Tripartite Authenticated Key Agreement Protocols from Pairings

Sattam S. Al-Riyami and Kenneth G. Paterson
Information Security Group,
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
satt@sultanateoman.com kenny.paterson@rhul.ac.uk

Abstract

Joux's protocol [29] is a one round, tripartite key agreement protocol that is more bandwidth-efficient than any previous three-party key agreement protocol. But it is insecure, suffering from a simple man-in-the-middle attack. This paper shows how to make Joux's protocol secure, presenting several tripartite, authenticated key agreement protocols that still require only one round of communication. A pass-optimal authenticated and key confirmed tripartite protocol that generalises the station-to-station protocol is also presented. The security properties of the new protocols are studied using provable security methods and heuristic approaches. Applications for the protocols are also discussed.

Areas: Secure protocols; key agreement; authentication; pairings.

1 Introduction

Asymmetric key agreement protocols are multi-party protocols in which entities exchange public information allowing them to create a common secret key that is known only to those entities and which cannot be predetermined by any party. This secret key, commonly called a *session key*, can then be used to create a confidential or integrity-protected communications channel amongst the entities. Beginning with the famous Diffie-Hellman protocol [19], a huge number of two-party key agreement protocols have been proposed (see [10] and [37, Chapter 12.6] for surveys). This reflects the fundamental nature of key exchange as a cryptographic primitive.

The situation where three or more parties share a secret key is often called *conference keying*. The three-party (or tripartite) case is of most practical importance not only because it is the most common size for electronic conferences but because it can be used to provide a range of services for two parties communicating. For example, a third party can be added to chair, or referee a conversation for ad hoc auditing, data recovery or escrow purposes.

One of the most exciting developments in recent years in the area of key agreement is Joux's tripartite key agreement protocol [29]. This protocol makes use of *pairings* on elliptic curves and requires each entity to transmit only a single broadcast message. This should be contrasted with the obvious extension of the Diffie-Hellman protocol to three parties, which requires two broadcasts per entity. However, just like the raw Diffie-Hellman, protocol, Joux's protocol is unauthenticated and suffers from man-in-the-middle attacks. See Section 3 for a description of Joux's protocol and an overview of pairings on elliptic curves.

1.1 Our Contribution

In this paper we show how to transform Joux's protocol into a secure tripartite protocol that still requires only a single broadcast per entity. In fact, we present four different one round, tripartite, authenticated key agreement (TAK) protocols in Section 4. These protocols all have the same protocol messages, but have different methods for calculating session keys from those messages.

Note that we do *not* make the naive claim that the possibility of generating four different session keys from one protocol run means that our protocol is four times as efficient as other protocols – there are far better ways of deriving multiple session keys than this! Rather we present four protocols and analyze their different security properties. Our one round protocols build on Joux's protocol and draw on ideas from the Unified Model [4], MTI [35] and MQV [32] protocols.

In Section 5, we consider proofs of security for our protocols. Our proofs use an adaptation of the Bellare-Rogaway model [7] to the public key setting. Our model is similar to that given by Blake-Wilson, Johnson and Menezes in [9] (though our model extends to the three-party situation, and our extension is different to that presented in the symmetric-key setting in [8]). Our proofs show that three of the TAK protocols are secure provided that the Bi-linear Diffie-Hellman Problem (BDHP) is hard. This computational problem forms the basis for the security of the identity-based encryption scheme of [11] – see Section 3 for more details on the BDHP. We do not provide a proof of security for our fourth protocol. But we note that the MQV protocol, on which our fourth TAK protocol is modelled, has never been proven secure in any model, though it has been widely adopted for standardisation [1, 3, 27, 28]. In view of the incomplete analysis currently available through provable approaches, we choose to supplement our proofs of security with ad hoc analysis in Section 6. This allows us to consider attacks on our protocols not included in the security model. In Section 7, we look at the scenario where one of the three parties is off-line. The protocol we give for this situation can be applied to key escrow with an off-line escrow agent.

In our penultimate section, Section 8, we examine pairing-based authenticated key agreement with key confirmation in the non-broadcast setting. The main point we make in that section is that a properly authenticated and key confirmed protocol based on pairings can be no more efficient (in terms of protocol passes) than the obvious extension of the station-to-station protocol [20] to three parties. Thus the apparent efficiency of Joux's protocol is lost when it is made secure and when an appropriate measure of efficiency is used.

The final section, Section 9, contains some conclusions and ideas for future work.

1.2 Related Work

As we have already noted, our work builds on that of Joux [29], and our one round protocols draw upon the Unified Model [4], MTI [35] and MQV [32] protocols. The tripartite AKC non-broadcast protocol presented builds on the STS [20] protocol. Our security model is inspired by Blake-Wilson, Johnson and Menezes' extension [9] of the Bellare-Rogaway model [7]. More recent work on models for secure protocols (in particular key agreement protocols) can be found in [6, 14, 16, 17, 42]. Work pertinent to the efficient implementation of our protocols can be found in [5, 21, 22]. Other work, reflecting the recent explosion of interest in using pairings to construct cryptographic schemes, can be found in [11, 12, 18, 24, 25, 31, 39, 41, 43].

2 Protocol Goals and Attributes

Here we discuss the various desirable attributes and goals that one may wish a key agreement protocol to possess.

2.1 Extensional Security Goals

An *extensional* goal [13, 40] for a protocol is defined to be a design goal that is independent of the protocol details. Here we list two desirable and widely-agreed extensional goals for key agreement protocols. Further discussion can be found in [37, Chapter 12]. The first goal we try to achieve is *implicit key authentication*. This goal, if met, assures an entity that only the intended other entities *can* compute a particular key. This level of authentication results in what is known as an *authenticated key agreement* (AK) protocol. If each entity is also assured that the intended other entities actually *have* computed the key, then the resulting protocol is called an *authenticated key agreement with confirmation* (AKC) protocol. Another goal is to provide a *good key*. This goal states that the key is selected uniformly at random from the key space, so that no adversary has an information-theoretic advantage when mounting a guessing strategy to determine the key.

2.2 Security Attributes

A number of desirable security attributes have been identified for key agreement protocols [9, 10, 32] and we borrow our definitions from these sources. Depending on the application scenario, these attributes can be vital in excluding realistic attacks.

Known session key security A protocol is *known session key secure* if it still achieves its goal in the face of an adversary who has learned some previous session keys.

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

No key-compromise impersonation Suppose A's long-term private key is disclosed. Then of course an adversary can impersonate A in any protocol in which A is identified by this key. We say that a protocol resists *key-compromise impersonation* when this loss does not enable an adversary to impersonate *other* entities as well and obtain the secret key.

No unknown key-share 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 or integrity [30].

No key control It should not be possible for any of the participants (or an adversary) to force the session key to a preselected value or predict the value of the session key. (See [38] for a discussion of how protocol participants can partially force the values of keys to particular values and how to prevent this using commitments).

2.3 Further Attributes

Two important computational attributes are low *computation overhead* and the ability to perform *precomputation*. It may be desirable that one or more entities (perhaps with limited computational environments) should perform less computation than the others.

It is an advantage when a protocol has low *communication overhead*, which means that only a small amount of data is transmitted. Designing protocols with a minimal number of *passes* and *rounds* is always desirable. The number of passes is the total number of messages exchanged in the protocol. A *round* consists of all the messages that can be sent and received in parallel within one time unit. A *broadcast* message is a message that is sent to every party in a protocol. These notions of communication complexity are each more or less appropriate depending on the particular network architecture that is in use. For example, wireless systems operate exclusively in broadcast mode, so that every packet is simultaneously available to all nodes. Then the number of rounds and broadcasts is a more natural way of measuring a protocol's communication complexity. On the other hand, the Internet Protocol running over a public network like the internet usually makes use of point-to-point communications, and then the number of passes is the right measure. As we shall see with Joux's protocol, communication advantages that a protocol apparently possesses can disappear when one either considers a different network architecture or more stringent security requirements.

If the messages transmitted in a protocol are *independent* of each other, in the sense that they can be sent and received in any order, then the protocol is *message independent.* A protocol is *role symmetric* when messages transmitted and computations performed by all the entities have the same structure. In the context of public key cryptography, *short-term public keys* (or *values*) are generally only used once to establish a session, and are sometimes called *ephemeral keys.* On the other hand, *long-term public keys* are static keys used primarily to authenticate the protocol participants.

2.4 Attacks

An attack occurs when the intended goals of a protocol are not met or the desired security attributes do not hold. A *passive* attack occurs when an adversary can prevent the protocol from accomplishing its goals by simply observing the protocol runs. Conversely, an *active* attack is one in which the adversary may delete, inject, alter or redirect messages, by interleaving multiple instantiations of the same protocol and the like. The formal model of security that we introduce in Section 5 is powerful enough to capture many kinds of active attack.

3 Key Agreement via Pairings on Elliptic Curves

Here, we briefly review some background on pairings on elliptic curves, and then present Joux's protocol [29].

3.1 Pairings

We use the same notation as in [12]. We let $\mathbb{G}_1$ be an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same order q. We assume the existence of an efficiently computable bi-linear map $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Typically, $\mathbb{G}_1$ will be a subgroup of the group of points on an elliptic curve over a finite field, $\mathbb{G}_2$ will be a subgroup of the multiplicative group of a related finite field and the map $\hat{e}$ will be derived from either the Weil or Tate pairing on the elliptic curve. We also assume that an element $P \in \mathbb{G}_1$ satisfying $\hat{e}(P,P) \neq 1_{\mathbb{G}_2}$ is known. By $\hat{e}$ being bi-linear, we mean that for $Q, W, Z \in \mathbb{G}_1$, both

$$\hat{e}(Q, W+Z) = \hat{e}(Q,W) \cdot \hat{e}(Q,Z) \quad \text{and} \quad \hat{e}(Q+W, Z) = \hat{e}(Q,Z) \cdot \hat{e}(W,Z)$$

hold.

When $a \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_1$, we write aQ for Q added to itself a times, also called scalar multiplication of Q by a. As a consequence of bi-linearity, we have that, for any $Q, W \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$:

$$\hat{e}(aQ, bW) = \hat{e}(Q,W)^{ab} = \hat{e}(abQ, W) = \ldots$$

a fact that will be used repeatedly in the sequel without comment.

We refer to [5, 12, 11, 21, 22] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security. We simply assume in what follows that suitable groups $\mathbb{G}_1$ and $\mathbb{G}_2$, a map $\hat{e}$ and an element $P \in \mathbb{G}_1$ have been chosen, and that elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ can be represented by bit strings of the appropriate lengths. We note that the computations that need to be carried out by entities in our protocols will always involve pairing computations, and that the complexity of these will generally dominate any other calculations. However, with recent advances in efficient implementation of pairings, [5, 22], the complexity of a pairing computation is now of a similar order to that of elliptic curve point multiplication.

3.2 Joux's Protocol

In [29], Joux introduced a very simple and elegant one-round protocol in which the secret session key for three parties could be created using just one broadcast per entity. In Joux's protocol, $a, b, c \in \mathbb{Z}_q^*$ are selected uniformly at random by A, B and C respectively. The ordering of protocol messages is unimportant and any of the three entities can initiate the protocol. In all the protocol messages "Sends to" is denoted by "$\rightarrow$".

Protocol messages:
$A \rightarrow B, C$: aP (1)
$B \rightarrow A, C$: bP (2)
$C \rightarrow A, B$: cP (3)
Protocol 1 (Joux's one round protocol).

Protocol description: Once the communication is over, A computes $K_A = \hat{e}(bP, cP)^a$, B computes $K_B = \hat{e}(aP, cP)^b$ and C computes $K_C = \hat{e}(aP, bP)^c$. By bi-linearity of $\hat{e}$, these are all equal to $K_{ABC} = \hat{e}(P, P)^{abc}$. This can serve as the secret key shared by A, B and C.

Although not explicitly mentioned in [29], the success of this protocol in achieving its aim of agreeing a good key for the three entities in the face of passive adversaries is related to the hardness of the Bi-linear Diffie-Hellman problem (BDHP) for the pair of groups $\mathbb{G}_1$, $\mathbb{G}_2$. This problem, formalised in [12], can be stated as:

Given P, aP, bP, cP with a, b and c chosen uniformly at random from $\mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc}$.

More properly, the session key should be derived by applying a suitable key derivation function to the quantity $\hat{e}(P, P)^{abc}$. For otherwise, an attacker might be able to get partial information about session keys even if the BDHP is hard. This motivates study of hard bits for the BDHP [23].

It is known that the BDHP is no harder than the computational Diffie-Hellman problems in either $\mathbb{G}_1$ or $\mathbb{G}_2$. The reverse relationship is not known. Typically then, one chooses parameters so that $\mathbb{G}_1$, a subgroup of the group of points on an elliptic curve, has around 2^{160} elements and so that $\mathbb{G}_2$ is a subgroup of $\mathbb{F}_r$ where r has roughly 1024 bits. See [12, 21] for details.

Unfortunately just like the unauthenticated two-party Diffie-Hellman protocol, Joux's protocol is subject to a simple man-in-the-middle attack. The attack is easily realised, and allows an adversary E to masquerade as any entity to any other entity in the network. We leave the construction of such an attack as a straightforward exercise.

4 One Round Tripartite Authenticated Key Agreement Protocols

The advantage of Joux's tripartite protocol over any previous tripartite key agreement protocol is that a session key can be established in just one round. The disadvantage is that this key is not authenticated, and this allows a man-in-the-middle attack.

In this section, we develop protocols which also need just one round, but which provide a key which is implicitly authenticated to all entities. Our AK protocols are generalisations of the standardised [1, 3, 27] Unified Model protocol [4], the MTI family of protocols [35] and the MQV protocol [32] to the setting of pairings. In fact we present a single protocol with four different methods for deriving a session key.

In order to provide session key authentication, some form of authenticated long-term private/public key pairs are needed. As with the other protocols, a certification authority (CA) is used in the initial set-up stage to provide certificates which bind users' identities to long-term keys. The certificate for entity A will be of the form:

$$\mathrm{Cert}_A = (\mathcal{I}_A \| \mu_A \| P \| \mathcal{S}_{\mathrm{CA}}(\mathcal{I}_A \| \mu_A \| P)).$$

Here $\mathcal{I}_A$ denotes the identity string of A, $\|$ denotes the concatenation of data items, and $\mathcal{S}_{\mathrm{CA}}$ denotes the CA's signature. Entity A's long-term public key is $\mu_A = xP$, where $x \in \mathbb{Z}_q^*$ is the long-term private key of A. Element P is a public value and is included in order to specify which element is used to construct μ_A and the short-term public values. Similarly Cert_B and Cert_C are the certificates for entities B and C, with $\mu_B = yP$ and $\mu_C = zP$ as their long-term public keys. Certificates could contain further information, such as validity periods.

As usual, in the protocol below, short-term keys $a, b, c \in \mathbb{Z}_q^*$ are selected uniformly at random by A, B and C respectively.

Protocol messages:
$A \to B, C$: $aP \| \mathrm{Cert}_A$ (1)
$B \to A, C$: $bP \| \mathrm{Cert}_B$ (2)
$C \to A, B$: $cP \| \mathrm{Cert}_C$ (3)
Protocol 2 (Tripartite Authenticated Key Agreement (TAK) protocol).

Protocol description: An entity A broadcasting to B and C, sends his fresh short-term public value aP along with a certificate Cert_A containing his long-term public key. Corresponding values and certificates are broadcast by B and C to A, C and A, B respectively. Notice that the protocol messages are just the same as in Joux's protocol, except for the addition of certificates. Each party verifies the authenticity of the two certificates he receives. If any check fails, the protocol should be aborted. When no check fails, one of four possible session keys described below should be computed. Below, H denotes a suitable hash function.

TAK key generation:

1. **Type 1 (TAK-1):**
   The keys computed by the entities are:
   $K_A = H(\hat{e}(bP, cP)^a \| \hat{e}(yP, zP)^x)$,
   $K_B = H(\hat{e}(aP, cP)^b \| \hat{e}(xP, zP)^y)$,
   $K_C = H(\hat{e}(aP, bP)^c \| \hat{e}(xP, yP)^z)$.
   By bi-linearity, all parties now share the session key $K_{ABC} = H(\hat{e}(P, P)^{abc} \| \hat{e}(P, P)^{xyz})$.

2. **Type 2 (TAK-2):**
   The keys computed by the entities are:
   $K_A = \hat{e}(bP, zP)^a \cdot \hat{e}(yP, cP)^a \cdot \hat{e}(bP, cP)^x$,
   $K_B = \hat{e}(aP, zP)^b \cdot \hat{e}(xP, cP)^b \cdot \hat{e}(aP, cP)^y$,
   $K_C = \hat{e}(aP, yP)^c \cdot \hat{e}(xP, bP)^c \cdot \hat{e}(aP, bP)^z$.
   The session key is $K_{ABC} = \hat{e}(P, P)^{(ab)z+(ac)y+(bc)x}$.

3. **Type 3 (TAK-3):**
   The keys computed by the entities are:

$K_A = \hat{e}(yP, cP)^x \cdot \hat{e}(bP, zP)^x \cdot \hat{e}(yP, zP)^a$,
$K_B = \hat{e}(aP, zP)^y \cdot \hat{e}(xP, cP)^y \cdot \hat{e}(xP, zP)^b$,
$K_C = \hat{e}(aP, yP)^z \cdot \hat{e}(xP, bP)^z \cdot \hat{e}(xP, yP)^c$.
The session key is $K_{ABC} = \hat{e}(P, P)^{(xy)c+(xz)b+(yz)a}$.

4. **Type 4 (TAK-4):**
The keys computed by the entities are:
$K_A = \hat{e}(bP + H(bP\|yP)yP, cP + H(cP\|zP)zP)^{a+H(aP\|xP)x}$,
$K_B = \hat{e}(aP + H(aP\|xP)xP, cP + H(cP\|zP)zP)^{b+H(bP\|yP)y}$,
$K_C = \hat{e}(aP + H(aP\|xP)xP, bP + H(bP\|yP)yP)^{c+H(cP\|zP)z}$.
The session key is
$K_{ABC} = \hat{e}(P, P)^{(a+H(aP\|xP)x)(b+H(bP\|yP)y)(c+H(cP\|zP)z)}$.

Notes:

- Joux's protocol [29] and our protocols are all vulnerable 'small subgroup' attacks and variants of them observed by Lim and Lee [33]. To protect against this, a verification algorithm should be applied by each entity to ensure that their received elements are actually in $\mathbb{G}_1$. This is fairly cheap to do when $\mathbb{G}_1$ is an elliptic curve group.
- In all four cases, key generation is role symmetric and each entity uses knowledge of both short-term and long-term keys to produce a unique shared secret key. No party has control over the resulting session key K_{ABC} and if any one of a, b, c is chosen uniformly at random, then K_{ABC} is a random element of $\mathbb{G}_2$. Of course delays in receipt of messages from the last entity should not be tolerated by the other entities, because after the last entity sees all the other participants' messages he is capable of fixing a small number of bits in the final shared secret key. See [38] for more details.
- Since all four keys are created after transmitting the same protocol messages, the communication overhead of each protocol version is identical. However, TAK-2 and TAK-3 key generation require more computation compared to TAK-1: in the former, each entity must make three pairing calculations, with the latter, just two. Better still, TAK-4 requires only a single pairing computation per entity. Protocols TAK-1 and TAK-3 can exploit pre-computation if entities know in advance with whom they will be sharing a key. In TAK-1, all entities can pre-compute the term $\hat{e}(P, P)^{xyz}$. With TAK-3, A can pre-compute $\hat{e}(P, P)^{ayz}$, with similar pre-computations for B and C. However, these terms cannot be re-used because fresh a, b and c should be used in each new protocol session.

Rationale for the TAK keys' algebraic forms:

- Protocol TAK-1 is analogous to the Unified Model protocol, whilst protocols TAK-2 and TAK-3 have their roots in the MTI/A0 protocol. The Unified Model protcol is similar to the MTI protocols but utilises a hash function H with concatenation to combine various components, instead of a multiplication. The MTI-like variant of TAK-1, in which the agreed key is $K_{ABC} = \hat{e}(P, P)^{abc+xyz}$, suffers from a severe form of key-compromise impersonation attack. This attack does not require the adversary to learn a long-term private key. Rather, the adversary only needs to obtain a session key and one of the short-term secret values used in a protocol run to mount the attack. We do not apply hash functions in TAK-2 and TAK-3 because this would still not prevent serious and practical attacks presented in Section 6. In any case, it would be prudent to derive session (and MAC keys for key confirmation if desired) by applying a hash function to each K_{ABC}. This would prevent problems arising from the possible existence of relatively easy bits in the BDHP. Using a hash function in TAK-1 has the additional benefit that it allows a security proof to be given (assuming H is replaced by a random oracle) – see Section 5.

- Protocol TAK-4 is modelled on the MQV protocol but avoids that protocol's unknown key-share weakness [30] by using a hash function H to combine long-term and short-term private keys. Here H's output is assumed to be onto $\mathbb{Z}_q^*$. Note that the protocol resulting from omission of this hash function produces the key $K_{ABC} = \hat{e}(P,P)^{(a+x)(b+y)(c+z)}$. However, this version of the protocol suffers from an unknown key-share weakness similar to that presented for the MQV protocol in [30], wherein the attacker *does* know the private key corresponding to his registered public key. As a consequence, this attack cannot be prevented by requiring the adversary to provide a proof of possession for her private key as part of the registration process. See Section 6.3 for further discussion of unknown key-share attacks.

- Other MTI-like protocols can be produced if A, B and C broadcast the ordered pairs $ayP\|azP$, $bxP\|bzP$ and $cxP\|cyP$ respectively (along with the appropriate certificates). This protocol can be used to produce the MTI/C0-like shared secret key $K_{ABC} = \hat{e}(P,P)^{abc}$, which for example A calculates by $K_{ABC} = \hat{e}(bxP, cxP)^{ax^{-2}}$. It can also be used to produce the MTI/B0-like shared secret key $K_{ABC} = \hat{e}(P,P)^{ab+bc+ca}$, which A can calculate by $K_{ABC} = \hat{e}(bxP, cxP)^{x^{-2}} \cdot \hat{e}(P, bxP)^{ax^{-1}} \cdot \hat{e}(P, cxP)^{ax^{-1}}$. Although these protocols produce a key with forward security, we do not consider them further here because they require significantly higher bandwidth and do not offer a security advantage over our other protocols. For example, both protocols suffer from key compromise impersonation attacks and the MTI/C0 analogue is also vulnerable to known session key attacks.

Our TAK protocols prevent simple man-in-the-middle attacks. This is because the long-term private keys are involved in the computation of each K_{ABC} prevents an adversary not in possession of a long-term private key from obtaining these session keys. However, other forms of active attack can still occur. We consider such attacks on a case-by-case basis in Section 6, after considering proofs of security.

5 Security Proofs

In this section, we introduce a security model for TAK protocols, and consider in detail the security of protocol TAK-1 in this model. We also sketch proofs of security for minor variants of protocols TAK-2 and TAK-3.

Our model is similar to those introduced in [7] and [9], with some simplifications that are possible because of the one round nature of our TAK protocols. In particular, we avoid the use of matching conversations (and session IDs introduced in later work [6]). Rather than fully describing our model, we highlight the differences to previous work.

Let k be a security parameter. As usual, we assume a set $\mathcal{I} = \{1, 2, \ldots, T_1(k)\}$ of protocol participants, where $T_1(k)$ is a polynomial bound in k on the number of participants. We will also use $A, B, C, \ldots$ to refer to protocol participants, while E is reserved for our adversary (who is not a participant). Each participant A is modelled by an oracle Π_A^s, which the adversary E can query at will. Here, s is a session number, that determines which random tape Π_A^s will use. In previous work, oracles were of the form $\Pi_{i,j}^s$ and modelled messages sent from participant i to participant j in session s. We remove this dependence on receiving parties; in all our protocols, all messages are broadcast.

Oracles exist in one of several possible states `Accept`, `Reject`, or `*`. In our protocols, an oracle accepts only after receipt of two properly formulated messages (containing different certificates to its own) and the transmission of two messages, not necessarily in that order (and with the received messages possibly originating from the adversary and not the oracles identified in the certificates in those messages). When an oracle accepts, we assume it accepts holding a session key K that is k bits in length. We also assume there is a key generation process $\mathcal{G}$ which produces a description of groups $\mathbb{G}_1$

and $\mathbb{G}_2$ and the map $\hat{e}$, assigns random tapes and oracles as necessary, distributes long-term private keys to participants, and prepares certificated long-term public keys. (Thus our model assumes a perfect certification process and does not capture attacks based on registration weaknesses like those described in Section 6.3). As usual, the benign adversary is defined to be one that simply passes messages to and fro between participants.

Adversary E is assumed to have complete control over the network, and we allow E to make three kinds of query to an oracle Π_A^s: `Send`, `Reveal` and `Corrupt`. These have the usual meanings, as per [9]: `Send` allows the adversary to send a message of her choice to Π_A^s and to record the response, or to induce Π_A^s to initiate a protocol run with participants of E's choosing. `Reveal` reveals the session key (if any) held by Π_A^s, while `Corrupt` produces the long-term private key of the oracle. Notice that when making a `Send` query, E does not need to specify the intended recipients of the oracle's reply. This is because, in our protocols, the oracle's replies are independent of these recipients anyway. In fact E can relay these messages to any party of her choosing.

In our TAK protocols, each party sends the same message to two other participants and receives two messages from those participants. In our model, we say that three oracles Π_A^s, Π_B^t and Π_C^u have *participated in a matched session* if they have received messages exclusively generated by one another (via the adversary). In other words, the two messages that Π_A^s has received are those generated by Π_B^t and Π_C^u, and these are the two oracles and sessions which received the single message from Π_A^s, and likewise for the other two participants.

In addition to the above queries, we allow E to make one further `Test` query of one of the oracles Π_A^s at any point during her attack. This oracle must be *fresh*: it must have accepted, not have been subject to a reveal query, be uncorrupted, not have participated in a matched session with any revealed oracle, and not have received messages containing the certificate of a corrupted oracle. The reply to this `Test` query is either the session key K held by the oracle, or a random k-bit string, the choice depending on a fair coin toss. The adversary's advantage, denoted $advantage^E(k)$, is the probability that E can distinguish K from the random string. Notice that we remove the unnatural restriction in earlier models [7, 9] that this `Test` query be the adversary's last interaction with the model.

As usual, a function $\epsilon(k)$ is negligible, if for every $c > 0$ there exists a $k_c > 0$ such that for all $k > k_c$, $\epsilon(k) > k^{-c}$.

We say that a protocol is a secure TAK protocol if:

1. In the presence of the benign adversary, and when oracles participate in a matched session, all the oracles always accept holding the same session key, which is distributed randomly and uniformly on $\{0, 1\}^k$.

2. If uncorrupted oracles Π_A^s, Π_B^t and Π_C^u participate in a matched session, then all three oracles accept and hold the same session key, which is again uniformly distributed on $\{0, 1\}^k$.

3. $advantage^E(k)$ is negligible.

The first two conditions here ensure that a secure TAK protocol does indeed distribute a key of the correct form. The last condition says that no adversary can obtain any information about the session key held by a fresh oracle.

With the description of our security model in hand, we now state:

Theorem 1 *Protocol TAK-1 is a secure TAK protocol, assuming that the adversary makes no* `Reveal` *queries, that the Bi-linear Diffie-Hellman problem (for the pair of groups $\mathbb{G}_1$ and $\mathbb{G}_2$) is hard and provided that H is a random oracle.*

We prove this theorem in Appendix A. Here we comment on the significance of this proof.

We emphasise that our proof of security does not allow the adversary to make `Reveal` queries of oracles. This means that our proof does not capture known session-key attacks. In fact protocol TAK-1 is vulnerable to a simple attack of this type. We describe the attack in full in Appendix B. The attack only works on TAK-1 because of the symmetry of the short-term components, and attacks of this type do not appear to apply to TAK-2, TAK-3 or TAK-4. The attack is analogous to known session key attacks well-understood for other protocols (see the comments following [9, Theorem 11] for an example).

Results analogous to Theorem 1 can be proved for variants of protocols TAK-2 and TAK-3 in which the various key components are concatenated and hashed, rather than multiplied. For example, we can show this for the TAK-2-like protocol TAK-2′ in which the key is equal to

$$H(\hat{e}(P,P)^{(ab)z} \| \hat{e}(P,P)^{(ac)y} \| \hat{e}(P,P)^{(bc)x}).$$

We still require for our proofs that the adversary makes no `Reveal` queries, but we conjecture that this restriction is actually unnecessary for these two variants. The proofs are very similar to the proof of Theorem 1 given in Appendix A. For example, for protocol TAK-2′, we simply need to replace one of oracle Π_B's replies to `Send` queries in F's simulation with the value $x_B P$, one of oracle Π_C's replies to `Send` queries with the value $x_C P$, and B and C's private values with random values known to F. We must also adjust F's parsing of H queries slightly. We leave the details to the reader.

In Section 8.1, we consider the confirmed versions of our four protocols. These are obtained in the usual way, by adding three confirmation messages (which are piggy-backed on other messages in a non-broadcast environment), one for each protocol participant. The confirmation messages use encryptions and signatures; these could be replaced with MACs using keys derived from the short term values exchanged during the protocol run. We expect that the techniques used to prove the security of Protocol 2 of [9] could be adapted to prove that the described confirmed versions of TAK-1, TAK-2′ and TAK-3′ are indeed secure AKC protocols.

Finally, no security proof has yet appeared for the MQV protocol (on which TAK-4 is based) although the MQV protocol is widely believed to be secure and has been standardised [1, 3, 27, 28]. As we shall see in the next section, current security models do not handle all the reasonable attack types and so need to be augmented by *ad hoc* analysis. We include TAK-4 because it is a very practical and heuristically secure protocol.

6 Heuristic Security Analysis of TAK Protocols

We present a variety of attacks on our TAK protocols that are not captured by the security models of the previous section. These are mostly inspired by earlier attacks on the two-party MTI protocols. Following this analysis, we summarise the security attributes of our TAK protocols in Table 1. In this section, we use E_A to indicate that the adversary E is impersonating A in sending or receiving messages intended for or originating from A. Similarly, $E_{B,C}$ denotes an adversary impersonating both B and C.

6.1 Key-Compromise Impersonation Attacks on TAK-1 and TAK-2

We present a key-compromise impersonation attack that occurs when E can obtain the long-term private key of one of the entities. Suppose E has obtained x, A's long-term private key. Then E can calculate $\hat{e}(P,P)^{xyz}$ using x and public data in B and C's certificates. Knowledge of this value now allows E to impersonate any entity engaging in a TAK-1 protocol run with A (and not just A).

TAK-2 is vulnerable to a related attack, which also occurs whenever E obtains x. In the attack E impersonates B and C to A, by choosing values a, b and broadcasting bP, cP to A. With knowledge of x, b and c, $E_{B,C}$ has enough components to create a TAK-2 key. Note that this attack requires the impersonation of two entities to A and not one as in the previous attack. These kinds of attacks do not appear to apply to TAK-3 or TAK-4 because of the combinations of long-term and short-term key components in computing K_{ABC} used in those protocols.

One way to defeat key-compromise attacks on the protocols in general is for each entity to use his long-term private key as an ECDSA signature key to sign short-term keys [2]. These signatures should be broadcast in the protocol along with the short-term keys and certificates. This prevents an attacker from 'cutting and pasting' certificates onto values δP for identities other than the one whose long-term key has been compromised. However, this clearly increases the bandwidth and computational requirements of the protocols. It is also generally accepted as bad practice to use key pairs for more than one purpose.

6.2 Forward Security Weakness in TAK-2 and TAK-3

A protocol is not forward secure if the compromise of the long-term private keys of one or more entities also allows an adversary to obtain session keys previously established between honest entities. With TAK-1, this weakness is not present, no matter how many long-term private keys are available to the adversary. Indeed if all three keys are available to her, then extracting the session key K_{ABC} from an old session can be shown to be equivalent to solving the BDHP. Thus TAK-1 is perfect forward secure. The same is true of TAK-4, because the key K_{ABC} agreed in that case also includes the component $\hat{e}(P,P)^{abc}$. However, it is straightforward to see that if an adversary obtains either two long-term private keys in TAK-3, or all three long-term private keys in TAK-2, then she has the ability to obtain old session keys (assuming she keeps a record of the public values aP, bP, cP). Thus TAK-2 and TAK-3 are not forward secure. Both protocols can be made perfectly forward secure, at extra computational cost, by using the key $K_{ABC} \cdot \hat{e}(P,P)^{abc}$ in place of the key K_{ABC} in each case.

6.3 Unknown Key-Share Attacks

A basic source substitution attack applies to all our TAK protocols. It utilises a potential registration weakness for public keys to create fraudulent certificates [36].

Adversary E registers A's public key μ_A as her own, creating $\text{Cert}_E = (\mathcal{I}_E \| \mu_A \| P \| \mathcal{S}_{\text{CA}}(\mathcal{I}_E \| \mu_A \| P))$. Then she intercepts A's message $aP \| \text{Cert}_A$ and replaces Cert_A with her own certificate Cert_E. Note that E registered the A's long-term public key xP as her own without knowing the value of x. Therefore she cannot learn the key K_{ABC}. However, B and C are fooled into thinking they have agreed a key with E, when in fact they have agreed a key with A. They will interpret any subsequent encrypted messages emanating from A as coming from E. This basic attack could be eliminated if the CA does not allow two entities to register the same long-term public key. However, this solution may not scale well to large or distributed systems. A better solution is discussed after highlighting another type of source substitution attack.

A second source substitution attack can be applied to TAK-2 and TAK-3. Even if the CA does the previous check, the adversary can still obtain a Cert_E from the CA which contains a component μ_E which is a multiple of μ_A. The adversary can then alter the short-term keys in subsequent protocol messages by appropriate multiples. As with the last attack the adversary does not create the shared key. Rather, the attack gives her the ability to fool two participants B, C into believing messages came from her rather then from the third (honest) participant A. This attack is described in Appendix C.

The adversary in our attacks *does not* know her long term private key. Therefore, all these source substitution attacks are easily prevented if the CA insists that each registering party provides a proof of possession of his private key when registering a public key. This can be achieved using a variety of methods. For example, one might use zero-knowledge techniques or regard the pair (x, xP) as an

ECDSA signature key pair and have the registering party sign a message of the CA's choice using this key pair. As an alternative, identities can be included in the key derivation function to prevent unknown key-share attacks.

6.4 Insider Attacks

Certain new kinds of insider attack must be considered when we move to tripartite protocols. For example, an insider A might be able to fool B into believing that they have participated in a protocol run with C, when in fact C has not been active. An active A can do this easily in our TAK protocols, simply by choosing C's value cP and injecting it into the network along with C's certificate and stopping B's message from reaching C. This kind of attack can have serious consequences for tripartite protocols: for example, if C acts as an on-line escrow agent, then B believes a shared key has been properly escrowed with C when in fact it has not. On the other hand, when C acts as an off-line escrow agent, as in the protocol we describe in Section 7, this insider attack is only significant if C is providing an auditing function. This lack of auditability is actually beneficial if protocol participants want to have a deniability property, as in protocols like IKE, IKEv2 and JFK [26].

Attacks of this type can be prevented in a number of ways. Adding a confirmation phase, as we do in Section 8, prevents them. An alternative requires a complete protocol re-design, but maintains a one round broadcast protocol: simply use the long-term keys to sign ephemeral values (rather than combining them with short-term keys as in our TAK protocols) and agree the key $\hat{e}(P,P)^{abc}$. This approach requires each participant to maintain a log of all ephemeral values used, or to use synchronized clocks and time-stamps, to prevent an attacker simply replaying an old message. This requirement along with the need to create and verify signatures for each protocol run makes this solution somewhat unwieldy.

6.5 Security Summary

In addition to the above attacks, we note that TAK-3 is vulnerable to a triangle attack of the type introduced by Burmester [15]. We do not include it here, as it is somewhat theoretical in nature.

Table 1 compares the security attributes that we believe our protocols TAK-1, TAK-2, TAK-3 and TAK-4 to possess. We have also included a comparison with the 'raw' Joux protocol.

Based on this table and the analysis in Section 5, we recommend the use of protocol TAK-4 or protocol TAK-3 along with pre-computation (in the event that the use of a hash function is not desirable). If perfect forward security is also required, then TAK-3 can be modified as described in Section 6.2. Protocol TAK-4 has the additional benefit of being the most computationally efficient of all our protocols. Of course, robust certification is needed for all of our TAK protocols in order to avoid unknown key-share attacks.

7 Tripartite Protocols with One Off-line Party

As we mentioned in the introduction, there is an application of tripartite key exchange protocols to the two-party case when one of the parties acts as an escrow agent. It may be more convenient that this agent be off-line, meaning that he receives messages but is not required to send any messages. In this section, we adapt our earlier protocols to this situation.

The protocol below is a modified version of TAK-2. The modification has the added benefit of eliminating the second of our source substitution attacks, and the key-compromise impersonation attack which were applied to TAK-2. We assume that C is the off-line party and that C's certificate Cert_C is pre-distributed, or is readily available to A and B.

	Joux	TAK-1	TAK-2	TAK-3	TAK-4
Implicit key authentication	No	Yes	Yes	Yes	Yes
Known session key secure	No	No	Yes	Yes	Yes
Perfect forward secure	n/a	Yes	No$^{(i)}$	No$^{(ii)}$	Yes
Key-compromise impersonation secure	n/a	No	No$^{(iii)}$	Yes	Yes
Unknown key-share secure	No	Yes$^{(iv)}$	Yes$^{(v)}$	Yes$^{(v)}$	Yes$^{(iv)}$

Table 1: Comparison of security goals and attributes for one round tripartite key agreement protocols.

(i) Not forward secure when a fatal compromise occurs on all three long-term private keys, but still forward secure for a compromise of two or less such keys.

(ii) Not forward secure when two long-term private keys are compromised, but still forward secure if only one is compromised.

(iii) Not key-compromise impersonation secure when both participants are impersonated, but is secure if an attempt is made to only impersonate one participant.

(iv) If the CA checks that public keys are only registered once, and if inconvenient use (v).

(v) If the CA verifies that each user is in possession of the long-term private key corresponding to his public key.

Protocol messages:
$A \to B, C$: $aP \| \mathrm{Cert}_A$ (1)
$B \to A, C$: $bP \| \mathrm{Cert}_B$ (2)
Protocol 6 (Off-line TAK protocol).

Protocol description: The protocol is as in TAK-2, but without the participation of C. Entities A and B use C's long-term public key zP in place of his short-term public value cP when calculating the session key. Thus, the session key computations carried out by the entities is $K_{ABC} = \hat{e}(P,P)^{(ab)z+(a)yz+(b)xz}$. Note that C can compute the key when required.

Both this protocol and the similarly constructed variant of TAK-4 producing the session key $K_{ABC} = \hat{e}(P,P)^{(a+H(aP\|xP)x)(b+H(bP\|yP)y)(z)}$ exhibit excellent security properties: they are resistant to all the previous attacks except the simple source substitution attack which is easily thwarted via robust registration procedures. They are also forward secure, obviously except when private key z is compromised. Here z can be viewed as an independent master key, which can be regularly updated along with the corresponding certificate.

8 Non-Broadcast, Tripartite AKC Protocols

Up to this point, we have considered protocols that are efficient in the broadcast setting: they have all required the transmission of one broadcast message per participant. As we mentioned in the introduction, the number of broadcasts is not always the most relevant measure of a protocol's use of communications bandwidth. A good example is the basic broadcast Joux protocol, which offers neither authentication nor confirmation of keys and requires six passes in a non-broadcast network. In this section we introduce a pairing-based tripartite key agreement protocol that also requires six passes, but that offers both key confirmation and key authentication, i.e. is an AKC protocol. We show that *any* such protocol requires at least six passes. We then compare our protocol to a tripartite version of the station-to-station protocol [20].

8.1 A Six Pass Pairing-Based AKC Protocol

Our notation in describing our pairing-based tripartite, authenticated key agreement with key confirmation (TAKC) protocol is largely as before. Additionally, $\mathcal{S}_A(\sigma)$ denotes A's signature on the string σ. We assume now that the CA's certificate Cert_A contains A's signature verification key. Also $E_K(\sigma)$ denotes encryption of string σ using a symmetric algorithm and key K, and χ denotes the string $aP\|bP\|cP$.

Protocol messages:

$A \to B$:	$aP\|\mathrm{Cert}_A$	(1)
$B \to C$:	$aP\|\mathrm{Cert}_A\|bP\|\mathrm{Cert}_B$	(2)
$C \to A$:	$bP\|\mathrm{Cert}_B\|cP\|\mathrm{Cert}_C\|E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A\|\mathcal{I}_B\|\chi))$	(3)
$A \to B$:	$cP\|\mathrm{Cert}_C\|E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A\|\mathcal{I}_B\|\chi))\|E_{K_{ABC}}(\mathcal{S}_A(\mathcal{I}_B\|\mathcal{I}_C\|\chi))$	(4)
$B \to C$:	$E_{K_{ABC}}(\mathcal{S}_A(\mathcal{I}_B\|\mathcal{I}_C\|\chi))\|E_{K_{ABC}}(\mathcal{S}_B(\mathcal{I}_A\|\mathcal{I}_C\|\chi))$	(5)
$B \to A$:	$E_{K_{ABC}}(\mathcal{S}_B(\mathcal{I}_A\|\mathcal{I}_C\|\chi))$	(6)

Protocol 3 (TAKC protocol from pairings).

Protocol description: Entity A initiates the protocol execution with message (1). After receiving message (2), entity C is able to calculate the session key $K_{ABC} = \hat{e}(P,P)^{abc}$. The same session key is calculated after receiving messages (3) and (4), for A and B respectively. Messages (3) and onwards contain signatures on the ephemeral values and identities in the particular protocol run. This provides key authenticity. These signatures are transmitted in encrypted form using the session key K_{ABC} and this provides key confirmation. The confirmations from C to B, A to C and B to A are piggy-backed and forwarded by the intermediate party in messages (3), (4) and (5) respectively. More properly, encryptions should use a key derived from K_{ABC} rather than K_{ABC} itself. The symmetric encryptions can be replaced by appending MACs to the signatures with the usual safeguards.

If the expected recipients' identities were not included in the signatures this protocol would be vulnerable to an extension of an attack due to Lowe [34]. This attack exploits an authentication error and allows a limited form of unknown key-share attack. To perform it, we assume adversary D has control of the network. The attack is as follows.

1. D_C intercepts message (2), then D forwards (2) replacing Cert_B with Cert_D to C as if it originated from D. Thus, C assumes he is sharing a key with A and D.
2. D_A intercepts message (3) en route to A. Now D_C forward this message replacing Cert_D with Cert_B to A.
3. Entity A receives (3) and continues with the protocol, sending message (4).
4. D blocks messages (5) and (6), so C and A assume an incomplete protocol run has occurred and terminate the protocol. However, on receipt of message (4), entity B already thinks he has completed a successful protocol run with A and C, whilst C might not even know B exists.

As usual in an unknown key-share attack, D cannot compute the shared key. The attack is limited because A and C end up with an aborted protocol run (rather than believing they have shared a key with B). The attack is defeated in our protocol because the inclusion of identities in signatures causes the protocol to terminate after message (3), when A realises that an illegal run has occurred.

We claim that no tripartite AKC protocol can use fewer than six passes. This can be reasoned as follows. Each of the three entities must receive two ephemeral keys to construct K_{ABC}, so a total of six ephemeral values must be received. But the first pass can contain only one ephemeral key (the one known by the sender in that pass), while subsequent passes can contain two. Thus a minimum of four passes are needed to distribute all the ephemeral values to all of the parties. So only after at least four passes is the last party (entity B in our protocol) capable of creating the key. This last party needs another two passes to provide key confirmation to the other two entities. So at least six passes are needed in total. Notice that this argument holds whether the network supports broadcasts or not.

8.2 A Six Pass Diffie-Hellman Based AKC Protocol

The station-to-station (STS) protocol is a three pass, two-party AKC protocol designed by Diffie, van Oorschot and Wiener [20] to defeat man-in-the-middle attacks. Here we extend the protocol to three parties and six passes, a pass-optimal protocol by the argument above.

An appropriate prime p and generator g mod p are selected. In Protocol 4 below, $a, b, c \in \mathbb{Z}_p^*$ are randomly generated ephemeral values and χ denotes the concatenation $g^a \| g^b \| g^c$. As before, $E_{K_{ABC}}(\cdot)$ donates symmetric encryption under session key K_{ABC}, and as before $\mathcal{S}_A(\cdot)$ denotes A's signature. Again, we assume that authentic versions of signature keys are available to the three participants. We have omitted modulo p operations for simplicity of presentation.

Sequence of protocol messages:

$A \to B$:	$g^a \| \mathrm{Cert}_A$	(1)
$B \to C$:	$g^a \| \mathrm{Cert}_A \| g^b \| \mathrm{Cert}_B \| g^{ab}$	(2)
$C \to A$:	$g^b \| \mathrm{Cert}_B \| g^c \| \mathrm{Cert}_C \| g^{bc} \| E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A \| \mathcal{I}_B \| \chi))$	(3)
$A \to B$:	$g^c \| \mathrm{Cert}_C \| g^{ac} \| E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A \| \mathcal{I}_B \| \chi)) \| E_{K_{ABC}}(\mathcal{S}_A(\mathcal{I}_B \| \mathcal{I}_C \| \chi))$	(4)
$B \to C$:	$E_{K_{ABC}}(\mathcal{S}_A(\mathcal{I}_B \| \mathcal{I}_C \| \chi)) \| E_{K_{ABC}}(\mathcal{S}_B(\mathcal{I}_A \| \mathcal{I}_C \| \chi))$	(5)
$B \to A$:	$E_{K_{ABC}}(\mathcal{S}_B(\mathcal{I}_A \| \mathcal{I}_C \| \chi))$	(6)

Protocol 4 (TAKC protocol generalising STS protocol).

Protocol description: The protocol is similar to protocol 3 in operation, with an additional extra computation done before steps (2)(3) and (4). The shared session key is $K_{ABC} = g^{abc} \bmod p$.

8.3 Analysis of AKC Protocols

Two immediate conclusions from our analysis can be drawn. Firstly, we have given a pairing-based, tripartite AKC protocol using just the same number of passes as are needed in Joux's protocol (but with the penalty of introducing message dependencies). Secondly, this AKC version of Joux's protocol is no more efficient in terms of passes than a 3-party version of the STS protocol! Thus when one considers confirmed protocols in a non-broadcast environment, the apparent advantage that Joux's protocol enjoys disappears. Of course, there is a two round broadcast version of the TAKC protocol (requiring 6 broadcasts and 12 passes). Both of our six pass AKC protocols can be done in 5 rounds in a broadcast environment.

9 Conclusions

We have taken Joux's one round tripartite key agreement protocol and used it to construct one round TAK protocols. We have considered security proofs and heuristic security analysis of our protocols, as well as an off-line version of our protocols. We have preserved the innate communications efficiency of Joux's protocol while enhancing its security functionality. We also considered tripartite variants of the STS protocol, suited to non-broadcast networks, showing that in this case, pairing-based protocols can offer no communication advantage over more traditional Diffie-Hellman style protocols.

Future work should consider the security of our protocols in more robust models, capturing a larger set of realistic attacks. Constructing multi-party key agreement protocols using our TAK protocols as a primitive might result in bandwidth-efficient protocols. Finally, it would be interesting to see if the methods of [14] could be emulated in the setting of pairings to produce TAK protocols secure in the standard model.

10 Acknowledgement

We would like to thank Chris Mitchell and Steven Galbraith for their comments on a draft of this paper.

References

[1] American National Standards Institute – ANSI X9.42. Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography, 2001.

[2] American National Standards Institute – ANSI X9.62. Public key cryptography for the financial services industry: the elliptic curve digital signature algorithm (ECDSA), 1998.

[3] American National Standards Institute – ANSI X9.63. Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography, 2001.

[4] R. Ankney, D. Johnson, and M. Matyas. The Unified Model – contribution to X9F1, October 1995.

[5] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, LNCS. Springer-Verlag, 2002.

[6] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *LNCS*, pages 139–155. Springer-Verlag, 2000.

[7] M. Bellare and P. Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.

[8] M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing STOC*, pages 57–66. ACM, 1995.

[9] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the sixth IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer-Verlag, 1997.

[10] S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman key agreement protocols. In S. Tavares and H. Meijer, editors, *5th Annual Workshop on Selected Areas in Cryptography (SAC '98)*, volume 1556 of *LNCS*, pages 339–361. Springer-Verlag, 1998.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[12] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, to appear. `http://www.crypto.stanford.edu/~dabo/abstracts/ibe.html`, full version of [11].

[13] C. Boyd. Towards extensional goals in authentication protocols. In *Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. `http://www.citeseer.nj.nec.com/boyd97towards.html/`.

[14] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In L.R. Knudsen, editor, *Advances in Cryptology - Eurocrypt 2002*, volume 2332 of *LNCS*, pages 321–336. Springer-Verlag, 2002.

[15] M. Burmester. On the risk of opening distributed keys. In Y. G. Desmedt, editor, *Advances in Cryptology CRYPTO ' 94, 14th Annual International Cryptology Conference*, volume 839 of *LNCS*, pages 308–317. Springer-Verlag, 1994.

[16] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *LNCS*, pages 453–474. Springer-Verlag, 2001.

[17] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Application of Cryptographic Techniques*, volume 2332 of *LNCS*, pages 337–351. Springer-Verlag, 2002.

[18] J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *LNCS*, pages 18–30. Springer-Verlag, 2002.

[19] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[20] W. Diffie, P.C. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[21] S.D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Proceedings of AsiaCrypt 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *LNCS*, pages 495–513. Springer-Verlag, 2001.

[22] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory 5th International Symposium, ANTS-V*, volume 2369 of *LNCS*, pages 324–337. Springer-Verlag, 2002.

[23] S.D. Galbraith, H.J. Hopkins, and I.E. Shparlinski. Secure Bilinear Diffie-Hellman bits. Cryptology ePrint Archive, Report 2002/155, 2002. `http://eprint.iacr.org/`.

[24] C. Gentry and A. Silverberg. Heirarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, 2002.

[25] F. Hess. Exponent group signature schemes and efficient identity based signature schemes based on pairings. Cryptology ePrint Archive, Report 2002/012, 2002. `http://eprint.iacr.org/`.

[26] P. Hoffman. Features of proposed successors to IKE. Internet Draft, draft-ietf-ipsec-soi-features-01.txt, 2002.

[27] IEEE P1363. Standard specifications for public key cryptography, 2000. `http://grouper.ieee.org/groups/1363/index.html`.

[28] ISO/IEC 15946-3. Information technology – security techniques – cryptographic techniques based on elliptic curves – part 3: Key establishment, awaiting publication.

[29] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of Algorithmic Number Theory Symposium – ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.

[30] B. Kaliski, Jr. An unknown key-share attack on the MQV key agreement protocol. *ACM Trans. on Information and Systems Security*, 4(3):275–288, 2001.

[31] M. Kim and K. Kim. A new identification scheme based on the bi-linear Diffie-Hellmann problem. In *Proc. ACISP 2002*, volume 2384 of *LNCS*, pages 362–378. Springer-Verlag, 2002.

[32] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. Technical Report CORR 98-05, Department of C & O, University of Waterloo, 1998. To appear in Designs, Codes and Cryptography.

[33] C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *LNCS*, pages 249–263. Springer-Verlag, 1997.

[34] G. Lowe. Some new attacks upon security protocols. In *PCSFW: Proceedings of The 9th Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, 1996.

[35] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *Trans. IECE of Japan*, E69:99–106, 1986.

[36] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentications. *2nd Workshop on Selected Areas in Cryptography (SAC'95)*, pages 22–32, May 1995.

[37] A. Menezes, P.C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.

[38] C. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, 34:980–981, 1998.

[39] K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[40] A. Roscoe. Intensional specifications of security protocols. In *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 28–38. IEEE Computer Society Press, 1996.

[41] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Sympoium on Cryptography and Information Security*, Okinawa, Japan, January 2000.

[42] V. Shoup. On formal models for secure key exchange. IBM Technical Report RZ 3120, 1999. `http://shoup.net/papers`.

[43] N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

Appendix A: Proof of Theorem 1

We provide a proof of Theorem 1.

Conditions 1 and 2: Given the assumption that H is a random oracle, these conditions follow directly from the protocol description.

Condition 3: Suppose that $advantage^E(k) = n(k)$ is non-negligible. We show how to construct from E an algorithm F which solves the BDHP with non-negligible probability. We describe F's operation. F's input is a description of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and the map $\hat{e}$, a non-identity element $P \in G_1$, and a triple of elements $x_AP, x_BP, x_CP \in \mathbb{G}_1$ with x_A, x_B, x_C chosen randomly from $\mathbb{Z}_q^*$. F's task is to compute and output the value $g^{x_Ax_Bx_C}$ where $g = \hat{e}(P, P)$.

F operates as follows. F chooses a triple $A, B, C \in \mathcal{I}$ uniformly at random. F simulates the running of the key generation algorithm $\mathcal{G}$, choosing all participants' long-term private keys randomly itself, and computing the corresponding long-term public keys and certificates, but with the exception of Π_A, Π_B and Π_C's keys. As public values for Π_A, Π_B and Π_C, F chooses the values x_AP, x_BP, x_CP respectively. Then F starts adversary E.

In E's attack, F will simulate all the oracles Π_i, $i \in \mathcal{I}$. So F must answer all the oracle queries that E makes. F answers E's queries as follows. F simply answers E's distinct H queries at random, maintaining a table of queries and responses as he proceeds. Note that we do not allow our adversary to make `Reveal` queries, so F does not need to answer any queries of this type. F answers any `Corrupt` queries by revealing the long-term private key, except for `Corrupt` queries on Π_A, Π_B or Π_C. In the event of such queries, F aborts. F replies to `Send` queries in the usual way, with correctly formulated responses $a_{i,s}P||Cert_{\Pi_i}$ for all oracles Π_i^s, where $a_{i,s} \in_R \mathbb{Z}_q^*$.

Finally, we consider how F responds to the `Test` query on oracle Π_i. F generates a random bit $b \in \{0, 1\}$. If $b = 0$, F should respond with the key held by Π_i, while if $b = 1$, F should respond with a random k-bit value. Now F is capable of answering the `Test` query correctly except when $b = 0$ and the tested oracle is an instance of Π_A, Π_B or Π_C. In this last case, F's response should be of the form $H(Q\|g^{x_Ax_Bx_C})$ where $Q \in \mathbb{G}_2$, involving the invocation of the random oracle. This use of H should be consistent with previous and future uses, but of course F does not know $g^{x_Ax_Bx_C}$, so cannot properly simulate the oracle in this case. Instead, F responds with a random k-bit value. This potentially introduces an imperfection into F's simulation, but we will argue below that this has no effect on success probabilities.

The final stage is as follows. Let $T_2(k)$ denote a polynomial bound on the number of H queries answered by F in the course of E's attack. F picks a value ℓ uniformly at random from $\{1, \ldots, T_2(k)\}$. Now F parses the ℓ-th H query into the form $Q||W$ where $Q, W \in \mathbb{G}_2$. If this is not possible, F aborts. If it is, then F outputs W as its guess for the value $g^{x_Ax_Bx_C}$ and stops.

Now we must evaluate the probability that F's output is correct. Notice that E's view of F's simulation of the oracles is indistinguishable from E's view in a real attack provided that F is not forced to abort when asked a `Corrupt` query and that E does not detect that F's simulation of the random oracle was deficient when responding to the `Test` query – more on these situations later. Now E picks some accepted oracle $\Pi_{i_1}^s$ for its `Test` query in a real attack. Suppose that $\Pi_{i_1}^s$ has received two messages containing certificates of oracles Π_{i_2} and Π_{i_3}. The session key held by oracle $\Pi_{i_1}^s$ will be of the form $H(Q||g^{x_{i_1}x_{i_2}x_{i_3}})$ where $Q \in \mathbb{G}_2$ and x_{i_j} is the long-term private key of Π_{i_j}, $1 \leq j \leq 3$. Since by definition, the oracles Π_{i_j} are uncorrupted, and E does not ask any `Reveal` queries, if E is to succeed in distinguishing this session key from a random string with non-negligible probability $n(k)$, then E must have queried H on an input of the form $Q||g^{x_{i_1}x_{i_2}x_{i_3}}$ at some point in its attack with some non-negligible probability $n'(k)$. The probability that this event occurs in F's simulation of the oracles is therefore also $n'(k)$. Since F outputs a random query from the list of $T_2(k)$ queries, has randomly distributed public keys x_AP, x_BP, x_CP amongst the $T_1(k)$ participants, and is only deemed successful if he does not abort and his output is of the form $g^{x_A x_B x_C}$, we see that F is successful with probability better than:

$$\frac{n'(k)}{T_1(k)^3 T_2(k).}$$

However, this is still non-negligible in k.

We claim that our argument is not affected by the imperfection introduced into F's simulation when E asks a `Corrupt` query that F cannot answer: to be successful, E must ask a `Test` query of an uncorrupted but accepted oracle which has received messages containing certificates of two further uncorrupted oracles. This means that for E to be successful, at least three distinct, uncorrupted oracles must remain. So F, having made a random choice of where to place public keys x_AP, x_BP, x_CP, has at least a $\frac{1}{T_1(k)^3}$ chance of not facing an unanswerable `Corrupt` query whenever E is successful. This factor is already taken into account in our analysis.

One problem remains: what effect on E's behaviour does F's deviation in giving a random response to the "difficult" `Test` query have? In particular, what effect does it have on success probabilities? E's behaviour can only differ from that in a true attack run if E detects any inconsistency in F's simulation of the random oracle. In turn, this can only happen if at some point in the attack, E queries H on an input of the form $Q \| g^{x_A x_B x_C}$. For otherwise, no inconsistency arises. At this point, E's behaviour becomes undefined. (In this situation, E might guess that F's response to the `Test` query is a random key ($b = 1$) rather than the "correct" key ($b = 0$). But we must also consider the possibility that E simply might not terminate.) So we assume that F simply aborts his simulation whenever E's attack lasts longer than some polynomial bound $T_3(k)$ on the length of a normal attack. Notice that H has been queried on an input of the form $Q \| g^{x_A x_B x_C}$ at some point in F's simulation, and that up until this point, E's view is indistinguishable from that in a real attack. Thus the number of H queries made by E will still be bounded by $T_2(k)$ up to this point, and an input of the required type will occur amongst these. So F's usual guessing strategy will be successful with probability $1/T_2(k)$ even when E's behaviour is affected F's inability to correctly respond to the `Test` query. Since this is the same success probability for guessing in the situation where everything proceeds normally, it is now easy to see that F's overall success probability is still at least

$$\frac{n'(k)}{T_1(k)^3 T_2(k).}$$

Appendix B: Known Session Key Attack on TAK-1

We present a known session key attack on TAK-1 that makes use of session interleaving and message reflection. In the attack, E interleaves three sessions and reflects messages originating from A back to A in the different protocol runs. The result is that the session keys agreed in the three runs are identical, so E, upon revealing one of them, gets keys for two subsequent sessions as well. In what follows, E_A indicates that the E is impersonating A in sending or receiving messages intended for or originating from A. Similarly, $E_{B,C}$ denotes an adversary impersonating both B and C.

A is convinced to initiate three sessions with E:

Session $\alpha : A \to E_{B,C} :\quad aP \| \mathrm{Cert}_A \quad (1^\alpha)$
Session $\beta : A \to E_{B,C} :\quad a'P \| \mathrm{Cert}_A \quad (1^\beta)$
Session $\gamma : A \to E_{B,C} :\quad a''P \| \mathrm{Cert}_A \quad (1^\gamma)$

E reflects and replays pretending to be B and C, to complete session α:

$E_B \to A :\quad a'P \| \mathrm{Cert}_B \quad (2^\alpha)$
$E_C \to A :\quad a''P \| \mathrm{Cert}_C \quad (3^\alpha)$

Similarly the second session is completed by $E_{B,C}$ sending $a''P\|\mathrm{Cert}_B$ (2^β) and $aP\|\mathrm{Cert}_C$ (3^β) to A. In the third parallel session she sends $aP\|\mathrm{Cert}_B$ (2^γ) and $a'P\|\mathrm{Cert}_C$ (3^γ) to A.

E now obtains the first session key $H(\hat{e}(P,P)^{aa'a''}\|\hat{e}(P,P)^{xyz})$. She then knows the keys for the next two sessions, as these are identical to this first session key.

Appendix C: Source Substitution Attacks on TAK-2 and TAK-3

We now present in detail the second source substitution attack on TAK-2, and then sketch the attack on TAK-3.

1. A sends $aP\|\mathrm{Cert}_A$ to $E_{B,C}$.
2. E computes $\mu_E = \delta^2\mu_A = \delta^2 xP$ and registers μ_E as part of her Cert_E.
3. E initiates a run of protocol TAK-2 by sending $\delta aP\|\mathrm{Cert}_E$ to B, C.
4. B sends $bP\|\mathrm{Cert}_B$ to E, C; C sends $cP\|\mathrm{Cert}_C$ to E, B.
5. B and C (following the protocol) compute $K_{EBC} = \hat{e}(P,P)^{(\delta ab)z+(\delta ac)y+(bc)\delta^2 x}$.
6. E_B sends $\delta bP\|\mathrm{Cert}_B$ to A.
7. E_C sends $\delta cP\|\mathrm{Cert}_C$ to A.
8. A (following the protocol) computes a key $K_{AE_{B,C}} = \hat{e}(P,P)^{(a\cdot\delta b)z+(a\cdot\delta c)y+(\delta b\cdot\delta c)x} = K_{EBC}$.
9. Now E, forwarding A's messages encrypted under key $K_{EBC} = K_{AE_{B,C}}$ to B and C, and fools them into believing that A's messages come from her.

The corresponding attack on TAK-3 is similar: μ_E in Cert_E becomes δxP, and the message broadcast by E in step 3 of the above attack is now $aP\|\mathrm{Cert}_E$. Thus, the session key created in steps 5 and 8 is $K_{EBC} = K_{AE_{B,C}} = \hat{e}(P,P)^{(\delta xy)c+(\delta xz)b+(yz)a}$. The attack is otherwise identical.

This attack does not seem to apply to TAK-1 or TAK-4 because of the way in which long-term private key components are separated from the short-term components in K_{ABC} in TAK-1 and due to the use of a hash function in TAK-4.