}_q^*$, it is hard to find $v \in \mathbb{Z}_q^*$ and $Q = bP \in \mathbb{G}_1^*$ such that $\hat{e}(P, P)^v = \hat{e}(Q, yP) = \hat{e}(P, P)^{by}$.*

This assumption is used in a few papers, e.g., [2]. Note that the problem in the assumption seems easier than the DL or the DH problem. If there exists an polynomial-time algorithm which solves the problem and at the same time finds the complete bits of b, then we can construct a polynomial-time algorithm to solve the DL problem as proved in Theorem 1. If there exists an polynomial-time algorithm which solves the problem without making use of any bit of b (e.g., hard-core bits of b) which cannot be computed by effective means (probabilistic polynomial-time algorithm) based on the DL assumption (note that the least significant bit of b can be easily computed), then the algorithm can be used to solve the related DH problem. However, we cannot rule out the possibility that the algorithm can get some bits of b in some way, so to compute v and bP because we cannot restrict the adversary's behavior to generate bP (because an adversary can possibly get some valuable information by interacting with the outside). On the other hand, if we assume that (1) an adversary can only choose b to generate bP; or (2) randomly chooses bP directly from $\mathbb{G}_1^*$; or (3) that for a $Q = bP$, (3.i) either the adversary knows b; or (3.ii) Q is a random element for the adversary, then Assumption 3 is true based on the DL and the DH assumption. Apart from Assumption 3, the security of the protocol is based on the following theorem. The proof of the theorem is straightforward.

Theorem 2 *Based on the BDH assumption, the problem, given $\langle q, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, P, a, b, c, xP, yP, zP, \rangle$ such that $a, b, c, x, y, z \in \mathbb{Z}_q^*$ and x, y, z are random integers, to compute $\hat{e}(P, P)^{axbycz}$ is hard.*

Note that the above theorems only indicate that the used primitives in the protocol have foundations, but this does not guarantee that the protocol is secure and achieves the good security attributes. There are some formalized security models of tripartite and group key agreement protocols used to prove the security, e.g., [1][3]. Basically these models are the extensions of Bellare-Rogaway's work [6]. As analyzed in [10], these models do not fully address the formalization problem of AK's, specially for those which use only (general) commutative[3] computation on random flips in the agreed key generation functions (we will show some cases later). Hence we do not try to prove the security of our proposal in one security model, but heuristically evaluate the security by attempting a list of attacks. In fact we can find that all the existing pairing-based one-round tripartite protocols are demonstrated not be able to achieve some security property by one of the attacks. We list these attacks, some are known in the literature, even though some of them are not feasible in our protocol and we hope that this list as a reference would help to design more secure protocols.

1. **The Man-In-The-Middle Attack.** E replaces the broadcast messages with new ones by choosing its own integers $a', b', c' \in \mathbb{Z}_q^*$. After verification, each party computes the session key respectively. E

1.	$A \to B, C$	$: T_A^1 = aP, T_A^2 = a(xP),$	$Cert_A$
$1'$.	$E_A \to B, C$	$: T_A^{1'} = a'P, T_A^{2'} = a'(xP),$	$Cert_A$
2.	$B \to A, C$	$: T_B^1 = bP, T_B^2 = b(yP),$	$Cert_B$
$2'$.	$E_B \to A, C$	$: T_B^{1'} = b'P, T_B^{2'} = b'(yP),$	$Cert_B$
3.	$C \to A, B$	$: T_C^1 = cP, T_C^2 = c(zP),$	$Cert_C$
$3'$.	$E_C \to A, B$	$: T_C^{1'} = c'P, T_C^{2'} = c'(zP),$	$Cert_C$

Fig. 5. Attack 1: Man-In-The-Middle Attack

cannot compute any of them although it knows a', b' and c'.
Notation: In the above attack, an adversary E impersonating A (denoted by E_A) intercepts and replaces message (1) sent from party A with a new message ($1'$), hence party B and C will only receive message $1'$ and regard it as the one generated by A. E also impersonates B and C in this attack. The following attack presentations use the similar notation.

[3] A *general-commutative* computation is such an operation that dos not distinguish which entity involved is initiator, and which is the protocol's responder [10].

$$K_A = \hat{e}(T_B^{2'}, T_C^{2'})^{ax} = \hat{e}(P,P)^{axb'yc'z}$$
$$K_B = \hat{e}(T_A^{2'}, T_C^{2'})^{by} = \hat{e}(P,P)^{a'xbyc'z}$$
$$K_C = \hat{e}(T_A^{2'}, T_B^{2'})^{cz} = \hat{e}(P,P)^{a'xb'ycz}$$

2. **The Key-Compromise Impersonation Attack.**
   - Case 1. E knows A's long-term private key x and it impersonates B to A and C by generating message (2). E cannot compute the session key $K = \hat{e}(P,P)^{axbycz}$, although it knows x and b.

1. $A \rightarrow E_B, C : T_A^1 = aP, T_A^2 = a(xP),\ Cert_A$
2. $E_B \rightarrow A, C : T_B^1 = bP, T_B^2 = b(yP),\ Cert_B$
3. $C \rightarrow A, B\ \ : T_C^1 = cP, T_C^2 = c(zP),\ Cert_C$

Fig. 6. Attack 2: Key-Compromise Impersonation Attack 1

   - Case 2. E knows A's long-term private key x and it impersonates B and C to A by generating message (2) and (3). E cannot compute the session key: $K = \hat{e}(P,P)^{axbycz}$ although it knows x, b and c.

1. $A \rightarrow E_B, E_C : T_A^1 = aP, T_A^2 = a(xP),\ Cert_A$
2. $E_B \rightarrow A, E_C : T_B^1 = bP, T_B^2 = b(yP),\ Cert_B$
3. $E_C \rightarrow A, E_B : T_C^1 = cP, T_C^2 = c(zP),\ Cert_C$

Fig. 7. Attack 3: Key-Compromise Impersonation Attack 2

   - Case 3. E knows A's long-term private key x and it impersonates B by randomly choosing u to generate a valid message (2), so to compute the session key $K = \hat{e}(T_A^2, T_C^2)^u = \hat{e}(P,P)^{axucz}$. This attack presented in [23] is feasible to Shim's protocol [20], but infeasible to our protocol, because given an integer u, it is hard to find bP satisfying $\hat{e}(bP, yP) = \hat{e}(uP, P)$ if Assumption 3 is true.

1. $A \rightarrow E_B, C : T_A^1 = aP, T_A^2 = a(xP),\ Cert_A$
2. $E_B \rightarrow A, C : T_B^1 = vP, T_B^2 = uP,\ \ \ \ Cert_B$
3. $C \rightarrow A, E_B : T_C^1 = cP, T_C^2 = c(zP),\ Cert_C$

Fig. 8. Attack 4: Key-Compromise Impersonation Attack 3

3. **The Known Session Key Attack.**
   - Case 1. E tries to use the knowledge of the session key of a previous session to attack a following session. E knows A's long-term private key x and it impersonates B and C to A by generating message (2) and (3). E recovers the agreed key $K = \hat{e}(P,P)^{axbycz}$ of this session by some means, but this does not help E to get any meaningful information, even it knows x, b and c, to launch further attacks. For example in the following attack, the session key $K' = \hat{e}(P,P)^{a'xbycz}$ cannot be recovered by E even with knowing $K = \hat{e}(P,P)^{axbycz}, b, c$ and x.

$$\begin{array}{l}
\hline
1.\ A \to E_B, E_C : T_A^1 = aP, T_A^2 = a(xP),\ Cert_A \\
2.\ E_B \to A, E_C : T_B^1 = bP, T_B^2 = b(yP),\ Cert_B \\
3.\ E_C \to A, E_B : T_C^1 = cP, T_C^2 = c(zP),\ Cert_C \\
\hline
\end{array}$$

Fig. 9. Attack 5: Known Session Key Attack 1-Step 1

$$\begin{array}{l}
\hline
1.\ A \to E_B, E_C : T_A^{1'} = a'P, T_A^{2'} = a'(xP),\ Cert_A \\
2.\ E_B \to A, E_C : T_B^1 = bP, T_B^2 = b(yP),\quad Cert_B \\
3.\ E_C \to A, E_B : T_C^1 = cP, T_C^2 = c(zP),\quad Cert_C \\
\hline
\end{array}$$

Fig. 10. Attack 5: Known Session Key Attack 1-Step 2

- Case 2. E tries to use the knowledge of the session key(s) of a (or some) session(s) to attack a precedent session. For example a known-key conspiracy attack to TAK-2 [1] is presented in [22]. We show the attack here, because it vividly demonstrates the weakness of the formalized model in [1]. In a session of TAK-2, after exchanging the messages, each party uses one of the functions

$$\begin{array}{l}
\hline
1.\ A \to B, C : T_A = aP,\ Cert_A \\
2.\ B \to A, C : T_B = bP,\ Cert_B \\
3.\ C \to A, C : T_C = cP,\ Cert_C \\
\hline
\end{array}$$

Fig. 11. TAK-2

$$\begin{array}{c}
\hline
K_A = \hat{e}(bP, zP)^a \cdot \hat{e}(yP, cP)^a \cdot \hat{e}(bP, cP)^x \\
K_B = \hat{e}(aP, zP)^b \cdot \hat{e}(xP, cP)^b \cdot \hat{e}(aP, cP)^y \\
K_C = \hat{e}(aP, yP)^c \cdot \hat{e}(xP, bP)^c \cdot \hat{e}(aP, bP)^z \\
\hline
\end{array}$$

Session Key Generation Functions of ATK-2

to compute the session key $K_{ABC} = K_A = K_B = K_C = \hat{e}(P, P)^{abz+acy+bcx}$. However, ATK-2 is vulnerable to the known session key attack. Adversaries D and E, which has private key v and w and public key vP and wP respectively, can collude to attack a session among A, B and C. D and E launch the attack by engaging the following three sessions with A, B and C separately. If D and E compromise K_{ADE}, K_{BDE} and K_{CDE} (note that D and E cannot compute these

$$\begin{array}{l}
\hline
1'.\ \ A \to D, E : T_A = a'P,\ Cert_A \\
2'.\ \ D \to A, E : T_D = bP,\ \ Cert_D \\
3'.\ \ E \to A, D : T_E = cP,\ \ Cert_E \\
\hline
1''.\ D \to B, E : T_D = aP,\ \ Cert_D \\
2''.\ B \to D, E : T_B = b'P,\ Cert_B \\
3''.\ E \to B, D : T_E = cP,\ \ Cert_E \\
\hline
1'''.\ D \to C, E : T_D = aP,\ \ Cert_D \\
2'''.\ E \to C, D : T_E = bP,\ \ Cert_E \\
3'''.\ C \to D, E : T_C = c'P,\ Cert_C \\
\hline
\end{array}$$

Fig. 12. Attack 6: Known Session Key Attack 2 on ATK-2

keys), D and E can recover the session key K_{ABC} as follows:

$$\begin{aligned}K_{ABC} = K_{ADE} \cdot K_{BDE} \cdot K_{CDE} \cdot (\hat{e}(a'P, bP) \cdot \hat{e}(aP, b'P) \cdot \hat{e}(aP, c'P))^{-w} \\ \cdot(\hat{e}(a'P, cP) \cdot \hat{e}(b'P, cP) \cdot \hat{e}(bP, c'P))^{-v}\end{aligned}$$

The authors proved a tweaked version of ATK-2 in [2] in a model without the *Reveal* query which addresses the known session key attack. However this attack shows that such model does not define a strong enough security notion of this type of protocol. On the other hand this type of attack is not feasible in our protocol.

- Case 3. As analyzed in [10], for those AK's which use only (general) commutative computations on random flips in the agreed key generation functions, two attacks, the concatenation attack and the general concatenation attack, are possible. For example the concatenation attack is feasible to TAK-1 [1], but this attack is not applicable in our protocol. However the general concatenation attack can be launched by E as follows. Assume there are two concurrent sessions among A, B and C. E replaces B's broadcast in session 1 with B's message in session 2 and replaces A and C's broadcast in session 2 with the one in session 1 respectively. After exchanging messages, A's

$$Session_1 = \begin{cases} 1.\ \ A \to B, C \ \ : T_A^1 = aP, T_A^2 = a(xP), \quad Cert_A \\ 2.\ \ B \to A, C \ \ : T_B^1 = bP, T_B^2 = b(yP), \quad Cert_B \\ 2'.\ E_B \to A, C : T_B^{1'} = b'P, T_B^{2'} = b'(yP), Cert_B \\ 3.\ \ C \to A, B \ \ : T_C^1 = cP, T_C^2 = c(zP), \quad Cert_C \end{cases}$$

$$Session_2 = \begin{cases} 1.\ \ B \to A, C \ \ : T_B^{1'} = b'P, T_B^{2'} = b'(yP), Cert_B \\ 2.\ \ A \to B, C \ \ : T_A^{1'} = a'P, T_A^{2'} = a'(xP), Cert_A \\ 2'.\ E_A \to B, C : T_A^1 = aP, T_A^2 = a(xP), \quad Cert_A \\ 3.\ \ C \to A, B \ \ : T_C^{1'} = c'P, T_C^{2'} = c'(zP), Cert_C \\ 3'.\ E_C \to A, B : T_C^1 = cP, T_C^2 = c(zP), \quad Cert_C \end{cases}$$

Fig. 13. Attack 7: Known Session Key Attack 3

session key in session 1 is $K_A^1 = \hat{e}(T_B^{1'}, T_C^1)^{ax} = \hat{e}(P, P)^{axb'ycz}$ which is the same as B's session key $K_B^2 = \hat{e}(T_A^1, T_C^1)^{b'y} = \hat{e}(P, P)^{axb'ycz}$ in session 2. So if E reveals K_B^2, it knows A's agreed key in another session. Similar attacks can be performed to B and C.

- Case 4. The following attack is extremely unlikely in practice and has only theoretical meaning. In a session, after A and C generate and broadcast message (1) and (3) respectively, B generates and sends its own message (2) upon receiving message (1), (3). E intercepts message (2) and at the same time successfully discloses B's long-term private key y by some means. E randomly chooses integer $m \in \mathbb{Z}_q^*$ and replaces B's broadcast message with the new one as shown in (2′). Obviously the transcript of B for the session is different from the ones of A and C. Moreover

1. $A \to B, C$	: $T_A^1 = aP, T_A^2 = a(xP)$,	$Cert_A$
2. $B \to A, C$	: $T_B^1 = bP, T_B^2 = b(yP)$,	$Cert_B$
2′. $E_B \to A, C$	: $T_B^{1'} = T_B^1 + mP, T_B^{2'} = T_B^2 + m(yP)$,	$Cert_B$
3. $C \to A, B$	: $T_C^1 = cP, T_C^2 = c(zP)$,	$Cert_C$

Fig. 14. Attack 8: Known Session Key Attack 4

this session have been accepted by B before its long-term key y is exposed. Hence we can regard that B is in a different session from the one that A and C have engaged in (B is not partnered with A or C). At least in the existing indistinguishability-based models of AK's, this is the case. Now E reveals A's session key $K_A = \hat{e}(P,P)^{axbycz} \cdot \hat{e}(P,P)^{axmycz}$ by some means (in the formulation models [3][6], this can be done by a *Reveal* query). Then E can compute B's session key $K_B = \hat{e}(P,P)^{axbycz} = K_A \cdot \hat{e}(axP, czP)^{-my}$. In this protocol, it is required that E must know B's long-term private key to launch the attack. While in many other protocols, E can launch the attack successfully without knowing this information. For example, in the tripartite case, the attack is feasible to TAK-2, TAK-3 [1] and for two-party case examples can be found in [10]. We simply present the attack to TAK-2 as follows. After revealing B's

1.	$A \rightarrow B, C$	$: T_A^1 = aP$
2.	$B \rightarrow A, C$	$: T_B^1 = bP$
$2'$.	$E_B \rightarrow A, C$	$: T_B^{1'} = T_B^1 + mP$
3.	$C \rightarrow A, B$	$: T_C^1 = cP$

Fig. 15. Known Session Key Attack 4 on ATK

session key $K_B = \hat{e}(aP, zP)^b \cdot \hat{e}(xP, cP)^b \cdot \hat{e}(aP, cP)^y$, E can compute A and C's session key by $K_A = K_C = K_B \cdot \hat{e}(aP, zP)^m \cdot \hat{e}(xP, cP)^m$. Note that the attacks in case 3 and 4 are common in a large category of protocols.

4. **The Unknown Key-Share Attack**. In [1] the authors presented a so called the second source substitution attack to TAK-2 and TAK-3. Assume in some way E successfully obtains its certification $Cert_E = (I_E \| y_E \| P \| S_{CA}(I_E \| y_E \| P))$ using $y_E = \delta^2 xP$ as its public key. Note that $y_A = xP$ is A's public key and E does not know its private key $\delta^2 x$. E launch the following attack. The session key

1.	$A \rightarrow E_B, E_C$	$: T_A^1 = aP, T_A^2 = a(xP),$	$Cert_A$
$1'$.	$E \rightarrow B, C$	$: T_A^{1'} = T_A^1 = aP, T_A^{2'} = \delta^2 T_A^2 = a\delta^2 xP,$	$Cert_E$
$2'$.	$B \rightarrow E, C$	$: T_B^{1'} = bP, T_B^{2'} = b(yP),$	$Cert_B$
$3'$.	$C \rightarrow E, B$	$: T_C^{1'} = cP, T_C^{2'} = c(zP),$	$Cert_C$
2.	$E_B \rightarrow A, E_C$	$: T_B^1 = \delta T_B^{1'} = \delta bP, T_B^2 = \delta T_B^{2'} = \delta b(yP),$	$Cert_B$
3.	$E_C \rightarrow A, E_B$	$: T_C^1 = \delta T_C^{1'} = \delta cP, T_C^2 = \delta T_C^{2'} = \delta c(zP),$	$Cert_C$

Fig. 16. Attack 9: Unknown Key-Share Attack

of B and C shared with E is $K_{BE} = K_{CE} = \hat{e}(P,P)^{a\delta^2 xbycz}$ and the session key of A intended to be shared with B and C is $K_{AE_B} = K_{AE_C} = \hat{e}(P,P)^{ax\delta^2 bycz} = K_{BE} = K_{CE}$. Now E forwards A's messages encrypted under key $K_{AE_B} = K_{AE_C}$ to B and C and fools them into believing that A's messages come from E. We can add the extra exponent $\hat{e}(P,P)^{xyz}$ into the session key generation function as $K = \hat{e}(P,P)^{axbycz\hat{e}(P,P)^{xyz}}$ to prevent this attack. But we do not adopt this operation and we will analyze the reason in the following part.

5. **Perfect Forward Secrecy.** If the long-term private key x, y and z are disclosed, the session key $K = \hat{e}(P,P)^{abcxyz}$ is still secure if we ignore case 4 in the known-key attacks and if a, b and c are kept secret or are eradicated immediately after the session.
6. **The Impersonation Attack.** In some cases party C requires that it would talk to B only when A engages in the session. No normal one-round protocol can provide such security assurance. B can simply replay A's message of a previous session to cheat C. To prevent this attack, three parties

should negotiate a session related unique information, e.g., a session counter, and securely bind the negotiated unique session information with messages.

4 Security Evaluation

From the evaluation in the last section, we find that the protocol is vulnerable to the attacks of case 3 (i.e., the general concatenation attack (Attack 7)) and case 4 (Attack 8) of the known-key attack and the second source substitution attack (Attack 9). Through the analysis in [10], the general concatenation attack is feasible to all two-party AK protocols which use only (general) commutative operations on random flips in the agreed key generation function, and Attack 7 echoes this point in the tripartite (so the group) key agreement protocols. In fact Attack 8 is also feasible to this type of protocol in the tripartite case. Because the existing formal models are sensitive to these attacks (i.e., the known-key attacks of case 3 and 4), we cannot use these formal models to prove the security of the protocol. The authors in [1] used a model without the *Reveal* query, which simulates the threat of compromising session keys, to prove the security of a tweaked version of ATK-2. They concluded that ATK-2 achieves the known session key security. However, Attack 6 on ATK-2 demonstrates that the security formulation without the *Reveal* query is problematic. Moreover, proposal 1 also shows one example in which the concatenation attack is not feasible while the general concatenation attack is applicable. Note that both the concatenation attack and the general concatenation attack are feasible to ATK-1.

As Shim's ([21]) and Zhang *et al.*'s ([24]) identity-based protocols use only (general) commutative computations on random flips as well, the attacks (Attack 7 and 8) are also applicable in these protocols even the messages are signed by the senders. Hence to design a protocol that is secure against these attacks, we have to introduce some noncommutative (asymmetric) computation on random flips in the key generation functions. A simple way to introduce the noncommutative computation is to apply a hash operation on the agreed secret and messages of a session to generate the session key (so called the session key derivation mechanism). Note that there is a slight difference in the tripartite case from two-party protocols to apply a hash operation on messages, because in the tripartite case each party's received messages could be in different orders. Fortunately parties can use many ways to unify the message order, so as to obtain the same transcript. For example, parties can treat the message as multi-precision numbers and sort the messages according to the numbers' value, or can sort the messages according to the lexicographical order of the parties' identifer. After introducing the hash operation to generate the session key, e.g., $K = H_2(\hat{e}(P,P)^{axbycz}, message_1 \| message_2 \| message_3)$, the attacks feasible to the original protocol are no longer applicable. Hence we do not need to add an extra component $\hat{e}(P,P)^{xyz}$ in the exponent of the key generation function to prevent the second source substitution attack and this will save a pairing computation which is very expensive. Note that the extra hash operation does not improve the protocol's security to defend the key-compromise impersonation attack and the man-in-the-middle attack. Hence protocol TAK-1, TAK-2, TAK-3, TAK-4 in [1] and the protocol in [20] are still insecure even with the new hash operation.

5 Conclusion

To prevent the man-in-the-middle attack on Joux's one-round tripartite key agreement protocol, many authenticated protocols are proposed, but only a few of them survive the attacks in the literature. We present a certification-based protocol by fixing Shim's work. Noting that the existing formal models cannot be used to prove the security of these protocols, we heuristically evaluate the protocol' security attributes by attempting a list of attacks. Some new attacks highlight the deficiency of the existing formal models and the list can be used as a reference of attacks for other tripartite and group key agreement protocols. As have found that in theory, no existing one-round tripartite protocol achieves all the desirable security attributes directly, we suggest that the session key derivation mechanism should be used in protocols.

References

1. S. S. Al-Riyami and K. G. Paterson, "Tripartite Authenticated Key Agreement Protocols from Pairings", IMA Conference on Cryptography and Coding, Lecture Notes in Computer Science 2898, Springer-Verlag (2003), pp. 332–359. See also Cryptology ePrint Archive, Report 2002/035.
2. S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography", Advances in Cryptology – Asiacrypt'2003, Lecture Notes in Computer Science, Springer-Verlag, to appear. See also Cryptology ePrint Archive, Report 2003/126.
3. E. Bresson, O. Chevassut and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange - the dynamic case", In C. Boyd, editor, Advances in Cryptology - Proceedings of AsiaCrypt 2001, pages 290-309, 2001. Springer-Verlag - LNCS Vol. 2248.
4. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", Advances in Cryptology - Crypto'2001, Lecture Notes in Computer Science Vol. 2139, pp.213-229, Springer-Verlag, Berlin, 2001.
5. C. Boyd and A. Mathuria, "Protocols for Authentication and Key Establishment", Springer, 2003
6. M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution", CRYPTO '93, LNCS 773, pages 232-249. Springer-Verlag, Berlin, 1994.
7. D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography", Contemporary Mathematics 324, American Mathematical Society, pp. 71–90, 2003. Full version.
8. S. Blake-Wilson, D. Johnson and A. Menezes, "Key Agreement Protocols and their Security Analysis", the Sixth IMA International Conference on Cryptography and Coding, Cirencester, England, 1997.
9. Z. Chen, "Security analysis on Nalla-Reddy's ID-based tripartite authenticated key agreement protocols", Cryptology ePrint Archive, Report 2003/103.
10. Z. Cheng, R. Comley and L. Vasiu, "Revisit The Indistinguishability-Based Model of Key Agreement Protocols", manuscript, Feb. 2004.
11. R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", Eurocrypt 2001, LNCS Vol.2045
12. J. H. Cheon and D. H. Lee, "Diffie-Hellman Problems and Bilinear Maps", Cryptology ePrint Archive, Report 2002/117.
13. A. Joux, "A one-round protocol for tripartite Diffie-Hellman", Algorithm Number Theory Symposium – ANTS-IV, Lecture Notes in Computer Science 1838, Springer-Verlag (2000), pp. 385–394.
14. A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", IEEE Transactions on Information Theory 39 (1993), pp. 1639–1646.
15. A. Menezes, P. van Oorschot and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.
16. D. Nalla, "ID-based tripartite key agreement with signatures", Cryptology ePrint Archive, Report 2003/144.
17. D. Nalla and K. C. Reddy, "ID-based tripartite Authenticated Key Agreement Protocols from pairings", Cryptology ePrint Archive, Report 2003/004.
18. O. Pereira, "Modelling and Security Analysis of Authenticated Group Key Agreement Protocols", Dissertation, 2003
19. V. Shoup, "On Formal Models for Security Key Exchange", Theory of Cryptography Library, 1999.
20. K. Shim, "Efficient one-round tripartite authenticated key agreement protocol from the Weil pairing", Electronics Letters 39 (2003), pp. 208–209
21. K. Shim, "A Man-in-the-middle Attack on Nalla-Reddy's ID-based Tripartite Authenticated Key Agreement Protocol", Cryptology ePrint Archive, Report 2003/115.
22. K. Shim, "Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols", Cryptology ePrint Archive, Report 2003/122.
23. H.-M. Sun and B.-T. Hsieh, "Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings", Cryptology ePrint Archive, Report 2003/113.
24. F. Zhang, S. Liu and K. Kim, "ID-Based One Round Authenticated Tripartite Key Agreement Protocol with Pairings", Cryptology ePrint Archive, Report 2002/122.