Pairing-Based One-Round Tripartite Key Agreement Protocols

Zhaohui Cheng[⋆⋆], Luminita Vasiu and Richard Comely

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

March 15, 2004

Abstract. Since Joux published the first pairing-based one-round tripartite key agreement protocol [11], many authenticated protocols have been proposed. However most of them were broken soon or proved not to achieve some desirable security attributes. In this paper we present three protocol variants based on Shim [17] and Zhang et al.'s work [21]. As the formalized model of this kind of AK protocols is not mature, the security properties of the protocols are heuristically investigated by attempting a list of attacks described in the literature and presented as a reference, that can be used to evaluate other 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 keys vary 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 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. If in a protocol, one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key, 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)*.

Some common security attributes are generally believed to be necessary for an AK or AKC.

1. *Known session key security:* each run of the protocol should result in a unique secret session key. The compromise of one session key should not compromise other session keys (e.g. parallel sessions, previous sessions and future sessions).

[⋆⋆] Associated with Olym-Tech Inc. as an external researcher.

2. *Forward secrecy:* if 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 a party 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:* party A should not be able to be coerced into sharing a key with party C when in fact A thinks that he is sharing the key with some party B.

Apart from the security requirements, the communication and computation cost are also the critical considerations when designing key agreement protocols. There are a great number of two-party or two-party with online trust center protocols (refer to [12] for surveys) available in the literature. Formalized models of AK's and AKC's have been developed, e.g. indistinguishability-based models [4][6] and simulation-based models [9][16]. Some formalizing work has also been done for group key agreement protocols, e.g. [2][15]. In 2000 Joux presented a new efficient one-round tripartite key agreement protocol [11] by using an old mathematical tool, i.e. pairing on elliptic curves. A (symmetric) pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with group $\mathbb{G}_1$ and $\mathbb{G}_2$ of a large prime order q, which has the following properties [3]:

1. Bilinear: For all $P, Q, R, S \in \mathbb{G}_1$, $\hat{e}(P+Q, R+S) = \hat{e}(P, R)\hat{e}(P, S)\hat{e}(Q, P)\hat{e}(Q, S)$[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 $Q = 0_{\mathbb{G}_1}$.
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.

$$\begin{array}{c} A \rightarrow B, C : aP \ (1) \\ B \rightarrow A, C : bP \ (2) \\ C \rightarrow A, B : cP \ (3) \end{array}$$

Joux's One-round Tripartite Key Agreement

After 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 [3].

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

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

Assumption 1 Bilinear Diffie-Hellman Assumption *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 (BDH) by:*

$$Adv_{\mathcal{G},\mathcal{A}}(k) = Pr[\mathcal{A}(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, sP, aP, bP) = \hat{e}(P,P)^{sab} \mid$$
$$\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle \leftarrow \mathbb{G}(1^k), \text{a generator } P \leftarrow \mathbb{G}_1, s, a, b \xleftarrow{R} \mathcal{Z}_q^*].$$

For any randomized polynomial time (in k) algorithm $\mathcal{A}$, the advantage $Adv_{\mathcal{G},\mathcal{A}}(k)$ is negligible (We say that it is difficult to solve this problem or the BDH is hard).

Remark 1 *One implication of the BDH assumption is that the used bilinear pairing $\hat{e}$ cannot be extended to a 3-multilinear [5] map $\breve{e}$ with an extra property, i.e. it is easy to compute k such that $\hat{e}(P,P)^k = \breve{e}(P,P,P)$. There are some obstacles to prove the truth of the implication. First, it is not clear yet whether such a 3-multilinear map satisfying the requirements (multilinear, non-degenerate, computable) exists. Second, if such 3-multilinear maps do exist, it is still a problem whether a 3-multilinear map can be extended*[3] *from the used bilinear map. Third, even if the bilinear map can be extended, it still requires that the extending structure allows the computation to find k such that $\hat{e}(P,P)^k = \breve{e}(P,P,P)$ to be done easily. Note that the nonexistence of this special 3-multilinear map does not prove the hardness of BDH but will help to reduce some people's doubt about the correctness of the assumption. On the other hand, the existence of extendable multilinears is a potential threat to many pairing based cryptosystems. For more information of the BDH assumption, please refer to [5][10].*

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][17] and identity-based protocols such as [13][14][18][21]. Unfortunately most of them have been broken or shown not to achieve some desirable security attributes by the attacks presented in [7][18][20][19]. In fact we will show that currently there is no one-round tripartite AK protocol achieving all four security attributes.

In this paper, we strengthen Shim's certification-based protocol [17] to prevent attacks and present two variants of an identity-based protocol [21]. And we heuristically evaluate these protocols' security by attempting a list of attacks. We hope that the attack list can be used as a reference to design this type of protocols in future before the formalized model for this type of protocols becomes mature. In fact every existing protocol can be broken by one or more attacks in the list. The paper is organized as follows. In section 2, we explain two existing protocols which are the basis of our proposals. Three protocols and a list of attacks are presented in the next section. We evaluate the protocols'

[3] Boneh et al. find that at least direct extension using tensor of Weil or Tate pairing has difficulty [5].

computation and communication complexity in section 4. We draw a conclusion in the end.

2 Two Existing Protocols

2.1 A Certification-Based 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 [17]. In the protocol each party randomly chooses an integer from Z_q^* and broadcasts a message consisting of its certification and the scalar result of its public key with the chosen random integer.

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

Shim's Certification-Based Protocol

After exchanging the messages, each party computes the session key using one of the following functions.

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

But it was shown that this protocol does not achieve the key-compromise impersonate resilience attribute [20]. 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.

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

Key-Compromise Impersonate Attack

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

2.2 Identity-Based Protocols

Since Boneh and Franklin's pioneering work [3] on the identity-based encryption system based on pairing, many identity-based cryptosystems have been developed. All the systems adopt a similar setup. In these systems, there is a Key

Generation Center (KGC), which with given security arguments 1^k generates system params $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, sP, n, H_1 \rangle$. $\mathbb{G}_1, \mathbb{G}_2$ are two cyclic groups with prime order q. P is the generator of $\mathbb{G}_1$. s is randomly chosen from $\mathbf{Z}_q^*$ as the KGC's private key. $P_{pub} = sP$ is the KGC's public key and $H_1 : \{0,1\}^n \rightarrow \mathbb{G}_1^*$ is a cryptographic hash function. In the system each party can apply the hash function H_1 on any party I's identification ID_I to find an element $Q_I = H_1(ID_I) \in \mathbb{G}_1$ and party I uses its identifer ID_I as the public key and gets its private key $S_I = sQ_I$ from the KGC.

Zhang et al. designed a tripartite AK [21] by using an identity-based signature scheme to provide implicit authentication. In the protocol, party A randomly chooses two integers a and $a^{'}$ from Z_q^* and computes the scalars aP and $a^{'}P$ and the signature of these two scalars. After the computation A broadcasts a message consisting of these three elements to B and C. Party B and C perform similar operations.

$$\begin{array}{l} A \rightarrow B, C : P_A = aP, P_A^{'} = a^{'}P, T_A = H(P_A, P_A^{'})S_A + aP_A^{'} \ (1) \\ B \rightarrow A, C : P_B = bP, P_B^{'} = b^{'}P, T_B = H(P_B, P_B^{'})S_B + bP_B^{'} \ (2) \\ C \rightarrow A, B : P_C = cP, P_C^{'} = c^{'}P, T_C = H(P_C, P_C^{'})S_C + cP_C^{'} \ (3) \end{array}$$

Zhang et al.'s Protocol

After exchanging the messages, party A verifies $\hat{e}(T_B + T_C, P) = \hat{e}(H(P_B, P_B^{'})Q_B + H(P_C, P_C^{'})Q_C, P_{pub})\hat{e}(P_B, P_B^{'})\hat{e}(P_C, P_C^{'})$. If the verification is passed, A can compute eight agreed session keys: $K_A^1 = \hat{e}(P_B, P_C)^a = \hat{e}(P,P)^{abc}$, $K_A^2 = \hat{e}(P_B, P_C^{'})^a = \hat{e}(P,P)^{abc^{'}}$, etc. Party B and C take similar actions. Hence in one session of this protocol, eight session keys can be established among three parties. Shim presented a variant [18] of Zhang et al.'s protocol by reducing one element from each message but to establish only one session key in each run of the protocol.

$$\begin{array}{l} A \rightarrow B, C : U_A = aP, V_A = H(U_A)S_A + aP_{pub} \ (1) \\ B \rightarrow A, C : U_B = bP, V_B = H(U_B)S_B + bP_{pub} \ (2) \\ C \rightarrow A, B : U_C = cP, V_C = H(U_C)S_C + cP_{pub} \ (3) \end{array}$$

Shim's Identity-Based Protocol

Party A verifies $\hat{e}(P, V_B + V_C) = \hat{e}(P_{pub}, H(U_B)Q_B + H(U_C)Q_C + U_B + U_C)$ and computes $K_A = \hat{e}(U_B, U_C)^a = \hat{e}(P,P)^{abc}$. B and C perform a similar operation to compute the session key $K = \hat{e}(P,P)^{abc}$.

3 Protocol Variants

In this section, we present three variant protocols based on Shim and Zhang et al.'s work. We heuristically evaluate the protocols' security by attempting a list of attacks described in the literature. After that we compare the complexity of our proposals with the existing work from the computation and communication point of view.

3.1 A Certification-Based Protocol

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. Each party randomly chooses an integer from Z_q^* and broadcasts a message consisting of two scalars and its certifications. 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.

Protocol 1

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

After exchanging the messages, party A verifies $\hat{e}(T_B^1, yP) = \hat{e}(T_B^2, P)$ and $\hat{e}(T_C^1, zP) = \hat{e}(T_C^2, P)$. Alternatively this verification can be done by checking $\hat{e}(T_B^1, yP)\hat{e}(T_C^1, zP) = \hat{e}(T_B^2 + T_C^2, P)$ (we prove the equivalence of the two checks later). 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.

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

Security Analysis

This protocol is based on the discrete logarithm (DL) assumption and the BDH assumption[4]. 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.

Assumption 2 The Discrete Logarithm 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 $y \in Z_q^*$ 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 Z_q^*, to find integer v and b such that $\hat{e}(P, P)^v = \hat{e}(P, P)^{by}$ is hard. (We refer to this problem as the bilinear equation (BEQ) problem.)*

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

The alternative check step $\hat{e}(T_B^1, yP)\hat{e}(T_C^1, zP) = \hat{e}(T_B^2 + T_C^2, P)$ guarantees that it is hard to compute integer b, c and t such that $\hat{e}(P, P)^{by+cz} = \hat{e}(P, P)^t$ without knowing both y and z.

[4] Note that if the BDH assumption is true so is the DL assumption.

Corollary 1 *Based on the DL assumption, the problem, given* $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, yP, zP, z\rangle$ *where* y *is a random integer in* Z_q^* *and* $z \in Z_q^*$*, to find integer* b, c *and* t *such that* $\hat{e}(P,P)^t = \hat{e}(P,P)^{by+cz}$ *is hard.*

Proof: If there exists a randomized polynomial algorithm $\mathcal{A}$ to find the integer b, c and t, we can construct an algorithm $\mathcal{B}$ to solve the BEQ problem. Given the BEQ challenge $\langle q, \hat{e}, P, yP\rangle$, $\mathcal{B}$ randomly chooses an integer $z \in Z_q^*$ and passes $\langle q, \hat{e}, P, yP, zP, z\rangle$ to $\mathcal{A}$ as the challenge. When $\mathcal{B}$ gets the result b, c, t from $\mathcal{A}$, it returns $v = t - cz$ and b as the response to the BEQ challenge. Obviously, if $\mathcal{A}$ solves the problem successfully, so does $\mathcal{B}$. $\square$
Hence although obviously there are more solutions to the alternative check equation than to the equation of the original check step, Corollary 1 indicates that the intractability to solve both equations are the same. 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* $\langle q, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, P, aP, bP, cP, xP, yP, zP, axP, byP, czP\rangle$ *with* $a, b, c, x, y, z \in Z_q^*$*, to compute* $\hat{e}(P,P)^{axbycz}$ *without knowing one pair of* $\langle a, x\rangle, \langle b, y\rangle$ *or* $\langle c, z\rangle$ *is hard.*

Note that the above proofs only indicate that the used primitives in the protocol have foundations, but this does not guarantee that the protocol is secure and achieves the desirable attributes in section 1. There are some formalized security models of tripartite and group key agreement protocols used to prove the security, e.g. [1][2]. Basically these models are the extensions of Bellare and Rogway's work [4] (B-R's model). As analyzed in [8], B-R's model does not fully address the formalization problem of AK's, especially for those which use only commutative computation on random flips in the agreed key generation functions. So these extension models for tripartite AK protocols are not mature either. 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 broken by one or more of following attacks. We list these attacks known in literature even though some of them are not feasible in our proposal and 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 $a^{'}, b^{'}, c^{'} \in Z_q^*$.

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

After verification, each party computes the session key respectively. E cannot compute any of them although it knows $a^{'}, b^{'}$ and $c^{'}$.

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

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).

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

     E cannot compute the session key: $K = \hat{e}(P,P)^{axbycz}$ although it knows x and b.

   - 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).

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

     E cannot compute the session key: $K = \hat{e}(P,P)^{axbycz}$ although it knows x, b and c.

   - 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 [20] is feasible to Shim's protocol in [17], but infeasible to our protocol, because given an integer v, it is hard to find bP satisfying $\hat{e}(bP, yP) = \hat{e}(vP, P)$ (Theorem 1).

     $$\begin{array}{ll} A \rightarrow E_B, C : T_A^1 = aP, T_A^2 = a(xP), Cert_A & (1) \\ E_B \rightarrow A, C : T_B^1 = vP, T_B^2 = uP, Cert_B & (2) \\ C \rightarrow A, E_B : T_C^1 = cP, T_C^2 = c(zP), Cert_C & (3) \end{array}$$

3. **The Known-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).

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

     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.

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

- 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 [19]. But this attack is not feasible in our protocol.

- Case 3. As commented in [1][6] and [8], for those AK's which use only commutative computation on random flips in the agreed key generation functions, two attacks, the concatenation attack and the general concatenation attack, are possibly feasible. 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. Assuming 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 of session 1 respectively.

$$session_1 = \begin{cases} A \to B, C : T_A^1 = aP, T_A^2 = a(xP), Cert_A \ (1) \\ B \to A, C : T_B^1 = bP, T_B^2 = b(yP), Cert_B \ (2)\ T_B^{1'}, T_B^{2'}, Cert_B \\ C \to A, B : T_C^1 = cP, T_C^2 = c(zP), Cert_C \ (3) \end{cases}$$

$$session_2 = \begin{cases} B \to A, C : T_B^{1'} = b'P, T_B^{2'} = b'(yP), Cert_B \ (1) \\ A \to B, C : T_A^{1'} = a'P, T_A^{2'} = a'(xP), Cert_A \ (2)\ T_A^1, T_A^2, Cert_A \\ C \to A, B : T_C^{1'} = c'P, T_C^{2'} = c'(zP), Cert_C \ (3)\ T_C^1, T_A^2, Cert_C \end{cases}$$

After exchanging messages, A's 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 in session 2 $K_B^2 = \hat{e}(T_A^1, T_C^1)^{b'y} = \hat{e}(P, P)^{axb'ycz}$. 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 somehow. E randomly chooses integer $m \in Z_q^*$ and tempts B's broadcast message with the new one as shown in $(2')$.

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

Obviously the transcript of B for the session is different from the ones of A and C. Moreover obviously this session have been accepted by B before long-term key y is exposed. Hence we can deem B is in a different session from the one that A and C had engaged in. At least in the existing indistinguishability-based model of AK, this is the case. Now E reveals A's session key $K_A = \hat{e}(P,P)^{axbycz}\hat{e}(P,P)^{axmycz}$ by some means. 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 tripartite case, the attack is feasible to TAK-2, TAK-3 [1] and for two-party case examples can be found in [8]. We simply present the attack to TAK-2 as follows.

$$\begin{array}{lll} A \rightarrow B, C & : T_A^1 = aP & (1) \\ B \rightarrow A, C & : T_B^1 = bP & (2) \\ E_B \rightarrow A, C & : T_B^{1'} = T_B^1 + mP, & (2') \\ C \rightarrow A, B & : T_C^1 = cP & (3) \end{array}$$

After revealing B's session key $K_B = \hat{e}(aP, zP)^b \hat{e}(xP, cP)^b \hat{e}(aP, cP)^y$, E can compute A and C's session key by $K_A = K_C = K_B \cdot \hat{e}(aP, zP)^m \cdot \hat{e}(xP, cP)^m$.

4. **The Unknown Key-Share Attack**. In [1] the authors presented 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.

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

The session key 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 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 a, b and c are kept secret or are eradicated immediately after the session when we ignore the case 4 of known-key attacks.

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.

Security Evaluation
From the above evaluation, we find that the protocol is vulnerable to the attacks in case 3 (the general concatenation attack) and case 4 of the known-key attack and the second source substitution attack. Through the analysis in [8], the general concatenation attack is feasible to all AK protocols which use only commutative operation on random flips in the agreed key generation function. In fact case 4 attack is also feasible to this kind of protocol in the tripartite case. As Shim and Zhang et al.'s identity-based protocols also use only commutative computation, the attacks 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 nonlinear computation on random flips in the key generation functions. A simple way to introduce the nonlinear computation is to apply a hash operation on the agreed secret and messages of a session to generate the session key (so called session key derivation mechanism). Note that there is a slight difference in the tripartite case from the two party protocol 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 [17] are still insecure even with the new hash operation.

3.2 Two Identity-Based Protocols

Based on Shim and Zhang et al.'s work we present two variants of the identity-based tripartite protocol.

Protocol 2

$$\begin{array}{l} A \to B, C : T_A^1 = aP, T_A^2 = aQ_A, T_A^3 = aS_A \ (1) \\ B \to A, C : T_B^1 = bP, T_B^2 = bQ_B, T_B^3 = bS_B \ (2) \\ C \to A, B : T_C^1 = cP, T_C^2 = cQ_C, T_C^3 = cS_C \ (3) \end{array}$$

After exchanging the messages, party A verifies $\hat{e}(T_B^1, Q_B) = \hat{e}(P, T_B^2)$, $\hat{e}(T_B^2, P_{pub})$ $= \hat{e}(T_B^3, P)$, $\hat{e}(T_C^1, Q_C) = \hat{e}(P, T_C^2)$ and $\hat{e}(T_C^2, P_{pub}) = \hat{e}(T_C^3, P)$. An alternative way is to check $\hat{e}(T_B^3 + T_C^3 + T_B^2 + T_C^2, P) = \hat{e}(T_B^2 + T_C^2, P_{pub})\hat{e}(Q_B, T_B^1)\hat{e}(Q_C, T_C^1)$. And if the check step is passed, A computes the session key $K = \hat{e}(T_B^1, T_C^1)^a =$ $\hat{e}(P, P)^{abc}$. Party B and C perform similar operations. Note that adversary E can tempt the messages by doing addition or deduction on each element of messages of the same party.

Protocol 3

$$\begin{array}{l} A \to B, C : T_A^1 = aP_{pub}, T_A^2 = H(T_A^1)S_A \ (1) \\ B \to A, C : T_B^1 = bP_{pub}, T_B^2 = H(T_B^1)S_B \ (2) \\ C \to A, B : T_C^1 = cP_{pub}, T_C^2 = H(T_C^1)S_C \ (3) \end{array}$$

After exchanging the messages, party A verifies $\hat{e}(P_{pub}, H(T_B^1)Q_B) = \hat{e}(P, T_B^2)$ and $\hat{e}(P_{pub}, H(T_C^1)Q_C) = \hat{e}(P, T_C^2)$. An alternative way is to check $\hat{e}(P, T_B^2 + T_C^2) = \hat{e}(P_{pub}, H(T_B^1)Q_B + H(T_C^1)Q_C)$. And if the check step is passed, A computes the session key $K = \hat{e}(T_B^1, T_C^1)^a = \hat{e}(P, P)^{abcs^2}$. Party B and C perform the similar operations. If an adversary E wants to generate message (2) to pass the verification, it should be able to solve equation $\hat{e}(P, Q_B)^{sn_1} = \hat{e}(P, P)^{n_2}$ or $\hat{e}(P, Q_B)^{sn_1} = \hat{e}(P, Q_B)^{n_3}$ to compute n_1 and n_2 or n_3, which is hard according to Theorem 1. Based on the same analysis, Shim and Zhang et al's identity-based protocols can be simplified too.

Note that the above two protocols also suffer from the known-key attacks addressed in the above section. Hence a hash operation is necessary to introduce a nonlinear computation in the key generation functions. Moreover, because the agreed key generated are purely based on the random flips, the random flips of each messages are extremely important to the protocols' security and should be protected as the long-term private keys. If the random flips of a message generated by a party are leaked, adversary E can impersonate the party. If the random flips of two messages of two parties are exposed, adversary E can launch the man-in-the-middle attack. For example, if E knows the random flips of two messages: msg_B^1 generated by B and msg_C^1 generated by C (two messages are not necessary from the same session), in another session, it can launch the following attack and compute the session keys of A, B and C in session 2.

$$\begin{array}{l} A \to E_B, E_C : msg_A^2 \ (1) \ msg_B^1, msg_C^1 : A \leftarrow E \\ B \to E_A, E_C : msg_B^2 \ (2) \ msg_A^2, msg_C^1 : B \leftarrow E \\ C \to E_A, E_B : msg_C^2 \ (3) \ msg_A^2, msg_B^1 : C \leftarrow E \end{array}$$

For this reason, the security notion of this kind of protocols based on signature is essentially different from the security notions of signature, authenticated encryption and signcryption. To counter this attack, one possible way is to introduce session counter or time related information to messages securely. This can be done relatively easily in protocol 3.

4 The Protocol Evaluation

Apart from the security consideration, when designing key agreement protocols the computation and communication cost are also of great concern. The pairing-based protocols need to take three different basic operations, i.e. pairing, scalar and exponent computation. Pairing computation is the most expensive one among the three and normally scalar points of an elliptic curve is faster than the exponent operation in the corresponding multiplicative number group. As we have addressed, the hash operation is always necessary to design secure protocols at least for use as the secret derivation function. For the protocols using certifications to provide implicit authentication, the verification operation of CA's signature in certifications is needed. The existing bilinear pairings, i.e. Weil pairing and Tate pairing, map the torsion points[5] of an elliptic curve to the elements of a subgroup of a multiplicative group and it is normally recommended to use multiplicative groups with orders of at least 1024 bit size to guarantee the hardness of the discrete logarithm problem in the groups. The pairing and scalar operations are in torsion point groups with orders decided by the embedding degree of the elliptic curve and the order of the multiplicative group. If we treat the signature in a certification as a point in the torsion group, then a message of a protocol consists of the sender's identifier and some points of an elliptic curve. We do not consider the overhead of compressing and recovering the points. The computation and communication complexity of our proposals, Shim and Zhang et al.'s protocols are listed in Table 1. We do not try to compare the complexity

	Pairing	Scalar	Exponent	Hash$^{(iii)}$	SigVer^{iv}	Bandwidth
Proposal 1	4	2	1	1	2 $(2/n)^{(i)}$	5 points + 1 identifer
Proposal 2	5	3	1	1	0	3 points + 1 identifer
Proposal 3	3	4	1	3+1	0	2 points + 1 identifer
Shim	3	5	1	3+1	0	2 points + 1 identifer
Zhang	5 $(8/8)^{(ii)}$	6 (6/8)	1 (8/8)	3+1 (3/8+1)	0	3 points + 1 identifer

^{i} : the signature of a certification needs to be verified once.
^{ii} : one run of Zhang et al.'s protocol can establish eight session keys.
^{iii} : an extra hash operation is introduced in the key generation function.
^{iv} : all protocols need to check whether public keys or certifications are revoked.

Table 1. The Complexity of Protocols

of certification-based protocols with identity-based protocols because these two categories of protocols have their own different application scenarios. For the four identity-based protocols, if the party groups are relatively fixed and agreed keys are established regularly in the fixed groups, Zhang et al.'s protocol is a

[5] In fact one element of the Tate pairing is a normal subset of the used torsion group.

good choice. On the other hand, proposal 3 and Shim's protocol are favorite ones.

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 and two identity-based proposals based on Zhang et al.'s protocol. By attempting a list of attacks known in the literature which can be used as a reference of attacks for other protocols, we heuristically evaluate the protocols' security attributes. As have found that no existing one-round tripartite achieves all the desirable attributes, 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. 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.
3. 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.
4. M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," CRYPTO '93, LNCS 773, pages 232-249. Springer-Verlag, Berlin, 1994.
5. D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography," Contemporary Mathematics 324, American Mathematical Society, pp. 71–90, 2003. Full version.
6. 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.
7. Z. Chen, "Security analysis on Nalla-Reddy's ID-based tripartite authenticated key agreement protocols," Cryptology ePrint Archive, Report 2003/103.
8. Z. Cheng and R. Comely, "Revisit The Indistinguishability-Based Model of Key Agreement Protocols," manuscript, Feb. 2004.
9. R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," Eurocrypt 2001, LNCS Vol.2045
10. J. H. Cheon and D. H. Lee, "Diffie-Hellman Problems and Bilinear Maps," Cryptology ePrint Archive, Report 2002/117.
11. 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.

12. A. Menezes, P. van Oorschot and S. Vanstone, “Handbook of Applied Cryptography,” CRC Press, 1996.
13. D. Nalla, “ID-based tripartite key agreement with signatures,” Cryptology ePrint Archive, Report 2003/144.
14. D. Nalla and K. C. Reddy, “ID-based tripartite Authenticated Key Agreement Protocols from pairings,” Cryptology ePrint Archive, Report 2003/004.
15. O. Pereira, “Modelling and Security Analysis of Authenticated Group Key Agreement Protocols,” Dissertation, 2003
16. V. Shoup, “On Formal Models for Security Key Exchange,” Theory of Cryptography Library, 1999.
17. K. Shim, “Efficient one-round tripartite authenticated key agreement protocol from the Weil pairing,” Electronics Letters 39 (2003), pp. 208–209
18. 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.
19. K. Shim, “Cryptanalysis of Al-Riyami-Paterson’s Authenticated Three Party Key Agreement Protocols,” Cryptology ePrint Archive, Report 2003/122.
20. H.-M. Sun and B.-T. Hsieh, “Security Analysis of Shim’s Authenticated Key Agreement Protocols from Pairings,” Cryptology ePrint Archive, Report 2003/113.
21. F. Zhang, S. Liu and K. Kim, “ID-Based One Round Authenticated Tripartite Key Agreement Protocol with Pairings,” Cryptology ePrint Archive, Report 2002/122.