Authenticated Three Party Key Agreement Protocols from Pairings

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

March 20, 2002

Abstract

This paper takes the pairing-based tripartite key agreement protocol of Joux [24] and develops it to produce three-party key agreement protocols offering additional security properties. We present a number of tripartite, one round, authenticated protocols related to the MTI and MQV protocols. We also present pass-optimal authenticated and key confirmed tripartite protocols that generalise the station-to-station protocol. The security properties of the new protocols are examined using heuristic methods. Applications for the protocols are also provided.

Keywords: Diffie-Hellman, tripartite, key agreement, protocol, elliptic curves, Weil pairing, Tate pairing.

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 [17], a huge number of two-party key agreement protocols have been proposed (see [6] and [30, 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* [14], [30, Chapter 12.8]. 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.

Recently, Joux [24] presented a tripartite key agreement protocol which makes use of *pairings* on elliptic curves and which requires each entity to make only a single broadcast. This should be contrasted with the obvious extension of the Diffie-Hellmann protocol to three parties, which requires two broadcasts per entity. However, just like Diffie-Hellmann, Joux's protocol is unauthenticated and suffers from man-in-the-middle attacks (see Section 3.2).

In this paper we examine in detail how to transform Joux's protocol into a protocol that can be used securely in real environments.

After definitional material in Section 2, we present Joux's protocol and the obvious attacks upon it in Section 3. Then in Section 4, we consider one round authenticated key agreement protocols for three parties. These protocols use ideas from Joux's protocol and the MTI [28] and MQV [26] protocols. Like the MTI and MQV protocols, they make use of long-term key pairs and certificates for those key pairs. We analyse a number of ad hoc attacks on our protocols, mostly inspired by previous work on the MTI and MQV protocols, and show how they can be prevented. We also compare the computational and communications efficiency of our protocols. Then in Section 5, 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 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. In our penultimate section, Section 6, we look at the scenario where one of the three parties is offline. The protocol we give for this situation can be applied to key escrow with an offline escrow agent. The final section, Section 7, contains some conclusions and ideas for future work.

It is appropriate to mention security proofs at this point. All of our security analysis is ad hoc and therefore our statements about security can be at best termed heuristic. We do benefit from the mistakes that other protocol designers have made and, we hope, avoid repeating them. But our approach is in contrast to the current trend for key exchange protocols that are provably secure (given an appropriate model and underlying hard problem), see for example [3, 4, 5, 6, 7, 12, 15, 35]. We believe this work on provable security to be useful, but still evolving, with some debate yet to be had concerning foundational matters. The results that can be proved (for now) with such techniques do not entirely capture all the attacks that might be considered realistic. As one example, the strongest, proven result for two-party authenticated key exchange in the public key setting in [6] makes the assumption that the adversary is *not* able to obtain any unconfirmed session keys. As a second example, the protocols in [6] do *not* prevent key-compromise impersonation attacks. Moreover, many of the proofs make heavy use of the random oracle assumption. We posit that for now provable security approaches must be supplemented by ad hoc analysis, and that is where we focus our security analysis in this paper.

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 [11, 33] for a protocol is defined to be a design goal that is independent of the protocol details. Here we list a number of desirable and widely-agreed extensional goals for key agreement protocols. Further discussion can be found in [30, Chapter 12], though most of the discussion there is pertinent only to the two party case.

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.

Key confirmation Here, an entity is assured that one or more other entities actually has possession of a particular key.

Explicit key authentication This goal is met when both implicit key authentication and key confirmation goals are met. This creates an *Authenticated Key Agreement with Key Confirmation* (AKC) protocol (notation is from [7]).

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 [6, 7, 26] 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 secrets 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 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.

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 [25].

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 [31] 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. Again, the requirements here may be asymmetric. Also, 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 [22]. 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, while the Internet Protocol running over a public network like the internet usually makes use of point-to-point communications. 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*. We reserve this term *non-interactive* for zero and one-pass protocols. 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 goals of a protocol are not met. 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 can delete, inject, alter or redirect messages, interleave multiple instantiations of the same protocol, and the like. Protocols for use in real-world applications should withstand active attacks.

3 Key Agreement via Pairings on Elliptic Curves

3.1 Pairings

We use the same notation as in [8]. 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 [2, 8, 9, 19, 20] 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.

We note that pairings have recently been used to create several cryptographic primitives, including ID-based encryption [8, 9] and signature schemes [16, 23, 32, 34], as well as an identity-based two-party key agreement protocol [36]. Our focus in this paper is the pairing-based tripartite key agreement protocol of Joux [24] which we will describe next.

3.2 Joux's Protocol

In [24], Joux introduced a very simple and elegant one-pass 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. 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: The ordering of protocol messages is unimportant and any of the three entities can initiate the protocol. 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 [24], 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-Hellmann problem (BDHP):

Given P, aP, bP, cP, *compute* $\hat{e}(P, P)^{abc}$.

This problem was formalised in [8]. Its hardness also forms the basis of security for the identity-based public-key encryption scheme of [8].

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, c.f. [10].

It is known that the BDHP is no harder than the computational Diffie-Hellmann 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 [8, 19] for details.

3.3 Man-in-the-Middle Attack on Joux's Protocol

Just like the unauthenticated two-party Diffie-Hellmann protocol, Joux's protocol is subject to a classic man-in-the-middle attack.

Suppose an adversary D is capable of intercepting A's communications with B and C, impersonating A to the other entities and impersonating the other entities to A. We write D_A to indicate that the adversary D is impersonating A in sending or receiving messages intended for or originating from A. Similarly, $D_{B,C}$ denotes an adversary impersonating both B and C.

Let $\delta, \delta', \delta'' \in \mathbb{Z}_q^*$ be random values of D's choice. We assume that A

initiates a run of Joux's protocol. The man-in-the-middle attack is then executed as follows:

1. $D_{B,C}$ intercepts aP from A, and D_A forwards δP to B, C.
2. D_A intercepts bP from B, and D_B forwards $\delta' P$ to A.
3. D_A intercepts cP from C, and D_C forwards $\delta'' P$ to A.

At the end of this attack, D impersonating A has agreed a key $K_{D_A BC} = \hat{e}(P,P)^{\delta bc}$ with B and C, while D impersonating B and C has agreed a second key $K_{AD_{B,C}} = \hat{e}(P,P)^{a\delta'\delta''}$ with A. If these keys are used to encrypt subsequent communications, then D, by appropriately decrypting and re-encrypting messages, can now continue his masquerade as A to B, C and B, C to A.

This attack can be extended when the adversary D has total control of the network: now D can share a separate session key with each user of the network and can masquerade as any entity to any other entity.

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 (since the messages are independent). The disadvantage is that this key is not authenticated.

We note that Joux in [24] wrongly compares his protocol to an unauthenticated two-party Diffie-Hellmann key exchange, claiming that the latter requires multiple rounds.

In this section, we develop protocols which also need just one round, but we aim to enlarge the set of security goals that are met with this round. Since key confirmation inherently requires at least an extra round of communication, none of the protocols in this section offer this property. (This can be added using by-now standard key derivation and MAC techniques if desired. The messages in the resulting protocols can be organised so as to require 6 broadcasts over 2 rounds, or 5 broadcasts over 3 rounds.) Instead, we make use of both short-term and long-term key components to provide authenticated key agreement. Thus our extensional goals are to provide good keys and implicit key authentication to all entities. Our protocols are generalisations of the MTI family of protocols [28] and the MQV protocol [26] to the setting of pairings. In fact we present a single protocol with four different methods for deriving a session key (using the messages transmitted in the protocol). These different derivations result in protocols with slightly different security attributes, and we examine these in detail. A summary is provided in Table 1.

As with the MTI 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 secret key of A. Element P is the public value and is included in order to specify which element is used to construct μ_A and the short term public values. Subtle attacks can exploit weaknesses in the registration processes used prior to generating these certificates – see section 4.1.3 for more discussion. 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 might 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:
$A \rightarrow B, C$: $aP \| \mathrm{Cert}_A$ (1)
$B \rightarrow A, C$: $bP \| \mathrm{Cert}_B$ (2)
$C \rightarrow A, B$: $cP \| \mathrm{Cert}_C$ (3)
Protocol 2 (Tripartite Authenticated Key Agreement (TAK) protocol).

Protocol description: The ordering of protocol messages is unimportant and any of the three entities can initiate the protocol. 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. 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.

TAK key generation:

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

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:

- Protocols TAK-1, TAK-2 and TAK-3 have their roots in the MTI protocols.
- Protocol TAK-4 is modelled on the MQV protocol but avoids that protocol's unknown key-share weakness [25] by using a cryptographic hash function H (whose output is assumed to be onto $\mathbb{Z}_q^*$) to combine long-term and short-term secret keys. It is the only one of our four protocols to explicitly require the use of a hash function. Note that the protocol resulting from omission of this hash function produces the key $K_{ABC} = \hat{e}(P,P)^{(a+x)(b+y)(c+z)}$, which is the product of the three keys K_{ABC} from TAK-1, TAK-2 and TAK-3. However, this version of the protocol suffers from an unknown key-share weakness similar to that presented for the MQV protocol in [25], 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 4.1.3 for further discussion of unknown key-share attacks.
- 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 $\mathbb{G}_2$.
- In all cases except the last, the key K_{ABC} could also be computed by concatenating the individual pairing 'components' (e.g. $\hat{e}(P,P)^{abc}$ and $\hat{e}(P,P)^{xyz}$ in TAK-1) and hashing rather than by multiplying components. This requires that the three parties are able to order the components in the same way. This can prevent certain attacks based on the algebraic structure of the various K_{ABC}. Where applicable, we note where this is so in our security analysis below.
- In any case, it would be prudent to derive session and MAC keys (for key confirmation if desired) by applying a hash function to K_{ABC} (and possibly other information).
- In all the protocols, every message should include a protocol name and version number to avoid accidental mixing of messages from different types of protocols. We have omitted these for simplicity of presentation.

- Since all four keys are created after transmitting the same protocol messages, the communication overhead of each protocol version is identical, standing at a single broadcast per entity. 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 $\hat{e}(P,P)^{xyz}$ and this can be re-used until expiration of long-term keys. 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, c should be used in each new protocol session.

Our TAK protocols prevent man-in-the-middle attacks of the type introduced in the previous section. This is because the way in which long-term secret keys are involved in the computation of each K_{ABC} prevents an adversary not in possession of a long-term secret key from obtaining these session keys. However, other forms of active attack can still occur. We consider such attacks next on a case-by-case basis.

4.1 Attacks on TAK Protocols

We present a variety of attacks on our TAK protocols. These are mostly inspired by earlier attacks on the two-party MTI protocols. Some of the attacks are preventable, and others require rather unrealistic scenarios. However, all of the attacks are important as they determine the security attributes of our various protocols. We summarise these attributes in Table 1.

4.1.1 Two Key-Compromise Attacks on TAK-1

The first of our attacks is a very serious attack on TAK-1, requiring the adversary D to obtain just a session key and one of the short-term secret keys used in a protocol run, and after which D is capable of impersonating any of the other entities in subsequent protocol runs. This attack is more severe than a key-compromise impersonation attack because it does not require the adversary to learn a long-term secret key.

The pre-requisites for the attack are:

1. D, by eavesdropping on a protocol run, has obtained the short-term public values bP and cP.
2. D has also obtained the session key $K_{ABC} = \hat{e}(P,P)^{abc+xyz}$ agreed in that protocol run.
3. D has also somehow acquired the short-term key a used in that run.

Adversary D can now calculate $K_{ABC} \cdot \hat{e}(bP, cP)^{-a} = \hat{e}(P,P)^{xyz}$. D can then go on to impersonate *any* of A, B or C in subsequent protocol runs. She does this simply by sending Cert_A, Cert_B or Cert_C along with her chosen short-term public value δP. Knowledge of $\hat{e}(P,P)^{xyz}$ and δ allows her to compute

session keys agreed in subsequent protocol runs. By symmetry, this attack can be mounted once D is in possession of any short-term secret key.

This attack is prevented by using a hash function to perform key derivation: now the session key revealed is the *hash* of K_{ABC} rather than K_{ABC} itself.

Another version of the attack occurs whenever D can obtain the long-term secret key of one of the entities, and as such is a key-compromise impersonation attack in the taxonomy of Section 2.2. Suppose D has obtained x, A's long-term secret key. Then D can calculate $\hat{e}(P,P)^{xyz}$ using x and public data in B and C's certificates. Knowledge of this value now allows D to impersonate any entity (not just A), as above. This attack cannot be prevented by session key derivation.

These attacks do not appear to apply to TAK-2, TAK-3 or TAK-4 because of the way in which the long-term components are combined with short-term key components in K_{ABC}.

One way to defeat key-compromise attacks on MTI-like protocols in general is for each entity to use his long-term secret key as an ECDSA signature key to sign short-term keys [1]. 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.

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

A protocol is not forward secure if the compromise of the long-term secret keys of one or more entities also allows an adversary to obtain session keys previously established between honest entities. With the key $K_{ABC} = \hat{e}(P,P)^{abc+xyz}$ as in protocol TAK-1, this weakness is not present, no matter how many long-term secret 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 secret keys in TAK-3, or all three long-term secret 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. (Note that hashing the concatenation of key components does not help here).

4.1.3 Unknown Key-Share Attacks

Basic Source Substitution Attacks on TAK-1 to TAK-4:

This is a practical attack, which utilizes a potential registration weakness for public keys to create fraudulent certificates [29]. Adversary D registers

A's public key μ_A as her own, creating $\text{Cert}_D = (\mathcal{I}_D \| \mu_A \| P \| \mathcal{S}_{\text{CA}}(\mathcal{I}_D \| \mu_A \| P))$. Then she intercepts A's message $aP \| \text{Cert}_A$ and replaces Cert_A with her own certificate Cert_D. Note that D 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 D, when in fact they have agreed a key with A. They will interpret any subsequent encrypted messages emanating from A as coming from D. 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 below.

Second Source Substitution on TAK-2 and TAK-3:

Even if the CA does the previous check, the adversary can still attack protocols TAK-2 and TAK-3: she obtains a Cert_D from the CA which contains a component μ_D which is a multiple of μ_A, and alters 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.

We next present in detail the attack on TAK-2, and then sketch the attack on TAK-3.

1. A sends $aP \| \text{Cert}_A$ to $D_{B,C}$.
2. D computes $\mu_D = \delta^2 \mu_A = \delta^2 xP$ and registers μ_D as part of her Cert_D.
3. D initiates a run of protocol TAK-2 by sending $\delta aP \| \text{Cert}_D$ to B, C.
4. B sends $bP \| \text{Cert}_B$ to D, C; C sends $cP \| \text{Cert}_C$ to D, B.
5. B and C (following the protocol) compute
$$K_{DBC} = \hat{e}(P,P)^{(\delta ab)z + (\delta ac)y + (bc)\delta^2 x}.$$
6. D_B sends $\delta bP \| \text{Cert}_B$ to A.
7. D_C sends $\delta cP \| \text{Cert}_C$ to A.
8. A (following the protocol) computes a key
$$K_{AD_{B,C}} = \hat{e}(P,P)^{(a \cdot \delta b)z + (a \cdot \delta c)y + (\delta b \cdot \delta c)x} = K_{DBC}.$$
9. Now D, forwarding A's messages encrypted under key $K_{DBC} = K_{AD_{B,C}}$ 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_D in Cert_D becomes δxP, and the message broadcast by D in step 3 of the above attack is now $aP \| \text{Cert}_D$. Thus, the session key created in steps 5 and 8 is $K_{DBC} = K_{AD_{B,C}} = \hat{e}(P,P)^{(\delta xy)c + (\delta xz)b + (yz)a}$. 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.

Unlike the unknown key-share attack on the MQV protocol [25], the adversary in our attacks does not know his 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 [21] 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.

4.1.4 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, D 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 D, upon obtaining one of them, gets keys for two subsequent sessions as well.

A is convinced to initiate three sessions with D:

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

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

$D_B \to A$: $a'P \| \mathrm{Cert}_B$ (2^{alpha})
$D_C \to A$: $a''P \| \mathrm{Cert}_C$ (3^α)

Similarly the second session is completed by $D_{B,C}$ 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.

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

This 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-known for other protocols (see the comments following [6, Theorem 11] for an example). Two-party protocols closely related to TAK-2 and TAK-3 are conjectured to be provably secure in [6].

4.1.5 Triangle Attack on TAK-3

The class of triangle attacks were introduced by Burmester [13]. They are usually somewhat theoretical in nature. Our triangle attack on TAK-3 allows an adversary D (who has a certificate Cert_D containing $\mu_D = \Delta P$) to compute a session key K_{ABC} previously shared by the honest parties A, B and C.

1. D eavesdrops to obtain aP, bP and cP from the session in which the session key $K_{ABC} = \hat{e}(P,P)^{(xy)c+(xz)b+(yz)a}$ is agreed between A, B, C.

2. D now initiates three protocol runs. The first one is:
   $D \to B, C$: $aP \| \text{Cert}_D$ (1^α)
   $B \to D, C$: $b'P \| \text{Cert}_B$ (2^α)
   $C \to D, B$: $c'P \| \text{Cert}_C$ (3^α)
   The session key agreed is $K_{DBC} = \hat{e}(P,P)^{(\Delta y)c'+(\Delta z)b'+(yz)a}$.

3. The second run is:
   $D \to A, C$: $bP \| \text{Cert}_D$ (1^β)
   $A \to D, C$: $a''P \| \text{Cert}_A$ (2^β)
   $C \to A, D$: $c''P \| \text{Cert}_C$ (3^β)
   The session key agreed is $K_{ADC} = \hat{e}(P,P)^{(x\Delta)c''+(xz)b+(\Delta z)a''}$.

4. And lastly:
   $D \to A, B$: $cP \| \text{Cert}_D$ (1^γ)
   $A \to B, D$: $a'''P \| \text{Cert}_A$ (2^γ)
   $B \to A, D$: $b'''P \| \text{Cert}_B$ (3^γ)
   The agreed session key is $K_{ABD} = \hat{e}(P,P)^{(xy)c+(x\Delta)b'''+(y\Delta)a'''}$.

5. Finally, session key

$$\begin{aligned} K_{ABC} &= K_{DBC} \cdot \hat{e}(P,P)^{-(\Delta y)c'-(\Delta z)b'} \\ &\quad \cdot K_{ADC} \cdot \hat{e}(P,P)^{-z\Delta c'''-\Delta z a'''} \\ &\quad \cdot K_{ABD} \cdot \hat{e}(P,P)^{-x\Delta b'''-y\Delta a'''} \end{aligned}$$

   can now be computed by D.

This triangle attack is possible because of the algebraic relationship between the long and short term key components in K_{ABC}. It can be thwarted using appropriate key derivation. This attack does not work on TAK-1 and TAK-2 because we can not isolate individual short term key components (e.g. in step 2 we cannot isolate a from fresh components b' and c'). This type of attack is also eliminated in TAK-4 because of the binding of each entity's short and long-term key using a hash function.

4.1.6 Security Summary

While we have not been able to exhaustively examine attacks on our protocols, and only used informal arguments in considering their security. Table 1 compares the security attributes 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, we prefer the use of protocol TAK-4 (which requires hashing) or protocol TAK-2. If perfect forward security is also required, then we recommend the use of TAK-3 modified as described in Section 4.1.2 along with pre-computation, instead of the similarly modified TAK-2. TAK-4 has the additional benefit of being the most computationally efficient of all our one round protocols. Of course, robust certification is needed for all of our TAK protocols in order to avoid unknown key-share attacks.

	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
KC impersonation secure	n/a	No	Yes	Yes	Yes
Unknown key-share secure	No	Yes$^{(iii)}$	Yes$^{(iv)}$	Yes$^{(iv)}$	Yes$^{(iii)}$

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 secret keys, but still forward secure for a compromise of two or less such keys.

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

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

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

5 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 does require at least six passes. We then compare our protocol to a tripartite version of the station-to-station protocol [18].

5.1 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 [27]. 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.

5.2 A Six Pass Diffie-Hellmann Based AKC Protocol

The station-to-station (STS) protocol is a three pass, two-party AKC protocol designed by Diffie, van Oorschot and Wiener [18] 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:

$A \to B$: $g^a \| \text{Cert}_A$ (1)

$B \to C$: $g^a \| \text{Cert}_A \| g^b \| \text{Cert}_B \| g^{ab}$ (2)

$C \to A$: $g^b \| \text{Cert}_B \| g^c \| \text{Cert}_C \| g^{bc} \| E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A \| \mathcal{I}_B \| \chi))$ (3)

$A \to B$: $g^c \| \text{Cert}_C \| g^{ac} \| E_{K_{ABC}}(\mathcal{S}_C(\mathcal{I}_A \| \mathcal{I}_B \| \chi)) \| E_{K_{ABC}}(\mathcal{S}_A(\mathcal{I}_B \| \mathcal{I}_C \| \chi))$ (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 $K_{ABC} = g^{abc} \bmod p$.

5.3 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 the apparent advantages that a protocol enjoys can disappear when one considers 'useful' versions of protocols and an appropriate model of communications.

6 Tripartite Protocols with One Offline Party

As we mentioned in the introduction, there is 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 offline, 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 which applied to TAK-2. We assume that C is the offline party and that C's certificate Cert_C is pre-distributed, or is readily available to A and B.

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 are

$$\begin{aligned} K_A &= \hat{e}(bP, zP)^a \cdot \hat{e}(yP, zP)^a \cdot \hat{e}(bP, zP)^x \\ K_B &= \hat{e}(aP, zP)^b \cdot \hat{e}(xP, zP)^b \cdot \hat{e}(aP, zP)^y \end{aligned}$$

and when required,

$$K_C = \hat{e}(aP, yP)^z \cdot \hat{e}(xP, bP)^z \cdot \hat{e}(aP, bP)^z.$$

Thus, all parties can share the session key $K_{ABC} = \hat{e}(P, P)^{(ab)z+(a)yz+(b)xz}$.

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, except when long-term secret key z is compromised. Here z can be viewed as an independent master key, which can be regularly updated.

7 Conclusion

We have constructed a suite of tripartite, authenticated key agreement protocols from pairings. For a single round protocol, our heuristic analysis suggests that TAK-4 is the most secure, followed by TAK-2. The latter protocol avoids explicit use of hash functions. In addition, both protocols appear to be robust in the situation when one party was offline.

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-Hellmann style protocols.

While we have considered variants of all the well-known attacks on our protocols, we cannot guarantee that room for new attacks has not been introduced. Notwithstanding our remarks on proofs of security in the introduction, it would clearly be desirable to develop appropriate models for security of conference key

agreement protocols and find pairing-based protocols that are provably secure in that setting. The work of [12, 15] provides an excellent start in this direction.

8 Acknowledgement

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

References

[1] American National Standards Institute — ANSI X9.62. Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA), 1999.

[2] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. Cryptology ePrint Archive, Report 2002/008, 2002. `http://eprint.iacr.org/`.

[3] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the 30th Annual Symposium on the Theory of Computing*, pages 419–428. ACM, 1998. `http://www.cs.ucsd.edu/users/mihir/papers/key-distribution.html/`.

[4] 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 *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1994.

[5] 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.

[6] S. Blake-Wilson and A. Menezes. Security proofs for entity authentication and authenticated key transport protocols employing asymmetric techniques. In B. Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 137–158. Springer Verlag, 1997.

[7] 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 *Lecture Notes in Computer Science*, pages 339–361. Springer Verlag, 1998.

[8] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing, 2001. `http://www.crypto.stanford.edu/~dabo/abstracts/ibe.html`.

[9] 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 *Lecture Notes in Computer Science*, pages 213–229. Springer Verlag, 2001.

[10] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellmann and related schemes. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 129–142. Springer Verlag, 1996.

[11] 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/`.

[12] 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*, Lecture Notes in Computer Science. Springer-Verlag, to appear.

[13] 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 *Lecture Notes in Computer Science*, pages 308–317. Springer-Verlag, 1994.

[14] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. De Santis, editor, *Advances in Cryptology EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer-Verlag, 1995.

[15] 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 *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001.

[16] J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. Cryptology ePrint Archive, Report 2002/018, 2002. `http://eprint.iacr.org/`.

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

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

[19] 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 *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001. `http://www.citeseer.nj.nec.com/galbraith00supersingular.html/`.

[20] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing, 2002. To appear.

[21] O. Goldreich. Randomness, interactive proofs, and zero-knowledge – a survey. In R. Herken, editor, *The Universal Turing Machine: A Half Century Survey*, pages 377–405. Oxford University Press, 1988.

[22] L. Gong. Efficient network authentication protocols: Lower bounds and optimal implementations. *Distributed Computing*, 9(3):131–145, 1995.

[23] 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/`.

[24] 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 *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[25] 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.

[26] 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.

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

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

[29] 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.

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

[31] C. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, 34:980–981, 1998. `http://www.isg.rhul.ac.uk/cjm/Publications.htm`.

[32] K.G. Paterson. ID-based signatures from pairings on elliptic curves. Cryptology ePrint Archive, Report 2002/004, 2002. `http://eprint.iacr.org/`.

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

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

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

[36] N.P. Smart. An identity based authenticated key agreement protocol based on the weil pairing. Cryptology ePrint Archive, Report 2001/111, 2001. `http://eprint.iacr.org/`.